Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2006 Proceedings

International Conference on Information Systems (ICIS)

December 2006

Policy-Driven Process Mapping (PDPM): Towards Process Design Automation

Harry Wang
University of Delaware

Leon Zhao
University of Arizona

Leon Zhao
University of Arizona

Liang Zhang
IBM Research

POLICY-DRIVEN PROCESS MAPPING (PDPM): TOWARD PROCESS DESIGN AUTOMATION

Design Science

Harry J. Wang
University of Delaware
Newark, DE
hjwang@lerner.udel.edu

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

Liang-Jie Zhang
IBM Research
Hawthorne, NY
zhanglj@us.ibm.com

Abstract

Recently, many organizations are investing a great amount to revamp their business policies in order to comply with the new regulations related to Sarbanes-Oxley. For organizations with large-scale business processes, maintaining those processes to meet the changing policies is a non-trivial problem. In this paper, we present an innovative process mapping methodology named Policy-Driven Process Mapping (PDPM) that takes advantage of detailed organizational policies. In particular, we show how narrative process-related policies can be formalized and how process maps can be systematically created via a set of rules and algorithms. Our research contributes to business process management by providing a theoretical foundation for the development of advanced process mapping tools, thereby taking the first step toward automatable process design procedures.

Keywords: Process mapping, organizational policy, process design automation, business process management

Introduction

With the development of information technology, the evolution of quality management standards, and new regulatory requirements, there is an imperative demand for organizations to understand and document their business processes (Cobb 2004). Process mapping has been adopted globally as an effective technique to enable organizations to view their business system graphically at any level of detail and complexity (Madison 2005). Although many process mapping projects have successfully helped organizations achieve a higher level of cross-functional collaborations and tangible cost reduction, traditional process mapping methods tend to be resource-intensive and time-consuming (Hunt 1996; Reijers et al. 2003). Given the pressing requirements for Sarbanes-Oxley compliance, organizations need to discover their processes in a much more efficient manner, which calls for more advanced process mapping tools and methodologies that can support process design that incorporates automated design procedures.

In order to support e-business, public organizations have digitalized their business policies for distribution on the Internet (Wang et al. 2005). These business policies define how organizations operate their business, which includes rich information about processes (Peltier 2004). In this paper, we propose an innovative process mapping procedure that leverages business policies to build process maps, which we refer to as Policy-Driven Process Mapping (PDPM). In PDPM, narrative process policies are first formalized, and then a set of rules and algorithms are developed to systematically extract process maps from formalized process policies.

As shown in Figure 1, PDPM advocates a new way of discovering processes complementing the participative and analytical approaches (Cobb 2004; Datta 1998; Hunt 1996; Madison 2005; Reijers et al. 2003; Weske et al.

1999). Specifically, PDPM capitalizes on existing business policies to extract useful process information by leveraging narrative policies and intuitive rules. In addition, PDPM also builds a formal linkage between business policies and processes, which makes policy-process compatibility maintenance more systematic.

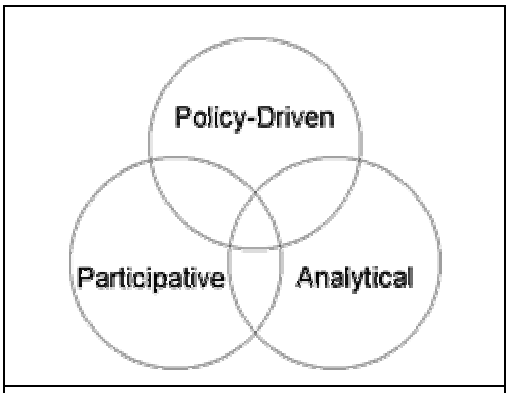

Figure 1. Process Mapping Approaches

In the next section, we briefly review the related literature. Then we formalize the concepts of process policy and process map. After that, a policy-driven process design procedure is presented and discussed. We then demonstrate and validate our PDPM approach via a case study and discuss the prototype system. Finally, we summarize our contributions and present our future research.

Literature Review

In the research of business process reengineering, identifying and analyzing existing organizational processes is the foundation for any further process changes and improvements (Kettinger et al. 1997). Process mapping is referred as the methodologies and related tools that help organizations identify, understand, and improve their AS-IS processes (Hunt 1996). Most process mapping projects are mainly conducted in a participative manner, where process information is obtained via extensive interviews, meetings, and workshops (Cobb 2004; Kettinger et al. 1997; Madison 2005). Participative process mapping projects can lead to detailed process information in the form of process policies; however, the collected data usually contains ambiguous, uncertain, and conflicting information (Reijers et al. 2003).

Different from a participative approach, an analytical approach aims to derive process model by using formal theory and techniques. Various formal methods such as linear programming (Aldowaisan et al. 1999), cost optimization (van der Aalst 2000), computational experiments (Hofacker et al. 2001), and probability theory (Datta 1998) have been applied to analytical process design and redesign. Workflow design based on data dependencies has also been proposed (Reijers et al. 2003; Sun et al. 2004). Those analytical approaches provide a mathematical foundation for the development of advanced process analysis tools.

Procedures of workflow application development have been discussed to achieve better quality and efficiency in workflow-related projects. Kwan and Balasubramanian (1998) proposed a workflow-aware information system development methodology by extending the traditional software engineering process with specific stages for workflow projects. A more comprehensive business process modeling methodology named ARIS (Architecture of Integrated Information Systems) is presented in Scheer (2000a), which has been adopted by companies such as SAP, IDS Scheer, etc. A detailed reference model for workflow application development processes is discussed in Weske et al. (1999). Although how to generate and analyze AS-IS process models has been identified as a major component in all the methodologies aforementioned, no formal procedure was given.

Most organizations have a large set of business policies in place to govern the way they conduct their business (Peltier 2004). Recent regulations related to Sarbanes-Oxley require organizations to clearly define and document their processes in organizational policies to increase internal control and governance (Cobb 2004). Wang et al. (2005) demonstrated that useful process information on control flow, data flow, and constraints can be extracted from business policies to construct process models.

