Decision Support Systems 51 (2011) 562–575

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Constraint-centric workflow change analytics

Harry Jiannan Wang [a,*], J. Leon Zhao [b]

[a] Department of Accounting and MIS, University of Delaware, Newark, DE, United States
[b] Department of Information Systems, City University of Hong Kong, Kowloon, Hong Kong, China

ARTICLE INFO

Article history:
Received 5 March 2010
Received in revised form 7 February 2011
Accepted 1 March 2011
Available online 5 March 2011

Keywords:
Workflow modeling
Workflow constraint
Workflow changes
First order logic
Business process management
Change management

ABSTRACT

In a globalized economic environment with volatile business requirements, continuous process improvement needs to be done regularly in various organizations. However, maintaining the consistency of workflow models under frequent changes is a significant challenge in the management of corporate information services. Unfortunately, few formal approaches are found in the literature for managing workflow changes systematically. In this paper, we propose an analytical framework for workflow change management through formal modeling of workflow constraints, leading to an approach called Constraint-centric Workflow Change Analytics (CWCA). A core component of CWCA is the formal definition and analysis of workflow change anomalies. We operationalize CWCA by developing a change anomaly detection algorithm and validate it in the context of procurement management. A prototype system based on an open-source rule engine is presented to provide a proof-of-concept implementation of CWCA.

1. Introduction

To be competitive in the global market, companies need to quickly adapt their business processes to various changes in the business environment, such as mergers/acquisitions, new regulations, and new customer demand. Various changes can occur in different workflow perspectives including control flow, data flow, organizational model, and workflow constraints. For instance, process reengineering or supply chain reconfiguration can lead to the alteration of task execution sequences and removal of non-value-added tasks, i.e. control flow changes. Mergers/acquisitions can result in resource reallocation and organization restructuring, i.e. organizational model changes. New governmental regulations such as the Sarbanes-Oxley Act may require new and revised business rules to be compliant, i.e. workflow constraint changes.

Existing research on workflow changes tends to focus on a limited number of perspectives of workflow, mostly control flow and data flow, paying little attention to dependencies among all workflow perspectives [13,15,22,29,40,44,45,53]. For example, industrial reorganization such as mergers/acquisitions may result in removing certain organizational roles, i.e. an organizational model change. Consequently, tasks assigned to those roles must be delegated to other resources, and the relevant workflow constraints must be revised properly; otherwise, a runtime role resolution error might occur and the corresponding tasks will not be executed. Given these frequent changes in business workflows, managing multi-perspective workflow changes is imperative in order to support business operations and continuous process improvement efficiently.

Many process modeling specifications have been used in practice, such as UML activity diagrams, BPMN, and Event-driven Process Chains (EPCs), but these specifications usually lack analytical capability and therefore cannot be used to formally model and analyze workflow changes. Formal languages have been applied to model workflows, such as Petri nets [1,34,52], Metagraphs [5], Communicating Sequential Processes (CSP) [58], formal logic [10,21], and PI Calculus [46]. In addition, several rule languages have been applied to analyze workflow constraints and specify workflow exceptions [9,13–15]. Nevertheless, little research is found on formal approaches that focus on workflow change analysis treating a workflow system as a whole.

In this paper, we aim to fill this research gap by proposing an analytical framework for managing multi-perspective workflow changes via formal modeling of workflow constraints, referred to as Constraint-centric Workflow Change Analytics (CWCA). The contributions of this paper are as follows. First, we propose a novel constraint-centric approach to analyzing multi-perspective workflow changes by specifying workflow change operations and dependencies formally. Second, we apply First Order Logic to formally define workflow change anomalies and develop an algorithm to detect those anomalies. Third, we validate the CWCA framework through a prototype system that provides insights into the integration of CWCA with existing workflow management systems and rule engines.

The rest of the paper proceeds as follows. We first review the relevant literature in Section 2. Then, we discuss the types of workflow changes and their dependencies in Section 3. In Section 4, we present a procurement process as a running case for the paper and propose a constraint-centric workflow modeling framework to specify different workflow perspectives for formal workflow change analysis. In Section 5, we formally define and analyze workflow change anomalies

* Corresponding author.
E-mail addresses: hjwang@udel.edu (H.J. Wang), jlzhao@cityu.edu.hk (J.L. Zhao).

0167-9236/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2011.03.001

and develop an anomaly detection method. A proof-of-concept system is also presented to demonstrate CWCA and validate the anomaly detection algorithm. We compare our framework with some other related approaches and discuss its limitations in Section 6. Finally, we conclude in Section 7 by summarizing our contributions.

2. Literature review

The structural perspective of workflow modeling is captured by control flow [49]. A control flow model usually indicates the tasks, their execution sequences, and the corresponding transition conditions. There have been extensive research efforts on control flow modeling, resulting in many process modeling methods. UML activity diagrams, EPCs, and BPMN are widely adopted graphical business process modeling standards. However, they lack a rigorous mathematical foundation and therefore have limited analytical capability. Petri nets have been used to represent and analyze process models [1,52]. A number of Petri nets extensions have been proposed including timed Petri nets and colored Petri nets, which have been used to model data flow, temporal constraints, and workflow events [8,28,34]. Many workflow structural properties such as reachability, deadlock, and livelock, can be formally verified using Petri nets. Tools have also been developed to support Petri-nets-based workflow modeling and simulation such as CPN Tools (http://wiki.daimi.au.dk/cpntools) and YAWL (http://www.yawl-system.com/). Metagraphs are another rigorous process modeling approach with mathematical foundation and strong analytical capability [5]. Metagraphs provide three different views of a process model, namely, task view, data view and resources view and are able to analyze interactions among those three views via matrix computations [5]. Various logic formalisms have also been applied to workflow modeling and analysis, such as propositional logic, temporal logic, event algebra, and concurrent transaction logic [10,21,38]. One unique feature of logic-based approaches is the capability to model and enforce various constraints on task dependency and execution orders. Communicating sequential process (CSP) [58] and Pi-calculus [46] have also been applied to represent workflow, where model checking techniques can be used to verify certain workflow properties.

Organizational modeling in workflow has been identified as an important research area in business process management [7], which provides the organizational context of workflow applications. Several organizational meta-models have been proposed to make workflow management systems more "organizational aware" [61]. The specification and validation of data flow in workflow system is critical, because data flow anomalies may prevent workflows from proper execution if not detected prior to workflow deployment [45,50]. Specifically, Sun et al. (2006) identify three types of data anomalies, namely missing data, redundant data, and conflicting data, and propose a data anomaly verification algorithm. The proper execution of workflows requires authorization constraints to enforce the assignment of tasks to organizational resources, such as human users, roles, organization units, or machine agents [15]. Different mechanisms have been proposed to specify and enforce workflow authorization constraints, such as Secure Petri Nets [3], logic-based constraint specification [9] and ECA (event-condition-action) rules [15].

Curtis et al. proposed four perspectives in process representation, namely, functional, informational, organizational, behavioral perspectives [19]. Functional perspective models what process elements are being executed, and what flows of informational entities, are relevant to the process elements. Informational perspective specifies the informational entities produced or manipulated by a process. Organizational perspective defines where and by which agents in the organizational model that process elements are performed. Behavioral perspective concerns with when and how process elements are performed through different workflow structures. In this paper, the four perspectives are substantiated via control flow, data flow, and organizational model, and workflow constraints, which are well-known concepts in workflow modeling. More specifically, control flow is related to both the functional and behavioral perspectives; data flow mainly models the informational perspective, organizational model specifies the organizational perspective; and workflow constraints serve as the glue to all four perspectives.

Another related research area is business rules research. Many rule representation languages have been developed to express business rules, enable rule reasoning, and facilitate rule extraction, reuse and integration [14,41]. There have been extensive implementation efforts in rule engines and development tools, such as JESS, CLIPS, and Drools. As a key enabling technology for process management, workflow technology has been applied in services computing research areas such as web services orchestration and choreograph [60], web-service-based process integration [11], and web-service-enabled process management for collaborative commerce[16]. A number of languages have been proposed to describe the process models of web services, such as Business Process Execution Language (BPEL) for web service orchestration [2], Web Services Choreography Description Language (WS-CDL) for web service choreography [30], XML Process Description Language (XPDL) for business process definition interchangeability [57], and First-order Logic Ontology for Web Services (FLOWS) [51]. In particular, FLOWS is developed by extending the Process Specification Languages (PSL) [25], which is based on First Order Logic. The goal of FLOWS is to enable reasoning about web service semantics by providing a fully expressive language and framework for modeling semantic aspects of service behavior [26]. Nevertheless, none of the languages list above is designed for analyzing multi-dimensional workflow changes. Our research in this paper leverages First Order Logic to model workflow changes and change consequences among different workflow perspectives, which can be incorporated into other languages and frameworks, e.g. FLOWS, to provide analytical capability for handling workflow changes.

Organizations are recognizing that workflow management systems must be able to adapt efficiently to changes in business in order to realize the real power of process automation [29]. As such, research in dynamic and adaptive workflow has received much attention [12,40]. Sadiq et al. (2000) found that changes to workflow models are often permanent as the result of process improvement, process reengineering, merger/acquisitions, etc., whereas changes in workflow instances are usually due to unforeseen and rare situations in process operations. They defined five workflow modification policies to cope with workflow changes, namely, Flush, Abort, Migrate, Adapt, and Build [44]. Change management involves thorough analysis of process structure and current process status in order to avoid process errors known as "dynamic change bugs" [53]. Several mathematical models have been proposed to formally represent workflow dynamic changes and identify "safe" ways to migrate existing instances without incurring dynamic change bugs [22,40,53]. Change adaptation means that there are some instances that need to be treated differently due to exceptions defined as "deviations from an ideal collaborative workflow process caused by errors, failures, resources or requirements changes" [31]. Different methods for exception specification and handling have been proposed in the literature, such as knowledge-based approach [31], active rule based approach [13], and meta modeling approach [17].

In sum, most previous works have focused on some limited perspectives of workflow changes such as those in terms of control flow and data flow, leaving out other important perspectives such as changes in organizational models and workflow constraints. In addition, research on formal approach to analyzing multi-perspective workflow changes has been scant. As such, systematic and comprehensive multi-perspective change management features are virtually not found in existing workflow management systems.

3. Workflow changes and dependencies

In this section, we first discuss various changes in different workflow perspectives, including control flow, data flow, organizational model, and workflow constraints. Then, we show how workflow consistency

can be affected by the interactions among those changes, which we refer to as change dependencies.

