Role-activity Diagrams Modeling Based on Workflow Mining

Weidong Zhao[1], Weihui Dai[2+], Anhua Wang[1], Xiaochun Fang[1]
[1]*Software School,Fudan University, Shanghai, 200433, China*
[2]*Management School,Fudan University, Shanghai, 200433, China*
[+]*Corresponding author: whdai@fudan.edu.cn*

Abstract

Role-activity diagram (RAD) is a basic role-oriented process model, but there lacks of an objective role identification method during RAD modeling. By means of the analysis of workflow logs-- workflow mining, the ratio of activities performed by actors is used to describe their work. The role is, therefore, identified by regarding actors with similar activities, who undertaking the same responsibility. On the basis of role identification, social network diagrams among actors are analyzed by considering activity dependence between them and their interactions are discussed. In this way, we can get role-activity diagrams. Experiments show that the proposed method is viable.

1. Introduction

Process modeling is the important step in process management, particularly when business processes are complex and many actors are involved. In comparison with widely used activity-based process modeling techniques, role-oriented process models describe business processes as actors and their interactions from the organization perspective [1,2]. For example, RolEnact describes business processes using role, state, activity and event [3]. In this approach, a certain role performs activities and changes the state of itself and other roles. Another typical role-oriented process modeling approach is role-activity diagrams (RAD), which are adapted from cross-functional process models using sub-processes to describe process roles' responsibility so as to highlight the interaction between roles [4]. However, RAD modeling is far from trivial: it requires a deep knowledge of processes and organizational model so as to identify roles. Until now, how to define roles is a difficult task. The few proposals for defining role are subjective, which usually refine job functions defined by business experts into roles.

Workflow management systems (WfMS) produce lots of logs, which contain useful information of activities and actors. Usually, process mining can reconstruct process models from real executions, i.e. workflow logs [5]. Thus workflow mining may be a more objective method for role identification. Since each business process has its business goal, which can be decomposed and allotted to different roles who cooperate with each other, we adopt process mining technique based on two hypotheses: actors playing the same role have similar duties, abilities and characteristics; activities performed by these actors stand for their work. So roles can be identified by discovering actors with similar work from workflow logs.

There exist some workflow mining methods. For example, the α algorithm can mine the WF-net, which denotes the dependence between activities [6]. Some researchers focus on organizational mining: actors are the center of business processes and social networks can be drawn out by analyzing business relations among actors [7]. Actors have to access applications to do their jobs and therefore need permissions. As an approach to role engineering, the role-mining tool ORCA aggregates permissions bottom-up into a role hierarchy using clustering analysis interactively with the aid of business experts [8]. Some publications also discuss role finding to implement role-based security administration based on access rights of roles in role engineering. In addition, Liu and Shen derive process views, which provide different process participants with various process information from activity-based process models through an order-preserving, bottom-up aggregation of activities [9]. Herein, roles include different levels or departments within an organization. The references reviewed above lay a foundation for RAD modeling. Roles describe tasks within business processes and they are implicitly in use. Taking the work similarity of actors as a criterion, workflow mining is used for role identification in this paper. It identifies roles by regarding the activities with more similarity as a sub-process corresponding to the same role. On the basis of role mining, the interaction between roles are analyzed through social network diagrams among actors, and finally role-activity diagrams is mined.

The remainder of this paper is organized as follows: Section 2 introduces some basic concepts for our approach. Section 3 gives a detailed analysis of role mining by two proposed algorithms and then social networks are used to analyze the interaction between roles for RAD modeling. Conclusions and future work are finally made in Section 4.

2. Basic concepts

Some basic definitions are first introduced in this section.

(1) Dependence between activities

Dependence between activities is hidden in workflow logs. The logs contain some information of activities and actors during the run-time of workflow instances. The set of all activities in workflow logs are denoted as $A = \{a_1, a_2,\ldots, a_m\}$, where $m = |A|$ is the number of activities in workflow logs. And the set of all process actors is denoted as $P = \{p_1, p_2,\ldots, p_n\}$, where n is the number of

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.992

actors. I is the set of all workflow instances in logs, and for any workflow instance $i \in I$, $i = (a_1 a_2 \ldots a_k)$, AS(i) is the set of activities in the instance i. Herein, we treat every activity as an atomic event with time interval, and assume that the sequence of activities $a_1 a_2 \ldots a_k$ represents the actual execution ordering.