The PDPM procedure presented in this paper fills a void in the workflow design research by providing a policy-driven process mapping (PDPM) approach for process analysts to extract process models from business policies. The PDPM approach incorporates features from both participative and analytical approaches and alleviates their drawbacks by leveraging systematic analysis of process policies. In this paper, we formalize the process design procedure, which provides a theoretical foundation for the policy-driven process mapping methodology. Our research goal is to develop a process design tool that provide automatable procedures that can help generate process maps from business process policies, thus eliminating incompatibilities between organizational policies and process maps.

Process Policy and Process Map Concepts

In this section, we first formalize concepts on process policy and process map to enable automatable process design procedures.

Definition 1 (Process Policy). Let P be a finite set of process policies, $P = \{p_1, p_2, ..., p_n\}$. We define a process policy $p_i \in P$ as an 8-tuple: $p_i =< pid_i, D_i, T_i, C_i, R_i, TS_i, TR_i, TD_i >$, where

- pid_i is the process policy ID, which is unique in order to identify the policy:
- $\forall p_i, p_j \in P, i \neq j, pid_i \neq pid_j$
- $D_i = \{d_1^i, d_2^i, ..., d_m^i\}$ is the set of data items related to p_i.
- $D = D_1 \cap D_2 \cap ... \cap D_n$ is the set of all data items in the process map.
- $T_i = \{t_1^i, t_2^i, ..., t_l^i\}$ is the set of tasks related to p_i. $T = T_1 \cap T_2 \cap ... \cap T_n$ is the set of all tasks in the process map.
- $C_i = \{c_1^i, c_2^i, ..., c_k^i\}$ is the set of constraints related to p_i. $C = C_1 \cap C_2 \cap ... \cap C_n$ is the set of all constraints in the process model. c_k^i is either an expression or a statement that returns a Boolean value.
- $R_i = \{r_1^i, r_2^i, ..., r_t^i\}$ is the set of resources related to p_i. $R = R_1 \cap R_2 \cap ... \cap R_n$ is the set of all resources in the process model. $r_t^i \in U \cap RL \cap OC \cap AG$, where U is set of users, RL is set of roles, e.g. general manager, salesperson, accountant, etc., OC is set of organizational units, e.g. bursar's office, graduate college, etc., and AG is set of machine agents, e.g. accounting system, ERP, etc.
- $TS_i = T_i \times T_i \times C_i = \{(t_a^i, t_b^i, c_c^i) : t_a^i$ is executed before t_b^i if c_c^i is true$\}$ is the set of task sequences related to p_i.
- $TR_i = T_i \times R_i = \{(t_l^i, r_k^i) : t_l^i$ is executed by $r_k^i\}$ is the set of task resource relationships in p_i.
- $TD_i = T_i \times D_i = \{(t_l^i, d_m^i) :$ d_m^i is used in $t_l^i\}$ is the set of task data relationships in p_i

Given an organization's travel reimbursement policy manual: *If Travel Reimbursement Form is submitted later than 60 days upon completing the travel, a Reasonable Exception Request Form must be submitted.* Assume the pid for this policy is 1. Then, two data items can be identified: d_1^1= "Travel Reimbursement Form (TRF)" and d_2^1="Reasonable Exception Request From (RERF)"; Two tasks can be extracted: t_1^1= "Submit TRF" and t_2^1="Submit RERF"; and there is one constraint: c_1^1= "TRF submission later than 60 days upon completing the travel". By applying process policy template, this policy can be expressed as p_1=<1, $\{d_1^1, d_2^1\}$, $\{t_1^1, t_2^1\}$, $\{c_1^1\}$, $\varnothing$, $\{(t_1^1, t_2^1, c_1^1)\}$, $\varnothing$, $\{(t_1^1, d_1^1), (t_2^1, d_2^1)\}$>. Note that because no resources can be identified by this policy, the corresponding sets are empty.

We introduce the definition of process map based on graph theory (Wilson 1996).

Definition 2 (Process Map). Given process policy $p_i =< pid_i, D_i, T_i, C_i, R_i, TS_i, TR_i, TD_i >$, a process map is defined as a directed graph $G =< V_t(G), V_c(G), V_{se}(G), A(G) >$, where:

- task vertex set $V_t(G) = \{(v_t, D_t, R_t) : v_t \in T, D_t \subset D, R_t \subset R, D_t \neq \varnothing, R_t \neq \varnothing\}$ is a non-empty finite set of all tasks in the process map and each task is associated with a set of data items D_t and resources R_t. $\forall v_t \in V_t(G), d^-(v_t) \geq 1, d^+(v_t) = 1$, where $d^-(v_t) / d^+(v_t)$ is v's in-degree/out-degree.
- decision vertex set $V_c(G) = C$ is a finite set of all constraints in the process map, and $\forall v_c \in V_c(G), d^-(v_c) \geq 1, d^+(v_c) = 2$.
- start/end vertex set $V_{se}(G) = \{s, e\}$, where s is the start node and e is the end node in the process map, and $d^-(s) = 0, d^+(s) = 1, d^-(e) \geq 1, d^+(s) = 0$.
- directed arc set $A(G)$ is a non-empty finite set of distinct ordered pairs of distinct elements of vertex set $V(G) = V_t(G) \cup V_c(G) \cup V_{se}(G)$

Besides enabling process analysis, Definition 2 also implies some assumptions we make in order to simplify the structure of process maps: 1) tasks are executed sequentially, and task parallelism can be introduced later to improve process performance; 2) each decision vertex has two and only two outgoing arcs, which means that each decision vertex is an exclusive OR and its value is a Boolean value returned by corresponding process constraint. It has been proved that any OR vertex can be converted to exclusive OR vertex (Bi et al. 2004); 3) a task can have only one outgoing arc and at least one incoming arc; 4) an end node can have more than one incoming arcs. As we will discuss later, a set of rules are developed to ensure that the process map follows the assumptions. Next, we show a systematic procedure to extract process maps from given process policies.