3.1. Workflow perspectives and changes

A workflow model can be studied from four perspectives, namely, control flow, data flow, organizational model, and workflow constraints [49]. Control flow concerns the execution orders of tasks. Five basic routing patterns, namely, sequence, fork, join, branch, and merge can be used to model most commonly used business processes, as shown in Fig. 1. *Sequence* means that tasks are executed sequentially. *Fork* and *Join* are used to model parallel routing. *Branch* and *Merge* are used to model conditional routing. More thorough studies of control flow constructs can be found on workflow patterns [54]. Data flow describes data items that are consumed, produced and transferred by tasks and routing constructs and their dependencies. Organizational model defines workflow-related resources, such as users, roles, organization units, machine agents, and applications, and their relationships. Workflow constraints serve as a hub that connects the aforementioned three perspectives by specifying rules for workflow execution, such as routing rules at branching nodes, assigning resources to tasks, and relating data items to tasks.

At the conceptual level, various workflow changes can be classified into three basic operations to different workflow perspectives, namely, *insertion*, *deletion*, and *modification*. Some workflow change examples using this classification are listed in Table 1. Given different workflow perspectives are related to one another, changes in one perspective can have significant impact on other perspectives. For instance, removing an organizational resource can cause all tasks depending on that resource unable to complete, i.e. an organizational model change leading to a control flow anomaly. Such workflow change anomalies must be detected and corrected to avoid costly run-time workflow errors. As aforementioned, control flow, data flow, and organizational model are connected to one another via workflow constraints, and therefore changes in one workflow perspective can only affect other perspectives through workflow constraint changes, i.e. *multi-perspective workflow change anomalies are essentially workflow constraint anomalies*. As a result, we suggest treating formal analysis of change dependencies between workflow constraints and other workflow perspectives as the core issue of workflow change management. This idea lays the foundation of our CWCA approach and is a cornerstone of our research.

3.2. Workflow change dependencies

Fig. 2 shows dependencies between workflow constraints and other three workflow perspectives. These dependencies illustrate how changes in one workflow perspective can affect workflow constraints, leading to workflow change anomalies.

- *Workflow Change Dependency 1* (*C–W Change Dependency*). Control flow changes can result in workflow constraint anomalies. In particular, insertion, deletion, and modification of tasks or routing constructs in a control flow can all lead to anomalies. For example, adding a new task requires new constraints to assign resources to the new task, and otherwise the new task cannot be executed. Deleting a routing construct such as a Branch or changing a Branch into a Fork construct can make the related routing constraints redundant.
- *Workflow Change Dependency 2* (*D–W Change Dependency*). Removing or modifying existing data items can cause workflow constraint anomalies. Constraints defined on data items may become invalid or redundant when data items change. For example, a routing constraint may define that if the value of a data item d_1 is greater than 1000 then execute task t_1. If d_1 is removed or the name of d_1 is modified, this routing constraint becomes invalid. Adding data items does not affect existing workflow constraints because we assume that new data items are not specified in the existing workflow constraints.
- *Workflow Change Dependency 3* (*O–W Change Dependency*). Workflow constraint anomalies can occur when removing or modifying existing resources in an organizational model. A control flow is tied to the organization model via resource assignment constraints. When an organizational model changes, workflow constraints are also subject to change in order to be consistent. For instance, a constraint may state that task "Purchasing" is handled by a "Purchasing Clerk". This constraint becomes invalid and needs to be revised if the role "Purchasing Clerk" is either removed or renamed due to organization restructuring.
- *Workflow Change Dependency 4* (*W–W Change Dependency*). Workflow constraints can relate to one another, and therefore changes in one constraint may have impact on other constraints. For example, a "binding of duties" constraint may specify that task "Prepare documents" and task "Send documents" must be handled by the same person [14]. This constraint can never be evaluated to true if the roles assigned by authorization constraints to both tasks do not have any users in common.

The four workflow change dependencies further illustrate that workflow constraint perspective is the center of workflow changes. Given that control flow, data flow, and organizational model are connected via workflow constraints, the four types of workflow changes provide a complete classification of workflow changes. Note that there may be change dependencies within a single workflow perspective, e.g. one data item depends on another data item. These intra-perspective workflow changes are not the focus of this paper and thus are not emphasized in this paper. In order to formally analyze workflow

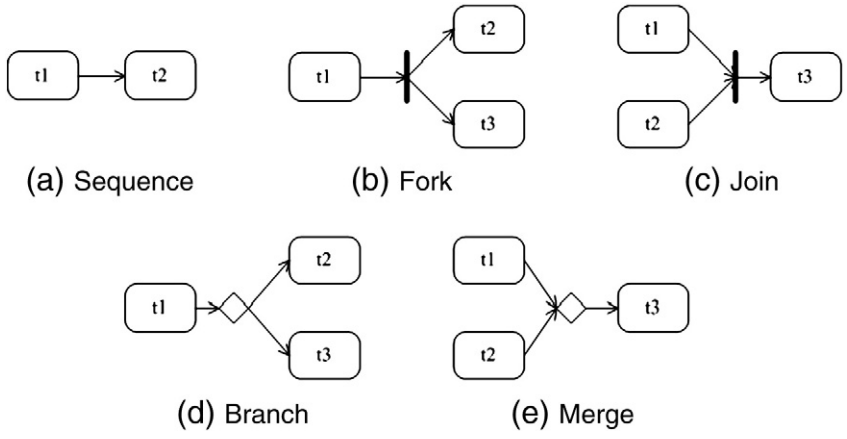

Fig. 1. Basic routing patterns.

Table 1
Examples of workflow changes.

	Insertion	Deletion	Modification
Control flow	• Adding tasks • Adding a routing constructs	• Remove tasks • Remove routing constructs	• Change task execution order • Change sequential execution into parallel execution
Organizational Model	• Hire new employee • Create new positions • Form new departments	• Employee terminations • Remove positions • Remove departments	• Reassign employees to a new positions • Restructure the organization
Data flow	Adding new data item	Remove data item	Update data item values
Workflow Constraints	• Specify new routing rules • Assign additional resources to tasks	• Remove resources from task execution • Remove routing constraints	• Revise routing conditions • Revise resource assignment rule

constraint changes and anomalies, we propose a constraint-centric workflow modeling framework as presented next.

4. Constraint-centric workflow modeling

In this section, we first present a procurement management case to illustrate various workflow constraints. Then, we formally define the four workflow perspectives, i.e. control flow, data flow, organizational model, and workflow constraint, using First Order Logic, which provides a formal framework for analyzing workflow changes.

4.1. A procurement management case

Procurement is a very important step in supply chain management, which involves many functional areas such as purchasing, inventory management, and warehouse operations [18]. A procurement workflow is given as a UML activity diagram in Fig. 3. This workflow is developed by simplifying the procurement process from SAP process reference models presented in [18].

The tasks in this procurement workflow are described in Table 2. Related data items include: d_1 — supervisor approval, d_2 — item total value, d_3 — available fund, d_4 — mgmt approval, d_5 — requested item quantity, and d_6 — requested item inventory. Fig. 4 shows the organizational model associated with this workflow. The employees for each role are specified as follows: GM (John), ED (Joe), PD (Jason), AD (Maggie), SE (Eric, Ray), PC (Peter), IC (Sam), DC (Dan, Jack), and AC (Steve, Ben).

To ensure proper workflow execution, workflow constraints must be specified to enforce business and operational rules. Fig. 5 lists the constraints for the procurement workflow. Constraints c_1 through c_8 are resource assignment constraints. Constraint c_9 is a separation of duties constraint, whereas c_{10} is a binding of duties constraint. Constraints c_{11} through c_{20} are routing rules for the five branch constructs in the process. Constraints c_{21} through c_{27} specify the input data for the branch constructs.

Next, we present a workflow specification framework based on First Order Logic, which is used to represent the procurement process and analyze workflow constraint anomalies.

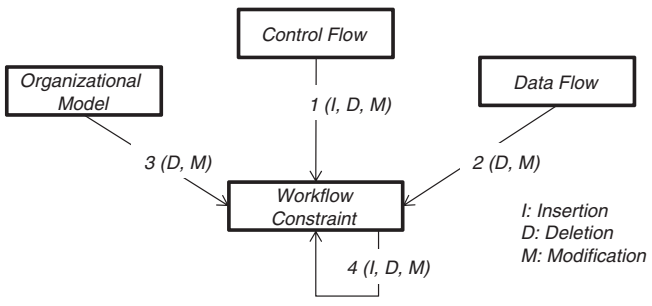

Fig. 2. Workflow change dependencies.

4.2. Constraint-centric workflow specification

Given the workflow change dependencies shown in Fig. 2, we need a unified representation for the four workflow perspectives in order to formally analyze workflow changes. In this paper, we extend previous research on workflow constraint specification by means of First Order Logic. We choose First Order Logic for several reasons: first, it has been widely used to present declarative knowledge [24,42], which is a natural choice for specifying workflow constraints. Second, it is sufficiently expressive to represent control flow, data flow, and organizational model by defining appropriate predicates with formal workflow semantics. Third, it enables us to specify formally workflow changes. Lastly, we can leverage logic programming and the wide array of rule engines to support the implementation of our CWCA approach.

The four workflow perspectives are formally defined as follows.

Definition 1. Control Flow

Control flow is a 5-tuple $<T, start, end, RC, L>$, where

- $T = \{t \mid t$ is a task of the workflow$\}$,
- *start* is the start node of the workflow,
- *end* is the end node of the workflow,
- $RC = \{rc \mid rc \in \{f, j, b, m\}\}$ is the set of routing constructs where f (fork) and j (join) are used to represent parallel execution and synchronization of a set of tasks, and b (branch) and m (merge) are used to choose one task from a set of tasks for execution.
- $L = \{next(x, y) \mid x, y \in T \cup \{start, end\} \cup RC$ and x precedes $y\}$ is the set of links in the control flow.

The control flow perspective defines the tasks, routing constructs, start and end nodes, and the task execution orders. Given Definition 1, the control flow of the procurement process can be formally represented as the set of logic facts and predicates in Fig. 6.

Definition 2. Data Flow

Data flow is a 3-tuple $<D, I, O>$, where

- $D = \{d \mid d$ is a data item used in the workflow$\}$,
- $I = \{input(n,d) \mid d \in D$ and $n \in T \cup \{start, end\} \cup RC\}$ is the set of predicates to specify input data to process nodes,
- $O = \{output(n, d) \mid d \in D$ and $n \in T \cup \{start, end\} \cup RC\}$ is the set of predicates to specify output data to process nodes.

The data flow perspective concerns what data objects are related to the workflow and what are inputs and outputs for different process nodes.

Definition 3. Organizational Model

Organizational model is a 3-tuple $<R, B, M>$, where

- $R = U \cup RO$ is the set of organizational resources, $U = \{u \mid u$ is a user in the organization$\}$, $RO = \{ro \mid ro$ is a role or position in the organization,
- $B = \{belong(a, b) \mid a \in U, b \in RO\}$ specifies user a has role b,
- $M = \{manage(j, k) \mid j, k \in R\}$ specifies resource j manages resource k.