Definition 1 We call that activity b depends on activity a, denoted as a>b, if $\exists i=(a_1 a_2 \ldots a_k) \in I$, $u \in \{1, 2, \ldots, k-1\}$, such that $a_u=a$, $a_{u+1}=b$. According to definition, a>b means that activity b is executed after activity a and there exists no other activities between.

Definition 2 We call activity b depends on activity a directly (direct dependence), if for each workflow instance $i \in I$, there only exists a>b but not b>a. That is $a \rightarrow b \Leftrightarrow a>b \text{ and } \forall i \in I, \neg \exists b>a$. It means that the dependence between activity a and b is consistent in all workflow instances. Moreover, it hints that there exists state transition from activity a to b.

Definition 3 Given I, we can find all the activity pairs, which accord with the direct dependence. The activity pairs is denoted as DEP, where DEP={(a, b)|$\forall a, b \in A$, a→b}. The dependence between activities is helpful to identify sub-processes corresponding to process roles, and it can be used for justifying whether it satisfies the order preservation defined in [9]. Table 1 shows a segment of the clothing production workflow logs produced by the Staffware system in a clothing factory. The activities include a, b, c, d, e, f and g: activity a is inquiry and quoting, b is contract signing, c is prototype designing, d is fabric requiring, e is fabric drawing, f is fabric purchasing and g is fabric processing. In table 1, the actors are numbered as p_{101}, p_{105}, p_{106}, p_{107}, p_{115} etc. respectively.

Table 1 segment of workflow logs

instance ID	activities	actors
115	a	p_{101}
115	b	p_{115}
116	a	p_{107}
115	c	p_{114}
116	b	p_{113}
117	a	p_{101}
115	d	p_{105}
116	d	p_{106}
115	e	p_{106}
117	b	P_{115}

Suppose that there are six workflow instances in the workflow log: i_1 = {abcg}, i_2 = {abdecg}, i_3 = {abcdeg}, i_4 = {abdeg}, i_5 = {abdceg}, i_6 = {abdfecg}, we can get the WF-net model as shown in Figure 1 using the α algorithm proposed in reference [6]. In this model, rectangulars denote activities, circles stand for states while arrows represent the state transition. Initially, the WF-net is in the start-state, states change continuously after activities are executed. As we see from Figure 1, there exist state transitions from the start-state to activity a until from g to the end-state.

The DEP of the clothing production workflow in Figure 1 is {(a, b), (a, c), (a, d), (a, e), (a, f), (a, g), (b, c), (b, d), (b, e), (b, f), (b, g), (c, g), (d, e), (d, f), (d, g), (e, g), (f, e), (f, g)}. Note that c→d and d→c do not hold because activity c>d and activity d>c both appear in the workflow log.

Definition 4 Adjacent activity set adjAS(b)={a|$a,b \in A_v$, $a \notin S$, $b \in S$, a>b or b>a, $S \subseteq A_v$}, where S and A_v are two activity sets. Definition 4 shows that for any activity in adjAS(b), it belongs to A_v but not S, and there exists dependence between the activity and other activities in S.

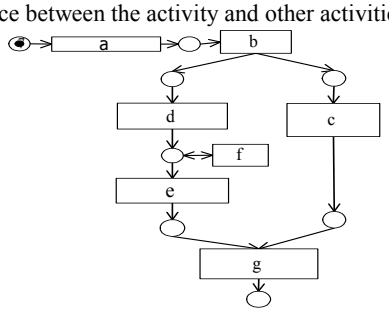

Figure 1 clothing production workflow

(2) Activity execution by actors

In this paper, roles are identified by the ratio of the times actors executing each activity to the total times of these actors executing all activities.