Policy-Driven Process Extraction Procedure

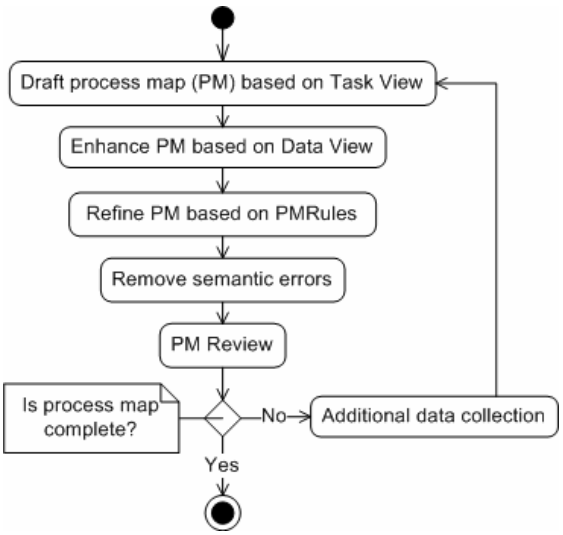

Figure 2. Detailed Process Map Extraction Procedure

Figure 2 shows the procedure to extract a process map from formalized process policies. We propose a set of new concepts to facilitate the extraction procedure, namely, task view, data view, and process map rules. Each step of the procedure is discussed in detail next:

Step 1: Draft process map based on task view. In this step, the first draft of process map is generated based on the task sequences *TS* identified from policy analysis. In particular, a task view is first constructed based on *TS* to simplify the drafting process.

Definition 3 (Task View). Given $T = \{t_1, t_2, ..., t_n\}$ are identified tasks and $C = \{c_1, c_2, ..., c_m\}$ are identified constraints. A task view *TV* is defined as a *(m+n+1)*-square matrix, whose rows and columns correspond to the set

$T \cup C \cup \{s\}$ and $T \cup C \cup \{e\}$ where s is the start node and e is the end node. tv_{ij}=1 and if the corresponding task, constraints, or start/end nodes are connected, otherwise tv_{ij} = 0.

Given the initial process map G, which is an empty graph, and task view *TV*, the algorithm shown in Table 1 is used to draft the process map.

Table 1. Algorithm for Drafting Process Map based on Task View

Algorithm 1:
Input: initial process map $G =< V_t(G), V_c(G), V_{se}(G), A(G) >$
where, $V_t(G) = V_c(G) = V_{se}(G) = A(G) = \varnothing$,
and task view *TV, where* $T = \{t_1, t_2, ..., t_n\}$, $C = \{c_1, c_2, ..., c_m\}$
Output: draft process map G'
Begin
$V_{se}(G) = \{s, e\}$ // add start and end node
$V_c(G) = C$ // add decision points
$V_t(G) = \{(v_t, D_t, R_t) : v_t \in T, D_t = R_t = \varnothing\}$ // add tasks
for each $tv_{ij} \in TV$ // add arcs among tasks and decision points based on task view
if $tv_{ij} = 1$ and $i < n, j < n$ then add (t_i, t_j) to $A(G)$
if $tv_{ij} = 1$ and $n < i < n + m, j < n$ then add (c_i, t_j) to $A(G)$
if $tv_{ij} = 1$ and $i < n, n < j < n + m$ then add (t_i, c_j) to $A(G)$
if $tv_{ij} = 1$ and $n < i < n + m, n < j < n + m$ then add (c_i, c_j) to $A(G)$
end for
$G' =< V_t(G), V_c(G), V_{se}(G), A(G) >$
End

Based on the task view, a preliminary process map can be easily drafted. For instance, a *TV* matrix with three tasks and two constraints is mapped into a draft process map in Figure 3. Normally task view only provides limited information on the process map, and additional enhancements and refinements are needed to complete the process map.

TV	t_1	t_2	t_3	c_1	c_2	e
t_1	0	1	0	0	0	0
t_2	0	0	0	0	0	0
t_3	0	0	0	0	0	0
c_1	0	0	1	0	0	0
c_2	1	0	0	0	0	0
s	0	0	0	0	0	0

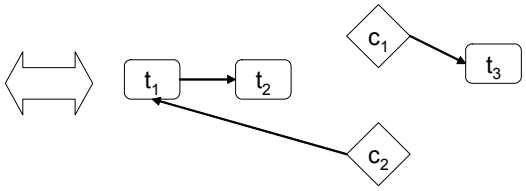

Figure 3. Drafting Process Map based on Task View

Step 2: Refine the Process Map based on Data View. In this step the draft process map *G'* is enhanced based on the identified task data relationship *TD*. A data view is constructed to help this process, which is defined as following:

Definition 4 (Data View). Given $T = \{t_1, t_2, ..., t_n\}$ are identified tasks, $C = \{c_1, c_2, ..., c_m\}$ are identified constraints, and $D = \{d_1, d_2, ..., d_k\}$ are identified data items. A data view *DV* is defined as a $(m+n) \times k$ matrix whose rows correspond to the set $T \cup C$, and whose columns correspond to the set D. dv_{ij} = O (dv_{ij} = I) if d_j is the output (input) of corresponding task or constraint, otherwise dv_{ij} = 0.

Based on data view, additional arcs between tasks and decisions can be identified. Intuitively, if a data item *d* is the output of task t_1 and input of t_2, then an arc (t_1,t_2) should be added. Formally, the algorithm shown in Table 2 is used to refine process map G'. Note that the task view is rebuilt based on data view at the end of the algorithm.

Table 2. Algorithm for Process Map Enhancement based on Data View

Algorithm 2:
Input: draft process map $G' =< V_t(G), V_c(G), V_{se}(G), A(G) >$
data view *DV, where* $T = \{t_1, t_2, ..., t_n\}$, $C = \{c_1, c_2, ..., c_m\}$, $D = \{d_1, d_2, ..., d_k\}$
Output: refined process map G''
Begin
  var TAIL // a set variable to store tails for new arcs
  var HEAD // a set variable to store heads for new arcs
for $j = 1...k$
for $i = 1...(n+m)$
  $dv_{ij} \in DV$
  if $dv_{ij} = O$ and $i < n$ then add t_i to TAIL
if $dv_{ij} = O$ and $n < i < n+m$ then add c_i to TAIL
if $dv_{ij} = I$ and $i < n$ then add t_i to HEAD
if $dv_{ij} = I$ and $n < i < n+m$ then add c_i to HEAD
  end for
end for
for $tail \in TAIL, head \in HEAD$
if $(tail, head) \notin A(G)$ then add (*tail,head*) to $A(G)$
end for
$G'' =< V_t(G), V_c(G), V_{se}(G), A(G) >$
run Algorithm 1 // rebuild the task view based on data view
End

For example, Figure 4 shows how the draft process map in Figure 3 can be enhanced based on a data view.