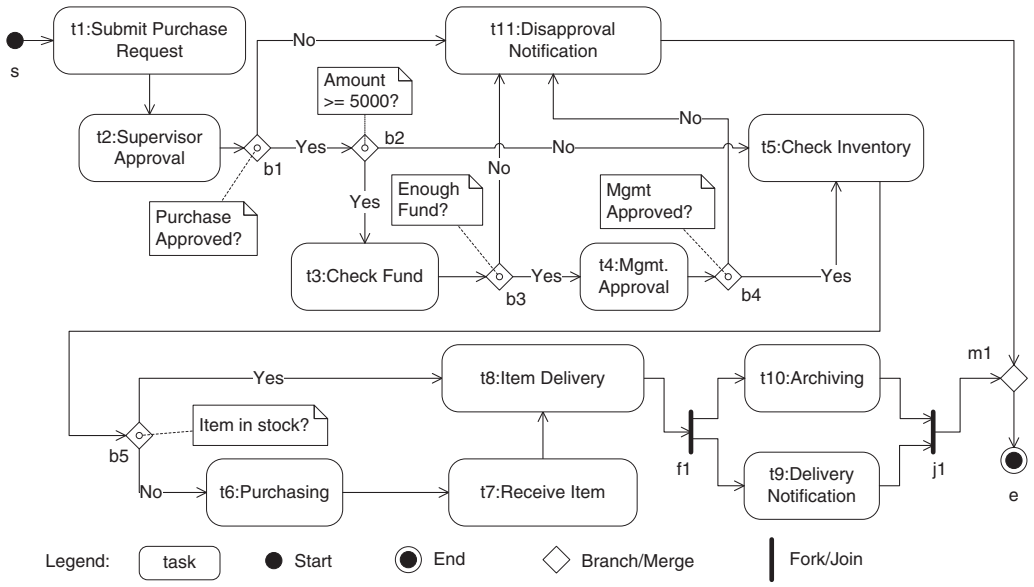

Fig. 3. A procurement workflow.

The organizational perspective defines organizational resources related to the workflow and their relationships. An organizational resource is defined as an entity that is capable of doing work, which may be either human or non-human [43]. In Definition 3, human resources are further classified into users, roles, and groups, whereas non-human resources are called agents. Predicates *belong*(*a*, *b*) and *manage*(*j*, *k*) are used to define the relationships among resources, e.g. associating users to roles and specifying role hierarchy. For example, *belong*(*John*, *GM*) means user *John* is associated with the role *GM* (*General Manager*) and *manage*(*GM*, *PD*) specifies role *GM* has control over *PD* (*Purchasing Director*). According to Definition 3, the organizational model for the procurement workflow can be expressed as in Fig. 7.

By Definitions 1, 2, and 3, control flow, data flow, and organizational model are specified separately from one another. This assumption is based on our experiences with several commercial process modeling tools, such as the IBM WebSphere Business Modeler, TIBCO iProcess Suite, and Ultimus BPM Suite, where those three perspectives are modeled separately first and then integrated together via proprietary system functions, i.e., workflow constraints. This assumption is not a limitation of our approach. Instead, it aligns very well with our innovative treatment of multi-perspective workflow changes through a constraint-centric approach as discussed previously in Section 3. Thus, workflow constraints are formally defined as follows.

Table 2
Descriptions of procurement tasks.

Task	*Name*	*Description*
t_1	Submit purchase request	Employees submit purchase request.
t_2	Supervisor approval	The supervisor of task initiator approves the request
t_3	Check fund	When request amount is greater than $1000, a checking is conducted to make sure there is sufficient fund.
t_4	Mgmt approval	When fund is sufficient, general manager needs to approve.
t_5	Check inventory	Check whether the requested item is in stock.
t_6	Purchasing	Purchase order is issued to vendor to buy the requested item.
t_7	Receive item	Item is received from vendor and checked for quality.
t_8	Item delivery	The requested item is delivered.
t_9	Delivery notification	Delivery notification is sent to the item requester.
t_{10}	Archiving	The transaction information is archived.
t_{11}	Disapproval notification	The requestor is notified about the request denial.

Definition 4. Workflow Constraint

A workflow constraint $c \in C$ is a logic formula used to connect control flow, data flow, and organizational model or enforce business and operational rules. A workflow constraint is in the following form:

$$H \leftarrow X_1, \ldots, X_n, not\, Y_1, \ldots, not\, Y_m, n, m \geq 0$$

where H, X_1, …, X_n, Y_1, …, Y_m are logic formulas and *not* is a negation by failure [24,36]. H is the *head* of the constraint, and X_1, …, X_n, *not* Y_1, …, *not* Y_m consist of the constraint *body*. A constraint is activated if and only if X_1, …, X_n, *not* Y_1, …, *not* Y_m are all true, resulting in H being true. Note a workflow constraint may contain only *head* where *body* is empty. In that case, H is said to be true.

Several object and predicate symbols have been defined in Definitions 1, 2, and 3, including *T*, *s*, *e*, *RC*, *next*, *D*, *input*, *output*, *R*, *belong*, and *manage*. In order to represent workflow constraints for the procurement workflow as shown in Fig. 5, additional predicate symbols need to be defined as shown in Table 3. With this succinct set of object and predicate symbols, the problem of unifying control flow, data flow, organization model, and workflow constraints is resolved by means of a set of logic formulas.

It is worth noting that specifying a complete set of predicates to model all kinds of workflows is not the focus of this paper. Instead, our goal is to formally introduce the new constraint-centric workflow change management approach using a small set of predicates that are enough to model all perspectives in the sample procurement management process. More object and predicate symbols can always be added as needed to make the CWCA modeling framework capable for modeling any specific workflow. The choice of what symbols to include depends on the workflow modeling context as well as the interests and concerns of workflow designer and stakeholders. With the CWCA modeling framework defined in this section, three types of constraints can be modeled: 1) routing constraints that specify the routing conditions for branches; 2) authorization constraints that associate resources with task, and 3) data constraints that relate data items with tasks and routing constructs. The constraint type can be specified as constraint metadata as defined below.

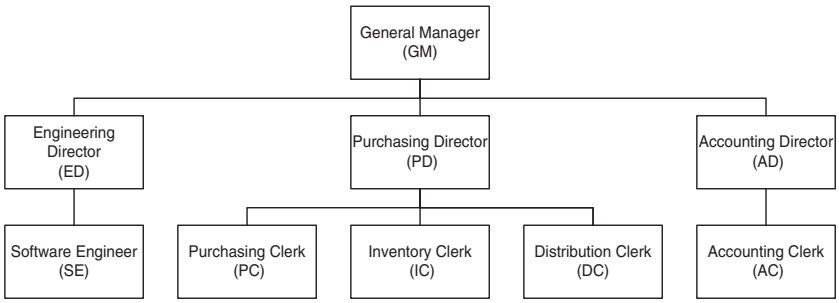

Fig. 4. Organization model for the procurement workflow.

Definition 5. Workflow Constraint Metadata

For each workflow constraint $c \in C$, its metadata is defined as a five tuple<*cid*, *TY*, *P*, *H*, *E*>, where

- *cid* is the unique identifier of the constraint,
- *TY* ∈ {routing, authorization, data} is the constraint type,
- *P* is the premise of the constraint
- *H* is the conclusion of the constraint, and
- $E = \{e \mid e \in T \cup RC \cup D \cup R \cup C$ is a workflow element involved in the constraint}.

Thus, the constraints for the procurement workflow in Fig. 5 can be formalized as shown in Fig. 8. In particular, constraints c_1 to c_{10} are authorization constraints. Constraint c_{11} to c_{20} are routing constraints. Constraints c_{21} to c_{27} are data constraints. To the best of our knowledge, the constraint-centric workflow modeling framework is the first attempt with using First Order Logic to unify the four workflow perspectives for managing workflow changes.

5. Workflow constraint anomalies

The unified logic representation of a workflow model enables us to specify formally workflow changes in multiple workflow perspectives. In this section, we define workflow constraint anomalies and present the associated anomaly detection algorithm. We use the procurement workflow case to validate the algorithm.

5.1. Types of workflow constraint anomalies

Workflow changes are formally defined in terms of a workflow change function.

Definition 6. Workflow Change Function

Let *W* be a workflow model $W = T \cup RC \cup D \cup R \cup C$, a workflow change function is defined as $f_c : W \rightarrow W'$ where $W' = T' \cup RC' \cup D' \cup R' \cup C'$ is the result of performing operations *OP* {*Insertion*, *Deletion*, *Modification*} on *W*.

Given the workflow models before and after changes, i.e. *W* and *W*′, workflow constraint anomalies can be classified into four categories, namely, *Missing Constraint Anomaly*, *Defective Constraint Anomaly*, *Redundant Constraint Anomaly*, and *Conflicting Constraint Anomaly*, as discussed next.

5.1.1. Missing constraint anomaly

Missing constraint anomaly occurs when a workflow cannot be properly executed due to a lack of certain constraints. Each of the following scenarios can cause missing constraint anomalies.

c_1: t_2 (Supervisor Approval) must be executed by the supervisor of the purchase requestor.
c_2: t_3 (Check Fund) must be executed by an Accounting Clerk (AC).
c_3: t_4 (Mgmt. Approval) must be executed by a General Manager (GM).
c_4: t_5 (Inventory Checking) is executed by an Inventory Clerk (IC).
c_5: t_6 (Purchasing) is handled by a Purchasing Clerk (PC)
c_6: t_7 (Receive Item) must be handled by a Distribution Clerk (DC).
c_7: t_8 (Item Delivery) must be handled by a Distribution Clerk (DC).
c_8: t_{10} (Archiving) is handled by a Purchasing Clerk (PC).
c_9: t_1 (Submit Purchase Request) and t_3 (Check Fund) cannot be done by the same person.
c_{10}: t_7 (Receive Item) and t_8 (Item Delivery) must be handled by the same person.
c_{11}: if supervisor approves the request, Branch construct b_2 will be activated.
c_{12}: if supervisor does not approve the request, t_{12} (Disapproval Notification) is activated.
c_{13}: if item total value is greater than or equal to $5000, t_3 (Check Fund) is executed.
c_{14}: if item total value is less than one thousand, t_5 (Check Inventory) is executed.
c_{15}: if there is enough fund, then t_4 (Mgmt Approval) is executed.
c_{16}: if there is not enough fund, t_{12} (Disapproval Notification) is activated.
c_{17}: if manager approves the request, t_5 (Check Inventory) is executed.
c_{18}: if manager does not approve the request, t_{12} (Disapproval Notification) is activated.
c_{19}: if requested items are in stock, t_8 (Item Delivery) is executed.
c_{20}: if requested items are not in stock, t_6 (Purchasing) is executed.
c_{21}: d_1 (supervisor approval) is the input data for Branch b_1.
c_{22}: d_2 (item total value) is the input data for Branch b_2.
c_{23}: d_2 (item total value) is the input data for Branch b_3.
c_{24}: d_3 (available fund) is the input data for Branch b_3.
c_{25}: d_4 (mgmt approval) is the input data for Branch b_4.
c_{26}: d_5 (requested item quantity) is the input data for Branch b_5.
c_{27}: d_6 (requested item inventory) is the input data for Branch b_5.