Definition 5 Activity execution frequency of an actor is defined as

$$R(p_i, a_j) = \frac{times(p_i, a_j)}{times(p_i)} \quad (1)$$

In the equation (1), $R(p_i, a_j)$ denotes the percentage of activity a_j in the work of actor p_i, $times(p_i, a_j)$ is the times of p_i executing a_j, while $times(p_i)$ is the total of p_i executing related activities. For example, if a production director named Tom has executed activities in the clothing production process discussed above 100 times in total, in which "Fabric requiring" is executed 30 times, thus R(Tom, Fabric requiring) = 0.3.

Definition 6 Vector of activity execution in the work of the actor p_i is defined as $S(p_i, A_u)=(R(p_i, a_1), R(p_i, a_2), \ldots, R(p_i, a_k))$, where k is the number of activities in A_u, $a_j \in A_u$, $1 \le j \le k$, $S(p_i, A_u)$ denotes the k-dimension vector of activities the actor p_i takes charge of. It describes the percentage of activity set A_u in work of actor p_i. For example, if R(Tom, fabric requiring)=0.3, R(Tom, fabric checkout) = 0.4 and R(Tom, production inspection) = 0.05, then the vector of the production director Tom executing these activities is (0.3, 0.4, 0.05). Note that if some activity has not been executed by the actor, the value of corresponding element in his vector of activity execution is 0.

Definition 7 Average execution vector of activity set A_u is defined as

$$AG(A_u)=\frac{S(p_1, A_u)+S(p_2, A_u)+\ldots+S(p_n, A_u)}{|PS(A_u)|}, p_i \in PS(A_u) \quad (2)$$

Where $PS(A_u)$ denotes the actor set who have executed at least one of the activities, which compose of activity set

A_u and $|PS(A_u)|$ denotes the number of the actors in $PS(A_u)$. The k-dimension average execution vector $AG(A_u)$ means the average of activity set A_u taken by the actor set $PS(A_u)$. It is equal to the sum of execution vector of each actor p_i $S(p_i, A_u)$ divided by $|PS(A_u)|$.

Definition 8 Difference degree of activity set A_u is defined as

$$D(A_u) = \frac{|S(p_1, A_u)-AG(A_u)|+...+|S(p_n, A_u)-AG(A_u)|}{|PS(A_u)|}, p_i \in PS(A_u) \quad (3)$$

$D(A_u)$ denotes the sum of all the difference between $S(p_i, A_u)$ and $AG(A_u)$ divided by the number of actors $|PS(A_u)|$. It describes the average difference of activity execution by the actors. Herein, we can compute the difference between vectors using Euclidean distance. Larger difference degree means more difference between actors in their work including authorities and responsibilities, while smaller difference degree means more similarity between actors. In this paper, the difference degree is used to identify process roles.

Below is an illustrative example. The matrix M subsumes times of the activities performed by the actors p_{101}, p_{105}, p_{106}, p_{107}, p_{114}, p_{115}, p_{120}, which are collected from workflow logs of the Staffware system. From the M, we can see that the total activity execution times of the actors are 16, 13, 8, 15, 20, 18 and 13 respectively. We can transform the matrix M into the matrix R denoting the percentage of the activities executed by each actor. For example, $R(p_{101}, \{a\}) = 0.625$.

$$M = \begin{array}{c} \\ p_{101} \\ p_{105} \\ p_{106} \\ p_{107} \\ p_{114} \\ p_{115} \\ p_{120} \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{pmatrix} 10 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 8 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 8 & 6 & 2 & 2 \\ 0 & 5 & 3 & 5 & 4 & 0 & 1 \\ 0 & 0 & 5 & 0 & 0 & 8 & 0 \end{pmatrix} \end{array}$$

$$R = \begin{array}{c} \\ p_{101} \\ p_{105} \\ p_{106} \\ p_{107} \\ p_{114} \\ p_{115} \\ p_{120} \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{pmatrix} 0.625 & 0.375 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.153 & 0.384 & 0.461 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0.533 & 0.466 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.4 & 0.3 & 0.1 & 0.1 \\ 0 & 0.277 & 0.166 & 0.277 & 0.222 & 0 & 0.055 \\ 0 & 0 & 0.385 & 0 & 0 & 0.615 & 0 \end{pmatrix} \end{array}$$