TV	d_1	d_2	d_3
t_1	0	0	**O**
t_2	**O**	0	**I**
t_3	0	**O**	0
c_1	**I**	0	0
c_2	0	**I**	0

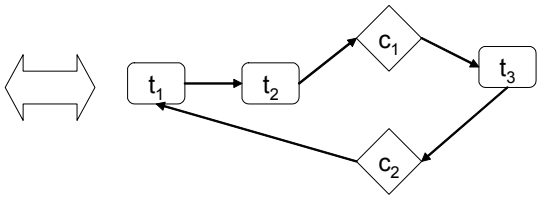

Figure 4. Enhance Process Map based on Data View

Step 3: Refine process map based on Process Map Rules (PMRules). The previous two steps try to get as much information on the process map as possible by adding arcs among tasks and decision points, which may result in a process map that does not confine to the process map definition specified in Definition 2. Such process maps are usually not well organized and hard to understand. Thus, we say a process map is ***syntactically correct***, if it satisfies all process map structure properties defined in Definition 2, which is formally specified in Definition 5.

Definition 5 (Syntactical Correctness of Process Map). Given task view *TV, where* $T = \{t_1, t_2, ..., t_n\}$, $C = \{c_1, c_2, ..., c_m\}$, a process map is syntactically correct if and only if:

$i = 1,...,n,n+m+1 \sum_{j=1}^{n+m+1} tv_{ij} = 1$ // single outing arc for tasks and start node

and $j = 1,...,n+m+1, \sum_{i=1}^{n+m+1} tv_{ij} \geq 1$ // at least one incoming arc for tasks, decisions, and end node

and $i = n+1,...,m, \sum_{j=1}^{n+m+1} tv_{ij} = 2$ // two outgoing arcs for decisions

In this step, the enhanced process map *G''* is further refined according to a set of *PMRules* for syntactical correctness. In particular, the following PMRules are enforced:

PMRule 1 (Single Task Outgoing Arc). If there exists tasks that have more than one outgoing arcs, those links must be merged. Given tasks in G'' are connected to either other tasks or decisions, we use the following two sub PMRules to enforce PMRule 1.

PMRule 1.1 (No One-to-Many Task-Task Arcs). If a task *t* links to more than one other tasks directly, the other tasks must be sequentially linked and connected to *t*.

PMRule 1.2 (No One-to-Many Task-Decision Arcs). Similar to PMRule 1.1, if a task *t* links to more than one decision points directly, those decision points must be sequentially linked and connected to *t.*

PMRule 2 (Boolean Decision Outgoing Arc). If a decision point has less than two outgoing arcs, additional arcs need to be added to link the decision point to other tasks or decision points. According to process policy definition, a constraint returns a Boolean value. Therefore, a decision point must have two and only two outgoing arcs.

PMRule 3 (Close Start/End Node). All tasks and decisions without incoming arc should be connected to the start node; and all tasks without outgoing arc should be connected to the end node; After that, if more than one task/decision is linked to the start node, those tasks/decisions need to be sequentially linked first and then connected to the start node; and if there are still no arcs linked to start and end nodes, link them directly.

Step 4: Remove semantic errors. Semantic error is defined as inappropriate arcs resulting in process maps that are logically wrong. For example, a direct arc from start node to end node represents a semantic error. In this step, a thorough walkthrough of the process map should be conducted to identify and remove semantic errors if any. Semantic errors inspection relies mainly on human intelligence, although some simple errors, such as the start-to-end arc aforementioned, can be identified by computers.

Step 5: Process review. In this step, the process map is reviewed to ensure it captures all tasks and constraints. This step is crucial to achieve the completeness of the process map, especially when the process policies are not very detailed. Process review usually includes review meetings with process owners, e.g. managers who oversee the process, and people who are conducting process operations. Given that a process map has been developed based on existing process policies, this review is more focused and efficient than data collections in traditional participative approach.

To the best of our knowledge, the policy-driven process mapping procedure presented in this section is the first attempt to formalize the procedure of process mapping based on policy analysis. PDPM advocates a new process mapping methodology and creates a critical step toward process design automation. In the next section, we demonstrate and validate the PDPM approach via a case study.

A Case Study: Validation of PDPM

In our previous research, we conducted a case study of a major public university. A set of process policies is shown in Table 3 (Wang et al. 2005). In this section, we show how those twelve narrative process policies are transformed into a process map using the PDPM approach.

Table 3. Identified Process Policies from a University Policy Manual

P1: All claims for the reimbursement of expenses for an approved university business travel are made on a Travel Reimbursement Form (TRF).
P2: If Travel Reimbursement Form is submitted later than 60 days upon completing the travel, a Reasonable Exception Request Form (RERF) must be submitted.
P3: If reasonable exception request is not approved, then the reimbursement amount will be taxable.
P4: If reimbursement amount (RA) exceeds the limit, a Travel Exception Form (TEF) must be filled.
P5: After the reimbursement form is approved, a check will be issued.
P6: If the quarterly travel exception review is denied, the reimbursed amount must be refunded back.
P7: Department or unit head must approve the travel.
P8: Department or unit head must approve the reasonable exception.
P9: Disbursement Services Center (DSC) must review the reimbursement form.
P10: The Office of Business and Financial Services (OBFS) reviews the travel exception form.
P11: Higher Education Travel Control Board (HETC) quarterly reviews the travel exceptions.
P12: Bursar's office (BO) is responsible to issue reimbursement check.

First, we formalize the policies according to process policy definition specified in Definition 1 as shown in Table 4. In particular, 11 tasks, 5 constraints, 5 data items, and 6 resources are identified.