Fig. 5. Key constraints for the procurement workflow.

start, end, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, b_1, b_2, b_3, b_4, b_5, f_1, j_1, m_1, next(start, t_1), next(t_1, t_2), next(t_2, b_1), next(b_1, b_2), next(b_1, t_{11}), next(b_2, t_3), next(b_2, t_5), next(t_3, b_3), next (b_3, t_{11}), next(b_3, t_4), next(t_4, b_4), next(b_4, t_{11}), next(b_4, t_5), next(t_5, b_5), next(b_5, t_6), next(b_5, t_8), next(t_6, t_7), next(t_7, t_8), next(t_8, f_1), next(f_1, t_{10}), next(f_1, t_9), next(t_{10}, j_1), next(t_9, j_1), next(j_1, m_1), next(t_{11}, m_1), next(m_1,end)

Fig. 6. Logic representation of the control flow.

manage(GM,ED), manage(GM,PD), manage(GM,AD),manage(ED,SE), manage(PD,PC), manage(PD,IC),manage(PD,DC), manage(AD,AC), belong(John, GM),belong (Joe, ED), belong(Jason, PD), belong(Maggie, AD),belong (Eric, SE), belong(Ray, SE), belong(Peter, PC), belong(Sam, IC), belong(Jack, DC), belong(Dan, DC), belong(Steve, AC), belong(Ben, AC)

Fig. 7. Logic representation of the organizational model.

Scenario 1 (Task without Resources). A task is given in the workflow model, but no resources are specified to execute the task. As a result, the task cannot be executed during runtime, resulting in halting of workflow execution at the task.

Scenario 2 (Branch without Routing Constraints). A branch routing construct is inserted in the workflow model, but no routing constraints are specified. This will cause a workflow execution fault.

Every branch routing construct represents a decision point since the control flow takes a particular path according to a given condition. Therefore, at least two outgoing links must be specified at each decision point, and thus at least two routing constraints must be in place to specify the routing conditions.

Proposition 1. Condition for Missing Constraint Anomaly

Given a task $t \in T'$ that t $\notin \cup_{i=1}^{n} c_i.E$ where $c_i \in C'$ and $c_i.TY =$ {authorization}, or a branch routing construct $b \in RC'$ that b $\notin \cup_{j=1}^{m} c_j.E$ where $c_j \in C'$ and $c_j.TY =$ {routing}, a workflow model after the change W' contains at least one missing constraint anomaly.

Discussion. $\cup_{i=1}^{n} c_i.E$ is the set of process elements referenced in all authorization constraints after the change. Given a task $t \in T'$, t is part of the workflow model after the change. Given t $\notin \cup_{i=1}^{n} c_i.E$, no authorization constraints are defined for t, leading to a missing constraint anomaly. Similarly, $\cup_{j=1}^{m} c_j.E$ is the set of process elements referenced in all routing constraints after the change. Given a branch $b \in RC'$, b is part of the workflow model after the change. Given $b \notin \cup_{j=1}^{m} c_j.E$, b is not referenced in any routing constraints, resulting in a missing constraint anomaly.

5.1.2. Invalid constraint anomaly

Constraints are identified as invalid if they refer to non-existing workflow elements. Invalid constraints can be further divided into two subtypes, namely, *defective constraint* and *redundant constraint*, based on whether the constraint is required for workflow execution. More specifically, each of the following scenarios can cause defective constraint anomalies.

Scenario 3 (Non-existing Resource). One or more resources in authorization constraints are not available. For example, constraint c_3: *execute*(*DC*, t_7) is defective when the role Distribution Clerk (DC) becomes unavailable due to organizational restructuring. Constraint c_3 cannot be enforced properly for t_7 during workflow execution, which will cause an execution halt.

Scenario 4 (Non-existing Target Task). The target tasks in a routing constraint become unavailable, resulting in defective routing constraints. For example, constraint c_{13}: $next(b_2, t_3) \leftarrow d_2 \geq 5000$ can cause a routing exception if t_3 is removed.

Scenario 5 (Non-existing Routing Data Item). The data item in a routing constraint become unavailable, resulting in defective routing constraints. For example, constraint c_{14}: $next(b_2, t_5) \leftarrow d_2 < 5000$ can cause a routing exception if d_2 becomes unavailable.

Proposition 2. Condition for Defective Constraint Anomaly

Given a constraint $c \in C'$ that $c.TY =$ {authorization}, $c.E \cap T \backslash T' = \emptyset$, $c.E \cap R \backslash R' \neq \emptyset$, or $c.TY =$ {routing}, $c.E \cap RC \backslash RC' = \emptyset$, $c.E \cap (T \backslash T' \cup D \backslash D') \neq \emptyset$, the workflow model after changes W' contains at least one defective constraint anomaly.

Discussion. $c.TY =$ {authorization} means c is an authorization constraint. $T \backslash T'$ is the set of removed tasks and $c.E \cap T \backslash T' = \emptyset$ specifies that c does not refer to non-existing tasks. $R \backslash R'$ is the set of removed resources and $c.E \cap R \backslash R' \neq \emptyset$ specifies that c references to some non-existing resources, leading to a defective constraint anomaly. $c.TY =$ {routing} means c is a routing constraint. $RC \backslash RC'$ is the set of removed routing constructs and $D \backslash D'$ is the set of removed data items. $c.E \cap RC \backslash RC' = \emptyset$ specifies that c does not refer to non-existing routing construct, and $c.E \cap (T \backslash T' \cup D \backslash D') \neq \emptyset$ specifies that contains either removed tasks or removed data items, resulting in a defective constraint anomaly.

It is noteworthy that although some defective constraints may be due to some non-existing workflow elements, those constraints are still required for workflow execution, and the errors should be fixed. On the other hand, if some invalid constraints are no longer needed for workflow execution, they are referred to as redundant constraints, which can be simply removed. Each of the following scenarios can lead to redundant constraints.

Scenario 6 (Redundant Authorization Constraint). Authorization constraints become redundant when the related task does not exist. For example, constraint c_2: *execute*(*AC*, t_3) becomes redundant when task t_3 is removed from the workflow although role Accounting Clerk (*AC*) may still exist.

Scenario 7 (Redundant Routing Constraint). Routing constraints become redundant when the related routing construct does not exist. For example, constraint c_{13}: $next(b_2, t_3) \leftarrow d_2 \geq 5000$ becomes redundant when routing construct b_2 is removed even when t_3 and d_2 both exist.

Scenario 8 (Redundant Data Constraint). Data constraints become redundant when a related data item, task or routing construct does not exist. For example, constraint c_{31}: *input* (b_1, d_1) becomes redundant when data d_1 is removed or branch b_1 is deleted, because there is no need to specify the input in cases.

Table 3
Summary of additional predicate symbols.

Symbols	Meaning
execute(x, y)	$x \in R$, $y \in T$, x is assigned to execute task y
cannot_execute(x, y)	$x \in R$, $y \in T$, x cannot execute task y
$=, <, \leq, >, \geq$	Relational operators: equal to, less than, less than or equal to, greater than, greater than or equal to.
$+, -, *, /$	Arithmetic operators: addition, subtraction, multiplication, division

c_1: $execute(x, t_2) \leftarrow manage(x, y), execute(y, t_1)$	c_{15}: $next(b_3, t_4) \leftarrow (d_3 - d_2) \geq 0$
c_2: $execute(AC, t_3)$	c_{16}: $next(b_3, t_{11}) \leftarrow (d_3 - d_2) < 0$
c_3: $execute(GM, t_4)$	c_{17}: $next(b_4, t_5) \leftarrow d_4$
c_4: $execute(IC, t_5)$	c_{18}: $next(b_4, t_{11}) \leftarrow not\ d_4$
c_5: $execute(PC, t_6)$	c_{19}: $next(b_5, t_8) \leftarrow (d_6 - d_5) \geq 0$
c_6: $execute(DC, t_7)$	c_{20}: $next(b_5, t_6) \leftarrow (d_6 - d_5) < 0$
c_7: $execute(DC, t_8)$	c_{21}: $input(b_1, d_1)$
c_8: $execute(PC, t_{10})$	c_{22}: $input(b_2, d_2)$
c_9: $cannot\_execute(u, t_3) \leftarrow execute(u, t_1)$	c_{23}: $input(b_3, d_2)$
c_{10}: $execute(u, t_8) \leftarrow execute(u, t_7)$	c_{24}: $input(b_3, d_3)$
c_{11}: $next(b_1, b_2) \leftarrow d_1$	c_{25}: $input(b_4, d_4)$
c_{12}: $next(b_1, t_{11}) \leftarrow not\ d_1$	c_{26}: $input(b_5, d_5)$
c_{13}: $next(b_2, t_3) \leftarrow d_2 \geq 5000$	c_{27}: $input(b_5, d_6)$
c_{14}: $next(b_2, t_5) \leftarrow d_2 < 5000$	

Fig. 8. Logic representation of workflow constraints.

Proposition 3. Condition for Redundant Constraint Anomaly

Given a constraint $c \in C'$ that $c.TY = \{\text{authorization}\}$, $c.E \cap T \backslash T' \neq \emptyset$, or $c.TY = \{\text{routing}\}$, $c.E \cap RC|RC' \neq \emptyset$, or $c.TY \in \{\text{data}\}$, $c.E \cap (T \backslash T' \cup RC \backslash RC' \cup D \backslash D' \cup R \backslash R') \neq \emptyset$, the workflow model after changes W' contains at least one redundant constraint anomaly.

Discussion. Given $c.TY = \{\text{authorization}\}$, c is an authorization constraint and $c.E \cap T \backslash T' \neq \emptyset$ means c refers to non-existing tasks. An authorization constraint is redundant if the associated tasks have been removed. Similarly, given $c.TY = \{\text{routing}\}$, $c.E \cap RC|RC' \neq \emptyset$, the routing constraint becomes redundant when the associated routing construct no longer exists. For data and association constraints, i.e. $c.TY \in \{\text{data}\}$, if any of the involved process elements does not exist, i.e. $c.E \cap (T \backslash T' \cup RC \backslash RC' \cup D \backslash D' \cup R \backslash R') \neq \emptyset$, the constraint is no longer needed for workflow execution as discussed in Scenario 8.

5.1.3. Conflicting constraint anomaly

A conflicting constraint anomaly occurs when there are constraints causing a contradiction. Each of the following scenarios can cause conflicting constraint anomalies.