Table 4. Formalized Travel Reimbursement Process Policies

PID	Data Items (D)	Tasks (T)	Constraints (C)	Resource	Task Sequences (TS)	Task Resources (TR)	Task Data (TD)
1	d_1 = TRF	t_1 = Submit d_1	NA	r_1 = traveler	NA	(t_1, r_1)	(t_1, d_1)
2	d_1 d_2 = RERF	t_1 t_2 = Submit d_2	c_1: TRF is submitted later than 60 days upon completing the travel	NA	(t_1, t_2, c_1)	(t_2, r_1)	(t_2, d_2)
3	d_2 d_3 =RA	t_3 = Approve d_2 t_4 = Make d_3 taxable	c_2: RERF is approved	NA	(t_3, t_4, c_2)	NA	(t_3, d_2)
4	d_3 d_4 = TEF	t_5 = Submit d_4	c_3: Reimbursement Amount exceeds the limit	NA	(null, t_5, c_3)	(t_5, r_1)	(t_5, d_4)
5	d_1 d_5 = Check	t_6 = Approve d_1 t_7 = Issue d_5	c_4:TRF is approved	NA	(t_6, t_7, c_4)	NA	(t_6, d_1) (t_7, d_6)
6	d_4, d_5	t_8 = Review d_4 quarterly t_9 = Refund d_5	c_5: TEF quarterly review is not passed	NA	(t_8, t_9, c_5)	(t_9, r_1)	(t_8, d_4) (t_9, d_5)
7	d_1	t_6	NA	r_2 = Dept. Head	NA	(t_6, r_2)	NA
8	d_2	t_3	NA	NA	NA	(t_3, r_2)	NA
9	d_1	t_{10} = Review d_1	NA	r_3 = DSC	NA	(t_{10}, r_3)	NA
10	d_4	t_{11} = Review d_4	NA	r_4 = OBFS	NA	(t_{11}, r_4)	NA
11	d_4	t_8	NA	r_5 = HETC	NA	(t_{12}, r_5)	NA
12	d_5	t_7	NA	r_6 = BO	NA	(t_7, r_6)	NA

Based on the task sequences in Table 4 (which corresponds to a task view that is omitted due to space limitations), a draft process map is constructed as shown in Figure 5. The draft process map contains limited information on the relationship among tasks and constraints, and we use the data view to enhance the draft process map. Table 5 shows the data view of travel reimbursement process. Based on the algorithms in Table 2, additional arcs are added, and the enhanced process map is depicted in Figure 6. However, the enhanced process map is not syntactically correct. For instance, t1 has five outgoing arcs and start node has no outgoing arc. Additional refinements are conducted using the PMRules. For example, task t1 has no incoming arc, which should link to the start node. Similarly, t9 has no outgoing arc, which should be linked to the end node. One outgoing arc is added to each decision point to represent the Boolean value returned by each constraint. Finally, the revised process map (Figure 7) is derived by enforcing all PMRules. Note that in this step, some arcs, which are identified from the task view and the data view, have been removed.

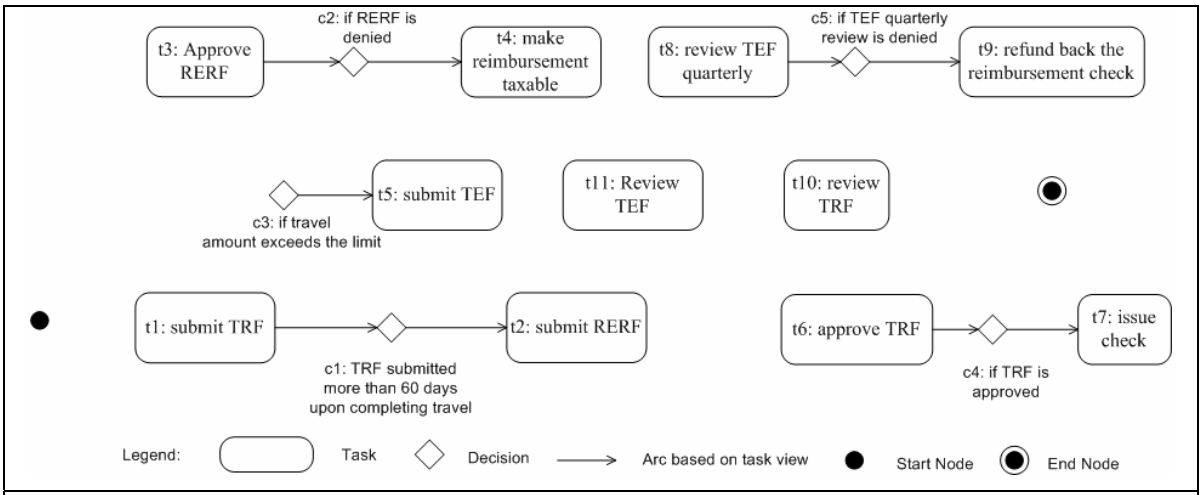

Figure 5. Draft Process Map based on the Task Sequences in Table 4

Table 5. Data View of Travel Reimbursement Process

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	c1	c2	c3	c4	c5
d1	O					I				I		I			I	
d2		O	I										I			
d3	O			I										I		
d4					O			I			I					I
d5							O		I							

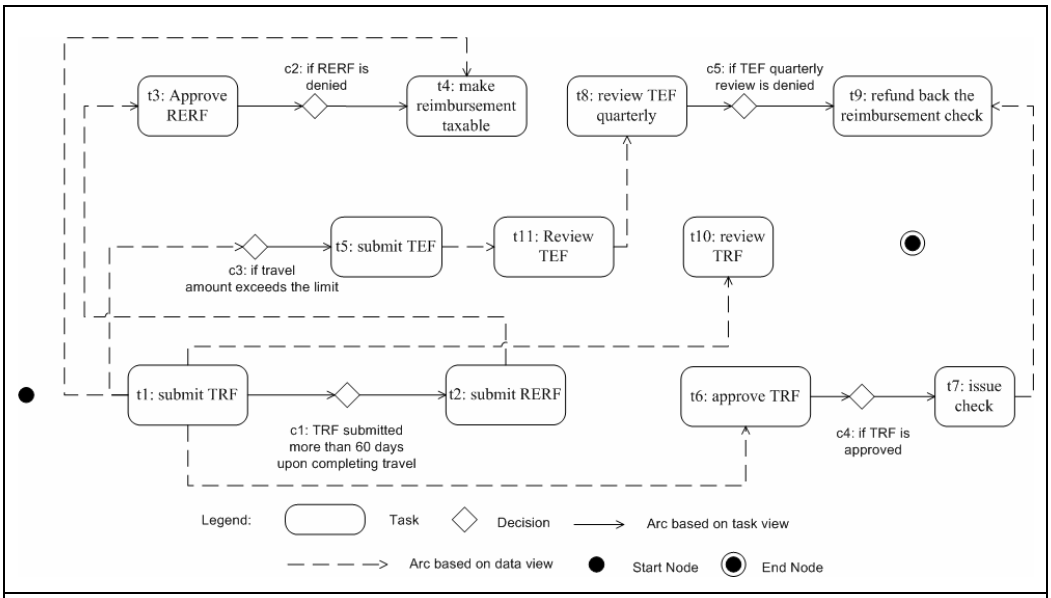

Figure 6. Enhanced Process Map based on Data View

Next, according to PDPM, two semantic errors are identified in Figure 7: t7 (issue the check) is linked to t9 (refund back the check), which is logically wrong, and t11(Review TEF) is linked to t8 (Review TEF quarterly), which results in the corresponding reimbursement request form never gets reviewed in t10. To remove those semantic errors, two constraints c6 and c7 are added, as shown in Figure 8. This updated process map contains all the necessary tasks and constraints for the travel reimbursement process, leading to the final process map.

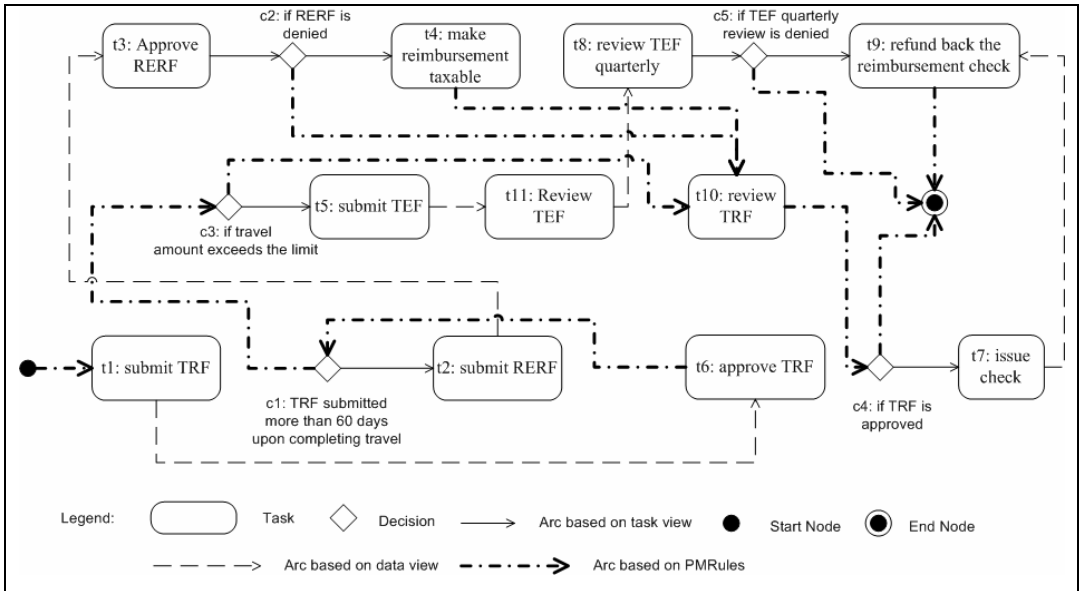

Figure 7. Revised Process Map based on PMRules

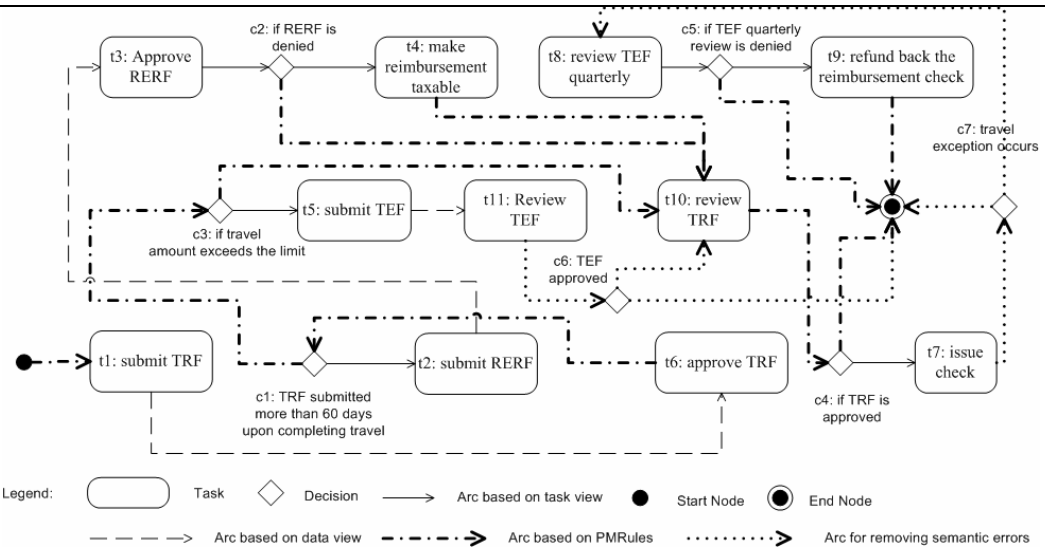

Figure 8. Process Map after Removing Semantic Errors

The process derivation procedure can also be depicted conveniently with a process matrix in Table 6. Initially, the process matrix contains the P's based on the given process policies. Then, the matrix is updated by adding D's for the arcs identified from the data view in Table 5. Next, the semantic analysis removes a number of arcs as represented in the process matrix by the symbols of P→0 and D→0, respectively. The arcs added after applying all PMRules are depicted with the R's. While the process maps in Figures 5-8 are easy to read by process analysts, the process matrix is easier to automate with computer procedures. Note that the process matrix is closely related to the task view defined previously. Next, we present a prototype system based on PDPM, which will be used by process analysts.