Scenario 9 (Conflicting Routing Constraint). A conflicting constraint occurs when two routing constraints create a routing contradiction. For example, if the following two constraints exist at a branch b: $next(b, t_1) \leftarrow d$ and $next(b, t_2) \leftarrow d$, then when d is true, the two target tasks from b lead to a conflict.

Scenario 10 (Conflicting Authorization Constraint). A conflicting constraint occurs when two authorization constraints are against each other. For example, if both constraints $execute(r, t)$, and $cannot\_execute(r, t)$ exist, a resource assignment confliction occurs for task t. Another example could be a binding of duties constraint as c_{10}: $execute(u, t_8) \leftarrow execute(u, t_7)$ can cause a conflict when the resources assigned to tasks t_7 and t_8 do not have any common users.

Proposition 4. Conditions for Conflicting Constraint Anomaly

Given three constraints $c_i, c_j, c_k \in C'$ if any of the following three conditions holds, the workflow model W' contains at least one conflicting constraint anomaly:

1) $c_i.TY = c_j.TY = \{\text{routing}\}$, $c_i.P = c_j.P$, $c_i.H = next(x, y)$, $c_j.H = next(x, z)$, where $x, y, z \in \{start, end\} \cup T \cup RC$, $x \notin \{f, end\}$, $y, z \notin \{start\}$, $y \neq z$
2) $c_i.TY = c_j.TY = \{\text{authorization}\}$, $c_i.P = c_j.P$, $c_i.E = c_j.E$, $c_i.H \neq c_j.H$
3) $c_i.TY = c_j.TY = c_k.TY = \{\text{authorization}\}$, $c_i.H = execute(\alpha, t_m)$, $c_j.H = execute(\beta, t_n)$, $c_k.H = execute(\gamma, t_n)$, $c_k.P = execute(\gamma, t_m)$, where $t_m, t_n \in T$, $\alpha, \beta \in RO$, $\gamma \in U$, $\nexists\ \gamma$, so that $belong(\gamma, \alpha)$ and $belong(\gamma, \beta)$ are both true.

Discussion. Given two constraints $c_i, c_j \in C'$ that $c_i.TY = c_j.TY = \{\text{routing}\}$, $c_i.P = c_j.P$ means the two routing constraints have the same premises, but $c_j.H = next(x, z)$, where $x, y, z \in \{start, end\} \cup T \cup RC$, $x \notin \{f, end\}$, $y, z \notin \{start\}$, $y \neq z$ specifies that there exists two control flow links starting from the same process node x leading to two different process nodes y and z. When x is not a *fork* routing construct, a conflicting constraint anomaly occurs. Given two constraints $c_i, c_j \in C'$ that $c_i.TY = c_j.TY = \{\text{authorization}\}$, $c_i.P = c_j.P$, $c_i.E = c_j.E$ indicate the two authorization constraints are defined on the same resources and tasks and the premises of the constraints are the same, but $c_i.H \neq c_j.H$ specifies that the predicates in the conclusions are different, i.e. both *execute* and *cannot_execute* exist, resulting in a conflicting authorization. $c_i.TY = c_j.TY = c_k.TY = \{\text{authorization}\}$, $c_i.H = execute(\alpha, t_m)$, $c_j.H = execute(\beta, t_n)$, $c_k.H = execute(\gamma, t_n)$, $c_k.P = execute(\gamma, t_m)$, $\alpha, \beta \in RO$, $\gamma \in U$ indicates that c_k is a binding of duties constraint that requires user γ to execute both tasks t_m and t_n, and c_i and c_j are two resource assignment constraints that assign α and β as the roles to execute tasks t_m and t_n respectively. Then, $\nexists\ \gamma$, so that $belong(\gamma, \alpha)$ and $belong(\gamma, \beta)$ are both true states that there are no users that belong to both α and β roles. In this case, the "binding of duties" constraint cannot be satisfied because proper task assignments are not possible with the two given resource assignment constraints due to a lack of common users assigned to the two relevant roles.

A Venn diagram representing different categories of workflow constraints with respect to constraint anomalies is developed as shown in Fig. 9. This Venn diagram is useful for understanding the relationships among workflow constraint anomalies and the basic principles of constraint management after a workflow change. In addition, we can show that the classification of workflow constraint is complete. Let C be the set of workflow constraints and *MI, IN, VA, RE, DE, CO* be the sets of missing, invalid, valid, redundant, defective, and conflicting constraints respectively. Given any workflow constraint $c \in C$, it can always be placed into the space of workflow constraints represented by the Venn diagram in Fig. 9. This can be shown algorithmically. For each constraint, it must be either existing or missing, i.e., $\forall\ c \in C$, $c \in MI \cup \neg MI$, without exception. For an existing constraint, it must be either valid or invalid based whether the constraint references non-existing workflow elements, i.e., $\forall\ c \in \neg MI$, $c \in IN \cup VA$. Note that these three constraint types are mutually exclusive, i.e. $MI \cap IN \cap VA = \emptyset$. Further, for each invalid constraint $c' \in IN$, if it is not required for workflow execution, it is a redundant constraint $c' \in RE$; otherwise it is a defective constraint $c' \in DE$. Lastly, for each valid constraint $c'' \in VA$, it either causes conflict, i.e. $c'' \in CO$, or does not lead to contradiction, i.e., $c'' \in \neg CO$. Thus, the space of workflow constraints as depicted in Fig. 9 is complete.

Propositions 1 through 4 define all possible conditions where workflow constraint anomalies can occur within our CWCA framework, they provide the foundation for designing algorithms to detect workflow constraint anomalies as we will discuss in the next section. It is worth noting that workflow constraints are often not explicitly modeled in workflow systems. Workflow constraints can also be informally defined as business rules related to workflow, which are often specified in policy and procedure documents. In this paper, we assume that workflow constraints have been properly identified and

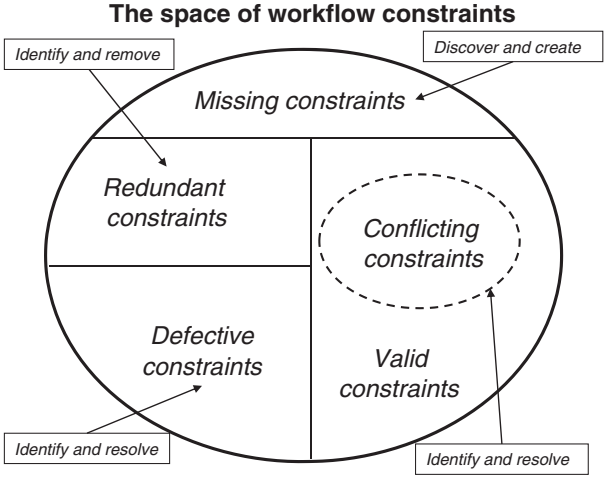

Fig. 9. Venn diagram of workflow constraint types.

PROCEDURE ConstraintAnomalyDetection
for each $t \in T'$ {*count*(*execute*(*x, t*), *m*)
  if ($m = 0$) print "missing constraint for t"} *// detect scenario 1*
for each $b \in RC'$ {*count*(*constraint*(c,*b*),*n*)
  if ($n < 2$) print "missing constraint for b"} *// detect scenario 2*
for each $c \in C'$ {
  if c.E $\in T' \cup RC' \cup D' \cup R' \cup C'$ {
    add c to set *VA // add c to the valid constraint set VA*
  else *// c is identified as an invalid constraint*
    if (c.TY = {data}) {print "redundant constraint c"} *// detect scenarios 8*
    if (c.TY = {authorization}) {
     if (c.E ∩ T\T ' ≠ ∅) print "redundant authorization constraint c" *// detect scenario 6*
    else if ($c.E \cap R \backslash R' \neq \emptyset$) print "defective constraint c due to missing resource" }*// detect scenario 3*
    if (c.TY = {routing}) {
      if ($c.E \cap RC \backslash RC' \neq \emptyset$) print "redundant routing constraint c" *// detect scenario 7*
      else if ($c.E \cap (T \backslash T' \cup D \backslash D') \neq \emptyset$) print "defective constraint c" }}*// detect scenarios 4 and 5*
for each pair $c_i, c_j \in VA$ {
    if (c_i.TY= c_j.TY={routing}, $c_i.P = c_j.P$, $c_i.H = next(x, y)$, $c_j.H = next(x, z)$, where $x, y, z \in \{start, end\} \cup T \cup RC$, x $\notin \{f, end\}$, $y, z \notin \{start\}$, $y \neq z$) {
     print "conflicting routing constraints c_i and c_j"} *// detect scenario 9*
    if ($c_i.TY = c_j.TY$ ={authorization}, $c_i.P = c_j.P$, c_i.E = c_j.E, $c_i.H \neq c_j.H$ {
     print "conflicting authorization constraint for c_i and c_j" } *//detect scenario 10*
    if ($c_i.TY = c_j.TY$ = {authorization}, $c_i.H = execute(\alpha, t_m)$, $c_j.H = execute(\beta, t_n)$, $\alpha, \beta \in RO$),
        there exists $c_k \in VA$ $c_k.H = execute(\gamma, t_n)$, $c_k.P = execute(\gamma, t_m)$, $\gamma \in U$,and $\nexists\ \gamma$, *belong*(γ, α) and *belong*(γ, β) are both true) {
     print "conflicting binding of duties constraint for c_i, c_j, and c_k"} *//detect scenario 10*

Fig. 10. Constraint anomaly detection algorithm.

specified using process mapping approaches [35,47,56]. The CWCA approach consists of a set of extensible anomaly detection rules to identify workflow anomalies during system changes and thus provides a flexible and extensible framework to accommodate new business environments. For example, our current approach can detect anomalies due to a missing resource, which requires that the system specifies all resources needed for executing existing tasks. However, in a new business environment, new detection rules may be needed to handle new workflow resources that have not been defined. The ability to extend the modeling framework with new predicates and rules is a strength of the CWCA approach.

5.2. Workflow constraint anomaly detection

Based on Propositions 1 through 4, we develop an algorithm to detect workflow constraint anomalies as shown in Fig. 10. One aggregation function supported by most existing logic programming languages and systems [20,24], namely, *count*(Q, n) is leveraged in the algorithm, where *count*(Q, n) returns the number of different answers of query ← Q to n. This algorithm also illustrates the precedence of the four propositions, i.e., the sequence of applying the propositions to detect workflow constraint anomalies. More specifically, we first check each task and routing construct to detect missing constraints. Then, we divide all existing constraints into valid and invalid constraints based on whether they refer to non-existing workflow elements. For invalid constraints, according to different types of constraints, we first detect redundant constraints and then defective constraints, because the conditions for redundant constraints are easier to check. After that, we detect conflicting constraints among all valid constraints. According to the four propositions, the algorithm shown in Fig. 10 is able to detect all constraint anomalies in our CWCA framework. Once the anomalies are detected, proper actions must be taken according to the anomaly types to ensure workflow consistency. In particular, additional constraints should be added to address missing constraint anomalies. Redundant constraints should be removed. Defective and conflicting constraints should be resolved to achieve constraint consistency. Note that the algorithm should be executed every time there is a change in the workflow model to update the anomaly list. In this paper, we focus on detecting workflow change anomalies, and leave detailed discussions on anomaly correction to future research.