Table 6. Process Matrix of the Travel Reimbursement Process Design

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	c1	c2	c3	c4	c5	e
t1	0	0	0	**D→0**	0	**D**	0	0	0	**D→**	0	**P→0**	0	**D→0**	**D→0**	0	0
t2	0	0	**D**	0	0	0	0	0	0	0	0	0	**D**	0	0	0	0
t3	0	0	0	0	0	0	0	0	0	0	0	0	**P**	0	0	0	0
t4	0	0	0	0	0	0	0	0	0	**R**	0	0	0	0	0	0	0
t5	0	0	0	0	0	0	0	**D→0**	0	0	**D**	0	0	0	0	**D→0**	0
t6	0	0	0	0	0	0	0	0	0	0	0	**R**	0	0	**P→0**	0	0
t7	0	0	0	0	0	0	0	0	**D**	0	0	0	0	0	0	0	0
t8	0	0	0	0	**P**	0	0	0	0	0	0	0	0	0	0	0	0
t9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**R**
t10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**R**	0	0
t11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c1	0	**P**	0	0	0	0	0	0	0	0	0	0	0	**R**	0	0	0
c2	0	0	0	**P**	0	0	0	0	0	**R**	0	0	0	0	0	0	0
c3	0	0	0	0	**P**	0	0	0	0	**R**	0	0	0	0	0	0	0
c4	0	0	0	0	0	0	**P**	0	0	0	0	0	0	0	0	0	**R**
c5	0	0	0	0	0	0	0	0	**P**	0	0	0	0	0	0	0	**R**
s	**R**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

System Architecture and Prototype

As shown in Figure 9, our Web-based PDPM prototype system consists of three major components, namely, PMWizard, PMVisualizer, and PMExporter. We use Oracle 10g to store formalized process policies, which is

referred as the *Process Policy Base*. Figure 10 shows the conceptual model of process policies. The corresponding database schema is shown in Table 7 with primary keys underlined.

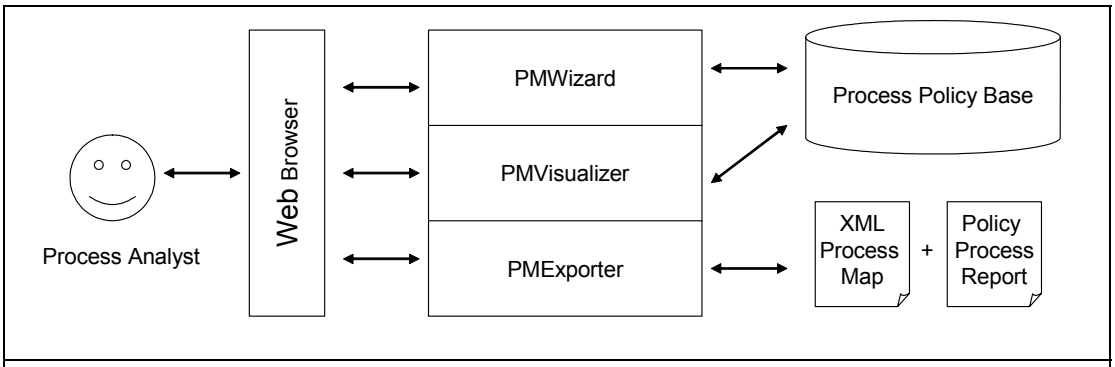

Figure 9. PDPM Prototype System Architecture

Table 7. Database Schema for Process Policy	
Policy (PID, PType, Keywords, Memo)	PolicyData(PID, DID)
Task (TID, TName)	PolicyResource(PID, RID)
DataItem (DID, DName)	PolicyConst(PID,CID)
PConstraint (CID, Content)	TaskSequence (TID1, TID2, CID)
PResource (RID, RName, Type)	TaskData (TID, DID)
PolicyTask(PID, TID)	TaskResource (TID, RID)