In order to validate the algorithm, we make some changes to the procurement workflow and apply the algorithm to detect potential anomalies. The changes are listed in Table 4 and the new workflow is shown in Fig. 11. In particular, a new task t_{12} (Vendor Selection) is added to incorporate a formal procedure for selecting the best vendor for the requested items. Tasks t_3 (Check Fund) is removed to improve the efficiency of the workflow by reassigning the "check fund" task to task t_2 (Supervisor Approval). t_{10} (Archiving) is removed because the integration of a new automatic archiving system with the procurement process. Some related routing constructs are also deleted, including b_3, f_1, and j_1. Due to a new organizational restriction, role 'Inventory Clerk' is merged into role 'Purchasing Clerk'. A data item is renamed d_2 (item total value), which corresponds to the deletion of d_2 and the insertion of d_7 (request item total). The new control flow is presented in Fig. 12, and the new organizational model is represented in Fig. 13.

By applying the algorithm to the modified workflow model, we can identify five missing constraint anomalies, seven redundant constraints, and six invalid constraints. Missing constraints are identified for tasks t_1 (Purchase Request), t_2 (Supervisor Approval), t_9 (Delivery Notification), t_{11} (Disapproval Notification), and t_{12} (Vender Selection). In particular, t_1 can be executed by any employee in the organization, and no resource in the existing organizational model

Table 4
Changes to the procurement process.

	Insertion	Deletion	Modification
Control flow	t_{12} (Vendor Selection)	t_3 (check fund) t_{10} (archiving) b_3, f_1, j_1	t_2 (Supervisor approval) is revised to check the fund
Organizational model	None	role 'inventory clerk'	Assign all inventory clerks as purchasing clerks
Data Flow	d_7 (request item total)	d_2 (item total value)	none

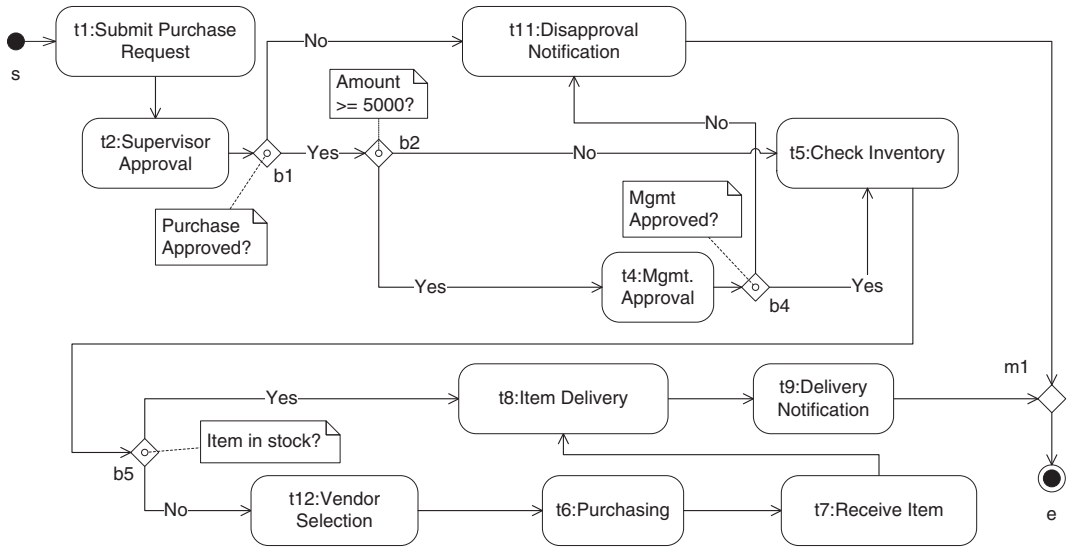

Fig. 11. Procurement process after changes.

shown in Fig. 4 can be assigned to t_1. To address this, we add one *special role* "*Any Employee*" *and assign it to* t_1 *by adding* a constraint c_{28}: *execute*("*Any Employee*", t_1). The role executing tasks t_2 is resolved based on role hierarchy. We need to add another special role "Supervisor" to execute t_2, in order to achieve constraint consistency. Therefore, one more constraint is added c_{29}: *execute*("Supervisor", t_2). For tasks t_9 and t_{10}, the email system should be the associated resource, which is not part of the organizational model in Fig. 13 either. To solve this problem, we add an machine agent role "Email System" and assign it to t_9 and t_{10} by adding two more constraints, i.e., c_{40}: *execute*("*Email System*", t_9) and c_{41}: *execute*("*Email System*", t_{10}). t_{12} (Vender Selection) is a newly inserted task and a resource such as the purchasing manager should be assigned to execute the task.

Due to the removal of task t_3, Constraints c_2 and c_9 become redundant, and Constraint c_{13} is now invalid. Because all resource assignment constraints associated with t_3 are no longer necessary, the separation of duty constraint defined on t_1 and t_3 is always true, and the routing constraint for b_2 cannot route the flow to t_3 anymore. Similarly, deletion of task t_{10} makes its resource assignment constraint c_{10} redundant. In the same way, removing b_3 results in redundant constraints c_{23}, and c_{24}, and invalid constraints c_{15} and c_{16}. Renaming data item d_2 makes the related constraints c_{13}, c_{14}, c_{15}, and c_{16} invalid. In this section, we used an example to illustrate our algorithm for detecting workflow constraint anomalies. Next, we further validate our approach by means of a proof-of-concept prototype system based on a rule engine and a workflow management system.

5.3. A proof-of-concept system

Although the anomaly detection algorithm shown in Fig. 10 can be implemented by either procedural languages or declarative languages, we choose a declarative rule-based approach to develop the CWCA prototype system. This is done for three reasons. First, CWCA is based on First Order Logic, which is a natural fit for a rule-based approach. Second, the anomaly detection algorithm is essentially a set of detection rules that can scale up in quantity and may change frequently due to new business requirements. The declarative rule-based approach can handle large number of rules with frequent changes better than a procedural approach [24]. Third, there is a recent trend in industry on integrating rule management systems with other enterprise systems, e.g., workflow management systems, to better align business with IT [37,39], so that a declarative CWCA implementation can maximally leverage existing organizational assets on rule engines and workflow management systems.

Fig. 14 shows the architecture of the CWCA prototype system, which contains two sub-systems, i.e. a workflow management system and a rule management system. In particular, we choose jBPM (http://www.jboss.org/jbossjbpm/) as the workflow management system and Drools (http://www.jboss.org/drools/) as the rule management system, because they are both open-source, contain user-friendly development tools based on Eclipse, and provide web-based interfaces for workflow execution and rule management. Further, both systems are being actively developed and supported by the JBoss community and can be neatly integrated into the JBoss Application Server.

The rules for detecting workflow constraint anomalies are defined using the Drools rule language, which is based on First Order Logic. Fig. 15 shows some anomaly detection rules in the Drools rule workbench in Eclipse. The workflow model is defined using the graphical process designer of jBPM. All basic workflow modeling constructs we defined in Section 4 are supported by the process designer. The process diagram can be exported into an XML file, which is transformed into rules to create the knowledge base for the detection rules to reason about. After the workflow model and detection rules are loaded into the rule base, the rule engine performs rule reference and presents the results to the rule design GUI as shown in Fig. 16.

The proof-of-concept prototype system helps demonstrate the feasibility of our CWCA approach and further validates the anomaly detection algorithm via a rule engine. Given that jBPM and Drools can be used as embedded systems and accessed via APIs, the CWCA framework can be integrated into existing applications to provide workflow capability that supports continuous process improvement.

start, end, $t_1, t_2, t_4, t_5, t_6, t_7, t_8, t_9, t_{11}, t_{12}, b_1, b_2, b_4, b_5, m_1$, *next*(*start*, t_1), *next*(t_1, t_2), *next*(t_2, b_1), *next*(b_1, b_2), *next*(b_1, t_{11}), *next*(b_2, t_4), *next*(b_2, t_5), *next*(t_4, b_4), *next*(b_4, t_{11}), *next*(b_4, t_5), *next*(t_5, b_5), *next*(b_5, t_{11}), *next*(b_5, t_8), *next*(t_{12}, t_6), *next*(t_6, t_7), *next*(t_7, t_8), *next*(t_8, t_9), *next*(t_9, m_1), *next*(t_{11}, m_1), *next*(m_1,*end*)

Fig. 12. Logic representation of revised control flow.

manage(*GM,ED*), *manage*(*GM,PD*), *manage*(*GM,AD*), *manage*(*ED,SE*), *manage*(*PD,PC*), *manage*(*PD,DC*), *manage*(*AD,AC*), *belong*(*John, GM*), *belong* (*Joe, ED*), *belong*(*Jason, PD*), *belong*(*Maggie, AD*), *belong* (*Eric, SE*), *belong*(*Ray, SE*), *belong*(*Peter, PC*), *belong*(*Sam, PC*), *belong*(*Jack, DC*), *belong*(*Dan, DC*), *belong*(*Steve, AC*), *belong*(*Ben, AC*)

Fig. 13. Logic representation of revised org model.

6. Discussion

In this section, we compare our Constraint-centric Workflow Change Analytics (CWCA) framework with some related approaches and discuss limitations of our research. Table 5 shows the comparison results between CWCA and other approaches, namely Petri nets [52,53], Metagraphs [5,6], Communicating Sequential Process (CSP) [58,59], where NA means information not found in the related references.

The first issue is the expressive power of the languages used in different approaches in terms of modeling the four workflow perspectives as defined in Section 4, including control flow, data flow, organizational model, and workflow constraints. Petri nets with its extensions including colored, timed, and hierarchical Petri nets and Metagraphs have been applied to represent control flow, data flow, and organizational models. Workflow constraints are essentially declarative rules, which cannot be easily represented by graphical formalisms such as Petri nets and Metagraphs. CSP has been used to provide formal semantics to BPMN [59], thus representing control flow and data flow. Workflow constraints are essentially declarative rules, which have been represented using First Order Logic in many research works. Although there has been research on translations between logic programs and Petri net models [32,48], research on leveraging Petri nets, Metagraphs, or CSP to model workflow constraints has been scant. In comparison with other three approaches, CWCA enables a unified representation of all four workflow perspectives by attaching formal workflow semantics to First Order Logic. However, the expressive power of CWCA framework defined in this paper in terms of modeling different workflow patterns are limited compared with Petri nets and CSP [54,59]. Therefore, extending CWCA's expressiveness is an important future research topic.

The second issue is supporting multi-perspective changes. One of the key innovations of CWCA is the ability to analyze change consequences among all four workflow perspectives within one unified constraint-centric framework. Petri nets have been applied to study change interactions within control flow and between control flow and data flow. In this paper, we focus on defining multi-perspective workflow changes and analyzing basic change anomalies, thus leaving change verification within a single perspective, such as control flow verification, to future research. Given CWCA is grounded on First Order Logic, it can be extended to control flow verification using the methods found in [10,21]. Furthermore, we are planning to analyze more complex change interactions among different perspectives using CWCA.

The third issue is the capability of handling dynamic workflow change. Dynamic workflow changes refer to the problem of managing change consequences to running instances of a workflow [12]. Petri nets have been extensively used in handling dynamic workflow changes [53]. Metagraphs and CSP have not been used to study workflow changes. In this paper, CWCA only deals with static change consequences during the workflow design time. However, CWCA could be extended to support dynamic workflow changes. More specifically, additional predicates will need to be defined to represent instances of workflow components and additional rules need to be specified for handling dynamic change anomalies. For instance, once a task is activated and a resource is assigned to the task instance, we could use *execute_instance* (*instance_id*, *task_name*, *resource_id*) to represent this resource assignment and create an additional rule: *missing_resource* (*resource_id*) ← *execute_instance* (*instance_id*, *task_name*, *resource_id*), *not resource(resource_id, resource_name)* to show a temporal missing resource anomaly whenever a resource become occupied when the corresponding task is still active.

The fourth issue is about the verification mechanism and tool supports. All four approaches shown in Table 5 are grounded in rigorous verification mechanisms. However, supporting tools for the Metagraphs approach has not been found. Although CPN Tools for Petri nets approach and FDR Model Checker for CSP approach are powerful tools widely used in research projects and industry practice, their supports in existing workflow management systems are very limited. The CWCA approach can be implemented using any rule engines as illustrated in Section 5.3. Various industry-strength rule engines are either freely available, such as Drools, XSB, CLIPS, and JESS, or have been offered by many leading software vendors, such as IBM ILOG, Oracle Business Rules, and Microsoft Windows Workflow Foundation Rules Engine. Thus, organizations can leverage more easily their existing IT assets to implement CWCA. We are refining and enhancing the prototype system and plan to use it in real organizations to further validate CWCA approach. Based on our experience with several workflow modeling and management systems, such as

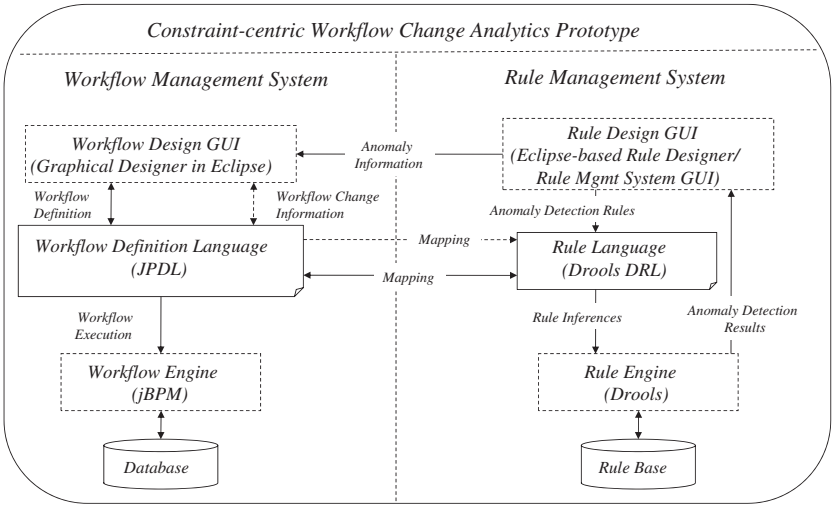

Fig. 14. Architecture of the prototype system.

PPSL.drl

```
rule "Missing Constraint for Decision Node"
ruleflow-group "Verification"
    when
        $n: Node (type == "decision")
        $next: Next(firstNode == $n)
        not Guard(next == $next)
    then
         insertLogical (new Anomaly ("Missing Constraint", "decision " +$n.getName() ));
end

rule "Missing Resource for Task"
ruleflow-group "Verification"
    when
        $n: Node (type == "task")
        not Execute(node == $n)
    then
         insertLogical (new Anomaly ("Missing Resource", "task " +$n.getName()));
end
```

Fig. 15. Anomaly detection rules in drools rule workbench.

Ultimus BPM Suite 7, jBoss jBPM, and IBM Websphere Business Modeler, there is no systematic support for multi-perspective workflow change analytics in those systems. For instance, Ultimus cannot detect defective data constraint anomaly during workflow design time and can only find task-without-resource anomaly during simulation. For jBoss jBPM, workflow constraints need to be manually specified in workflow model specification file using jPDL language. Thus, jBPM does not provide any support for design-time workflow change analysis. IBM Websphere Business Modeler can detect some of the change anomalies via their proprietary approaches, such as missing routing constraint anomaly, non-existing resources anomaly, and missing routing data item anomaly. It prevents some anomalies, such as the missing resource anomaly, by assigning a default built-in resource to any new task. However, IBM does not provide a systematic discussion on how IBM Webshphere Business Modeler handles workflow changes.

Another important issue is about the performance of rule engines in terms of supporting large number of facts and rules. Many benchmarking and performance tests have been conducted to analyze and evaluate the performance and scalability of different rule engines [4,27,33]. The recent OpenRuleBench project (http://rulebench.projects.semwebcentral.org/) evaluates eleven rule engines with different technologies, such as XSB, Yap, Ontobroker, Jess, and Drools, for their performance and scalability for a number of problems using large data sets with up to 1,000,000 facts [33]. The results show that many rule engines have reasonable performance with large data sets. For example, the Wine ontology has 815 rules and 654 facts and many of its predicates are recursively dependent on each other through chains of rules. In the Wine ontology test, many rule engines, such as XSB and Ontobroker, can complete the evaluation test within few seconds. Drools, the rule engine we use to develop our prototype system, achieved the best score for the DBLP test, which contains 2,500,000 facts, but lagged behind other systems for other tests [33]. Given that our CWCA approach is based on First Order Logic, we could leverage any rule engine with the best performance to implement our approach. Drools is developed based on the well-known RETE algorithm, we can evaluate the performance of our approach more formally by leveraging research on the computational complexity of RETE [23]. Given the number of rules (P), number of facts (W), and average number of patterns/conditional elements (C), the algorithmic complexity of RETE in average case and worst case are $O(PW)$ and $O(PW^C)$ respectively. As shown in the Section 5, we defined ten scenarios for workflow constraint anomalies in total, so that P and C for our CWCA approach are relatively small numbers. W in CWCA depends on the number of tasks, data items, constraints, and their relationships, which varies for different workflow models. Based on our case study of several process reference models from vendors such as SAP, and Oracle [55], most workflows in enterprises have only dozens of tasks and constraints. Therefore, W in our proposed CWCA knowledge base should be much smaller than the data sets used in the OpenRuleBench tests aforementioned and our proposed system should be able to handle most workflow change analysis with good performance.

Problems | Javadoc | Declaration | Console

```
<terminated> PPSLExample [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe
****Process Changes*******
delete t3
delete t10
delete b3
delete IC
delete d2
add t12
add d7

Missing Resource: task t12
Invalid Constraint: c16 - because b3 is missing
Invalid Constraint: c15 - because b3 is missing
Redundant Constraint: c9 - because t10 is missing
Invalid Constraint: c13 - because t3 is missing
Redundant Constraint: c2 - because t3 is missing
```

Fig. 16. A screenshot of constraint detection results.

Besides the issues discussed above, there are other challenges for implementing CWCA. For example, workflow models are usually specified using different notations, such as BPMN, Petri nets, BPEL, and EPCs, and CWCA may be implemented using different rule engines with different rule languages. Translating workflow models in different notations into the logic formulas in different rule languages is challenging tedious process. Another challenge is the integration of a rule engine into the workflow system for seamless analysis of workflow changes. Some rule engines, such as Drools, can be easily embedded into Java application, whereas other rules systems, such as WebSphere ILOG Business Rule Management Systems, must be accessed via APIs. Given the heterogeneity of various implementation environments, seamlessly integration of CWCA with existing workflow management system requires additional research.

7. Conclusions

In order to support continuous process improvement systematically, there is a critical need to manage workflow changes more rigorously. In this paper, we presented a constraint-centric analytical approach for managing workflow changes, named Constraint-centric Workflow Change Analytics (CWCA). CWCA advocates a unified view of workflow changes based on workflow constraint analysis and enables a

Table 5
Comparison between CWCA and other approaches.

	CWCA	Petri nets	Metagraphs	CSP
Control flow	Yes	Yes	Yes	Yes
Data flow	Yes	Yes	Yes	Yes
Org model	Yes	Yes	Yes	No
Constraint	Yes	No	No	No
Static change	Yes	Yes	NA	NA
Multi-perspective change support	Yes	Some	Some	NA
Dynamic change	No	Yes	NA	NA
Verification Mechanism	Rule inference	Petri-nets-based Simulation	Matrix Operations	Model Checking
Tool support	Rule engines	CPN tools	NA	FDR model checker

comprehensive representation of multiple workflow perspectives for the purpose of detecting workflow change anomalies. More specifically, multi-perspective workflow change anomalies are formally defined and classified into four categories, namely, missing, redundant, defective, and conflicting anomalies. In order to validate the CWCA approach, we developed an anomaly detection algorithm and a proof-of-concept system using a rule-based approach. Our research fills a critical void in workflow change management and has significant impact on continuous process improvement and workflow management. The CWCA approach presented in this paper has some unique features and innovations compared with the existing body of work on workflow modeling and change management. At the same time, CWCA has great potential for future extensions and refinements to provide more utility for workflow change management.

Acknowledgement

This work was partially supported by a SRG grant from the City University of Hong Kong (No. 7002522).