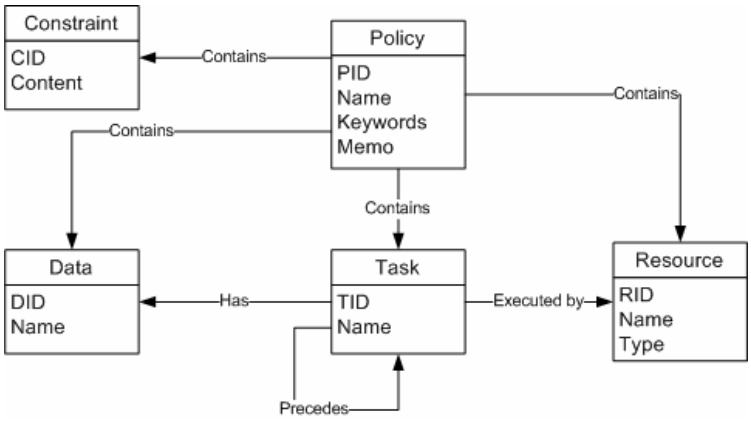

Figure 10. Entity-Relationship Diagram for Process Policy

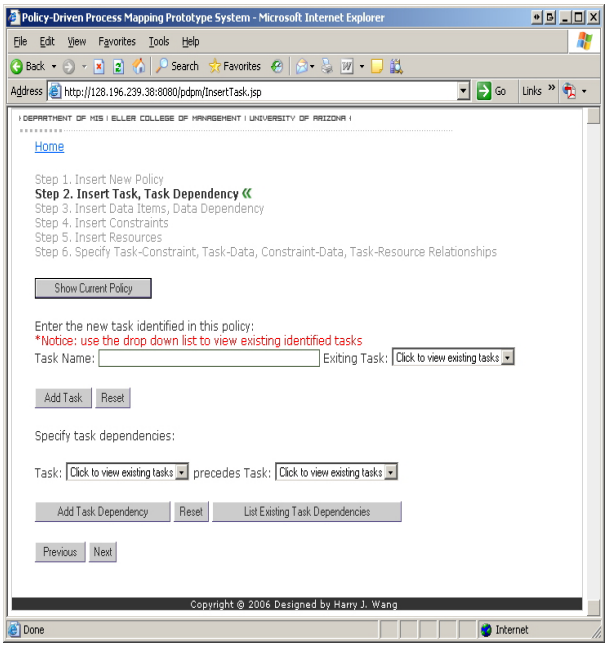

Figure 11. Screenshot of PMWizard

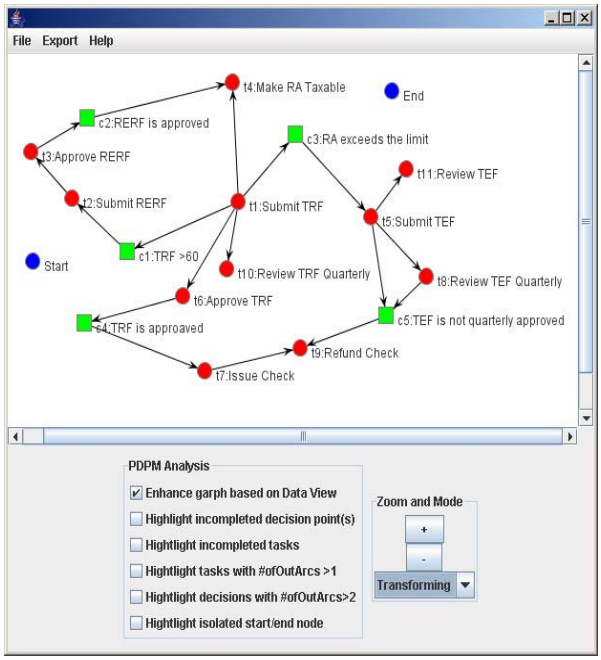

Figure 12. Screenshot of PMVisualizer

PMWizard is a process policy analysis tool (Figure 11) that provides step-by-step instructions for process policy formalization. When a new process policy is entered, it is stored according to the process policy schema in Table 7. *PMVisualizer* is implemented on top of Java Universal Network/Graph Framework (JUNG) (http://jung.sourceforge.net/) with special extensions for policy-based process mapping. As shown in Figure 12, the draft travel reimbursement process map (see also Figure 6) is shown as a directed graph. *PMVisualizer* allows process analysts to visually edit process map, e.g. adding/deleting vertices and arcs. *PMExporter* is used to export a process map into an XML format for exporting to other systems and generate process policy reports.

Conclusions

In this paper, we propose an innovative process mapping approach by means of formal process policy analysis, which is referred as Policy-Driven Process Mapping (PDPM). Our research contributions are three-fold: First, PDPM is a new process mapping methodology different from the existing participative and analytical process mapping methods. Second, we formalized the process policy and process map, developed several new concepts such as task view, data view, and process map rules, and designed a set of process mapping algorithms. These artifacts build the theoretical foundation for systematic process policy analysis and process mapping automation. Finally, we implemented a prototype system to validate and demonstrate the PDPM approach.

We are currently continuing our research in a number of directions. First, we are extending our approach by relaxing the assumptions to support parallelism and allow more complex decision vertices. Second, we are working on the classification of process semantic errors and mechanisms to identify those errors. Third, we plan to conduct user studies to test the usability of our prototype system and further validate PDPM approach by comparing it with other approaches in terms of cost and benefits.

References

Aldowaisan, T.A., and Gaafar, L.K. "Business process reengineering: an approach for process mapping," *Omega* (27:5) 1999, pp 515-524.

Bi, H., and Zhao, L. "Process logic for verifying the correctness of business process models," P*roceedings of the International Conference on Information Systems (ICIS 2004)*, Washington, D.C., 2004.

Cobb, C.G. *Enterprise Process Mapping: Integrating Systems for Compliance and Business Excellence* ASQ Quality Press, 2004, p. 128.

Datta, A. "Automating the discovery of AS-IS business process models: Probabilistic and algorithmic approaches," *Information Systems Research* (9:3) 1998, pp 275-301.

Hofacker, I., and Vetschera, R. "Algorithmical approaches to business process design," *Computers & Operations Research* (28:13) 2001, pp 1253-1275.

Hunt, V.D. *Process Mapping : How to Reengineer Your Business Processes* Wiley, 1996, p. 288.

Kettinger, W.J., Teng, J.T.C., and Guha, S. "Business process change: A study of methodologies, techniques, and tools," *MIS Quarterly* (21:1) 1997, pp 55-80.

Madison, D. *Process Mapping, Process Improvement and Process Management* Paton Press, 2005, p. 320.

Peltier, T.R. *Information Security Policies and Procedures: A Practitioner's Reference*, (2 ed.) Auerbach Publication, 2004.

Reijers, H.A., Limam, S., and van der Aalst, W.M.P. "Product-based workflow design," *Journal of Management Information Systems* (20:1), Sum 2003, pp 229-262.

Scheer, A.-W. *ARIS - Business Process Modeling*, (Third ed.) Springer-Verlag, 2000a.

Sun, S.X., and Zhao, J.L. "A Data Flow Approach to Workflow Design," *Proceedings of the Workshop on Information Technologies and Systems (WITS)*, Washington D.C., 2004.

van der Aalst, W.M.P. "Reengineering knock-out processes," *Decision Support Systems* (30:4) 2000, pp 451-468.

Wang, H.J., and Zhao, J.L. "Policy-driven Business Process Modeling in E-business," *Proceedings of the Fourth Workshop on E-business (WeB 2005)*, Las Vegas, Nevada, 2005.

Weske, M., Goesmann, T., Holten, R., and Striemer, R. "A Reference Model for Workflow Application Development Processes," *Proceedings of the International Conference on Work activities Coordination and Collaboration*, 1999, pp. 1-10.

Wilson, R.J. *Introduction to Graph Theory*, (4th ed.) Longman Group Ltd, Harlow, UK, 1996.