References

[1] N.R. Adam, V. Atluri, W.K. Huang, Modeling and analysis of workflows using Petri nets, Journal of Intelligent Information Systems 10 (2) (1998) 131–158.
[2] T. Andrews, et al., Business process execution language for web services, version 1.1, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems, May 5 2003.
[3] V. Atluri, W.-k. Huang, V. Atluri, W.-k. Huang, An authorization model for workflows, Proceedings of Proceedings of the 4th European Symposium on Research in Computer Security: Computer Security, 1996, pp. 44–64.
[4] M. Bali, Drools JBoss rules 5.0 Developer's Guide, Packt Publishing, 2009.
[5] A. Basu, R.W. Blanning, A formal approach to workflow analysis, Information Systems Research 11 (1) (2000) 17–36.
[6] A. Basu, R.W. Blanning, Synthesis and decomposition of processes in organizations, Information Systems Research 14 (4) (2003) 337–355.
[7] A. Basu, A. Kumar, Research commentary: workflow management issues in e-Business, Information Systems Research 13 (1) (2002) 1–14.
[8] B. Berthomieu, M. Diaz, T. Cnrs, Modeling and verification of time dependent systems using timePetri nets, IEEE Transactions on Software Engineering 17 (3) (1991) 259–273.
[9] E. Bertino, E. Ferrari, V. Atluri, The specification and enforcement of authorization constraints in workflow management systems, ACM Transactions on Information and System Security 2 (1) (1999) 65–104.
[10] H.H. Bi, J.L. Zhao, Applying propositional logic to workflow verification, Information Technology and Management 5 (3–4) (2004) 293–318.
[11] M.B. Blake, H. Gomaa, Agent-oriented compositional approaches to services-based cross-organizational workflow, Decision Support Systems 40 (1) (2005) 31–50.
[12] F. Casati, S. Ceri, B. Pernici, G. Pozzi, Workflow evolution, Data & Knowledge Engineering 24 (3) (1998) 211–238.
[13] F. Casati, S. Ceri, S. Paraboschi, G. Pozzi, Specification and implementation of exceptions in workflow management systems, ACM Transactions on Database Systems 24 (3) (1999) 405–451.
[14] F. Casati, S. Castano, M. Fugini, I. Mirbel, B. Pernici, Using patterns to design rules in workflows, IEEE Transactions on Software Engineering 26 (8) (2000) 760–785.
[15] F. Casati, S. Castano, M. Fugini, Managing workflow authorization constraints through active database technology, Information Systems Frontiers 3 (3) (2001) 319–338.
[16] M. Chen, D. Zhang, L. Zhou, Empowering collaborative commerce with Web services enabled business process management systems, Decision Support Systems 43 (2) (2007) 530–546.
[17] D.K.W. Chiu, Q. Li, K. Karlapalem, A meta modeling approach to workflow management systems supporting exception handling, Information Systems 24 (2) (1999) 159–184.
[18] T. Curran, G. Keller, A. Ladd, SAP R/3 business blueprint : understanding the business process reference model, Prentice Hall PTR, Upper Saddle River, N.J., 1998.
[19] B. Curtis, M.I. Kellner, J. Over, Process modeling, Communications of the ACM 35 (9) (1992) 75–90.
[20] S.K. Das, Deductive databases and logic programming, Addison-Wesley, 1992.
[21] H. Davulcu, M. Kifer, C.R. Ramakrishnan, I.V. Ramakrishnan, Logic based modeling and analysis of workflows, Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 1998, pp. 25–33.
[22] C. Ellis, K. Keddara, ML-DEWS: modeling language to support dynamic evolution within workflow systems, Computer Supported Cooperative Work 9 (2000) 293–333.
[23] C.L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem, Artificial Intelligence 19 (1) (1982) 17–37.
[24] J.C. Giarratano, G. Riley, Expert systems: principles and programming, Course Technology (2005).
[25] M. Gruninger, C. Menzel, The process specification language (PSL) theory and applications, AI magazine 24 (3) (2003) 63–74.
[26] M. Gruninger, R. Hull, S.A. McIlraith, Short overview of flows: a first-order logic ontology for web services, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering (2008).
[27] Illation Ltd, Business Rule Engine Benchmarks, http://illation.com.au/benchmarks, 2007.
[28] K. Jensen, Coloured Petri nets: basic concepts, analysis methods, and practical use, Springer, 1992.
[29] P.J. Kammer, G.A. Bolcer, Richard N. Taylor, A.S. Hitomi, M. Bergman, Techniques for Supporting Dynamic and Adaptive Workflow, Computer Supported Cooperative Work 9 (2000) 269–292.
[30] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, C. Barreto, Web services choreography description language version 1.0, W3C Working Draft 17 (2004), 10-20041217.
[31] M. Klein, C. Dellarocas, A knowledge-based approach to handling exceptions in workflow systems, Computer Supported Cooperative Work 9 (2000) 399–412.
[32] L. Li, High-level Petri net model of logic program with negation, IEEE Transactions on Knowledge and Data Engineering 6 (3) (1994) 382–395.
[33] S. Liang, P. Fodor, H. Wan, M. Kifer, OpenRuleBench: an analysis of the performance of rule engines, Proceedings of 18th International World Wide Web Conference, Madrid, Spain, 2009.
[34] R. Liu, A. Kumar, W. Van Der Aalst, A formal modeling approach for supply chain event management, Decision Support Systems 43 (3) (2007) 761–778.
[35] D. Madison, Process Mapping, Process Improvement and Process Management, Paton Press, 2005.
[36] A. Margaris, First order Mathematical logic, Courier Dover Publications, 1990.
[37] T. Morgan, Business rules and information systems: aligning IT with business goals, Addison-Wesley Professional, 2002.
[38] S. Mukherjee, H. Davulcu, M. Kifer, P. Senkul, G. Yang, Logic Based Approaches to Workflow Modeling and Verification, Logics for Emerging Applications of Databases, Springer, 2003, pp. 167–203.
[39] Oracle, State of the Business Process Management Market 2008, 2008.
[40] M. Reichert, P. Dadam, ADEPTflex: supporting dynamic changes of workflows without losing control, Journal of Intelligent Information Systems 10 (2) (1998) 93–129.
[41] R.G. Ross, The business rule approach, IEEE Computer 36 (5) (2003) 85–87.
[42] S.J. Russell, P. Norvig, J.F. Canny, J. Malik, D.D. Edwards, Artificial intelligence: a modern approach, Prentice Hall Englewood Cliffs, NJ, 1995.
[43] N. Russell, W.M.P.v.d. Aalst, A.H.M.t. Hofstede, D. Edmond, Workflow resource patterns: identification, representation and tool support, Proceedings of Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 05), Porto, Portugal, 2005.
[44] S.W. Sadiq, O. Marjanovic, M.E. Orlowska, Managing change and time in dynamic workflow processes, International Journal of Cooperative Information Systems 9 (1–2) (2000) 93–116.
[45] S. Sadiq, M. Orlowska, W. Sadiq, C. Foulger, Data flow and validation in workflow modelling, Proceedings of the Fifteenth Conference on Australasian Database, Dunedin, New Zealand, 2004, pp. 207–214.
[46] D. Sangiorgi, D. Walker, The pi-calculus: a theory of mobile processes, Cambridge University Press, 2001.
[47] A.-W. Scheer, ARIS — business process modeling, Third ed.Springer, 2000.
[48] T. Shimura, J. Lobo, T. Murata, An extended Petri net model for normal logic programs, IEEE Transactions on Knowledge and Data Engineering 7 (1) (1995) 150–162.
[49] E.A. Stohr, J.L. Zhao, Workflow automation: overview and research issues, Info. Systems Frontiers: Special Issue on Workflow Automation 3 (3) (2001) 281–296.

[50] S.X. Sun, J.L. Zhao, J.F. Nunamaker, O.R.L. Sheng, Formulating the data flow perspective for business process management, Information systems research 17 (4) (2006) 374–391.
[51] SWSF Committee, Semantic Web Service Ontology (SWSO): First-order Logic Ontology for Web Services (FLOWS), 2005.
[52] W.M.P. van der Aalst, The application of Petri nets to workflow management, Journal of Circuits, Systems and Computers 8 (1) (1998) 21–66.
[53] W.M.P. van der Aalst, Exterminating the dynamic change bug: a concrete approach to support workflow change, Information Systems Frontiers 3 (3) (2001) 297–317.
[54] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distributed and Parallel Databases 14 (3) (2003) 5–51.
[55] H.J. Wang, H. Wu, Supporting process design for e-business via an integrated process repository, Information Technology and Management (in press).
[56] H.J. Wang, J.L. Zhao, L.-J. Zhang, Policy-Driven Process Mapping (PDPM): discovering process models from business policies, Decision Support Systems 48 (1) (2009) 267–281.
[57] WfMC, XPDL 2.1 Specification, http://www.wfmc.org/index.php?option=com_docman&task=doc_details&gid=132&Itemid=126, 2008.
[58] P.Y.H. Wong, J. Gibbons, A process-algebraic approach to workflow specification and refinement, Lecture Notes in Computer Science 482 (9) (2007) 51.
[59] P.Y.H. Wong, J. Gibbons, A process semantics for BPMN, Proceedings of 10th International Conference on Formal Engineering Methods, 2008.
[60] M. zur Muehlen, J.V. Nickerson, K.D. Swenson, Developing Web Services Choreography Standards-The Case of REST vs. SOAP, Decision Support Systems 37 (2004) 2004.
[61] M. zur Muhlen, Organizational management in workflow applications — issues and perspectives, Information Technology and Management 5 (3) (2004) 271–291.

Harry Jiannan Wang is an Assistant Professor of Management Information Systems in the Lerner College of Business and Economics at the University of Delaware. He received Ph.D. in Management Information Systems from the Eller College of Management, University of Arizona, and B.S. in Management Information Systems from Tianjin University, China. His research interests involve business process management, workflow technologies and applications, services computing, and enterprise systems. He has published several research articles in academic journals and conferences such as Decision Support Systems, Communication of the AIS, Information Technology and Management, International Journal of Web Services Research, Journal of Information Systems and E-Business Management, IEEE IT Professional, International Conference on Information Systems (ICIS), Workshop on Information Technologies and Systems (WITS), and Americas Conference on Information Systems (AMCIS).

J. Leon Zhao is Head and Chair Professor in Information Systems, City University of Hong Kong. He was Eller Professor in the Department of Management Information Systems, University of Arizona before January 2009. He also taught previously at HKUST and College of William and Mary, respectively. He holds Ph.D. and M.S. degrees from the Haas School of Business, UC Berkeley, M.S. degree from UC Davis, and B.S. degree from Beijing Institute of Agricultural Mechanization. His research is on information technology and management, with a particular focus on workflow technology and applications in knowledge distribution, e-learning, supply chain management, organizational performance management, and services computing. His research has been supported by NSF, SAP, and other sponsors. He received an IBM Faculty Award in 2005 for his work in business process management and services computing and was awarded Chang Jiang Scholar Chair Professorship at Tsinghua University by the Ministry of Education of China in 2009. He has been associate editor of ACM Transactions on MIS, Information Systems Research, IEEE Transactions on Services Computing, Decision Support Systems, and Electronic Commerce Research and Applications. He has been chair or program chair for numerous conferences including the 5th International Conference on Design Science Research in Information Systems and Technology (DESRIST'10), the IEEE International Conference on Services Computing, Bangalore, India (SCC'09), the 2007 China Summer Workshop on Information Management (CSWIM'07), and the 2005 Workshop on Information Technology and Systems (WITS'05).