The matrix R shows that $S(p_{101}, \{a\}) = (0.625)$, $S(p_{107}, \{a\}) = (0.533)$, thus according to definition 7 and 8, it is obvious that $AG(\{a\}) = (0.625 + 0.533)/2 = 0.579$ and $D(\{a\}) = \frac{\sqrt{(0.625-0.579)^2} + \sqrt{(0.533-0.579)^2}}{2} = 0.046$. Similarly, we can get that $D(\{b\}) = 0.063$, $D(\{c\}) = 0.092$, $D(\{d\}) = 0.051$, $D(\{e\}) = 0.088$, $D(\{f\}) = 0.257$ and $D(\{g\}) = 0.41$. Further, consider the activity set {a, b}, there are only three actors involved, i.e. p_{101}, p_{107} and p_{115}. In order to compute the difference degree of {a, b}, we first compute $S(p_{101}, \{a, b\}) = (0.625, 0.375)$, $S(p_{107}, \{a, b\}) = (0.533, 0.466)$, $S(p_{115}, \{a, b\}) = (0, 0.277)$ and since $AG(\{a, b\}) = ((0.625 + 0.533)/3, (0.375 + 0.466 + 0.277)/3) = (0.386, 0.373)$, according to definition 8, the difference degree of {a, b} is $D(\{a, b\}) = 0.27$.

3. Role mining

Role identification is the basis of RAD modeling. To be more objective, we identify roles using workflow mining in the following discussion. It is assumed that actors who play the same role are responsible for the sub-process (activity set) whose difference degree is less than the predefined threshold T, which is correlated with activities and actors in workflow models. An alternative approach to choose T is to obtain some role identification results with different thresholds, and then to choose the more suitable one, which is the closest to the reality with the help of business experts.

The sub-process corresponding to each role must satisfy the principle of order preservation [9]. In this paper it means sub-processes should meet the direct dependence between activities.

3.1 Mining the activity set executed by each role

Algorithm 1 is proposed to discover sub-processes in which activities are executed by actors playing the same roles. The algorithm first examines activity sets, which have close relation with the current activity set and then extends the scope until the difference degree is larger than T, then a role having the responsibility of current activity set is created. Each time there exists dependence between a newly added activity set with the current activity set, they are combined into a bigger sub-process. The algorithm then marks these activities and examines those unmarked activities left. When all activities have been marked, the role identification is finished.

Since dealing with processes with loop structure is far from trivial in order-preserving role-activity diagrams, this paper only touches on workflow models without loop structure.

Algorithm 1 Mining activity set taken by each role
Input: activity set A, dependency set DEP;
Output: sub-process corresponding to each role SPA
function GenSubProcesses(A, DEP, T)
begin
  i = 1
  while $\forall SA_j$ (0<j≤i) and a∈ A, $\exists a \notin SA_j$ **do**
    $SA_i = \varnothing$;
    The remainder activity set remAS = A – {a | a∈ SA_u, 0<u≤i};
    if the number of activities in remAS is 1 **then**
      SA_i = remAS;
      return SPA = {SA_v | 0<v≤i};
    end if
    select an activity a, a∈remAS, so that D({a}) is the smallest;
    SA_i = {a};
    repeat
      temp = SA_i;
      AJ = adjAS(remAS, SA_i);
      while not all activities in AJ are selected **do**
      select an activity a in AJ;
          potAS = depCheck(SA_i∪{a}, remAS, DEP);
          if D(potAS)<T **then**
            SA_i = potAS;
            remAS = remAS - SA_i;
          end if
    end while

```
    until SA_i = temp
  i = i + 1;
  end while
  return SPA = {SA_v | 0<v≤i};

function DepCheck(SA, A, DEP)
begin
  repeat
    temp = SA;
    RL = {a | b∈SA, a∈A-SA };
    while RL ≠ ∅ do
        select an activity a in RL;
        if ∃b, c∈SA, so that ((a, b) ∈DEP and (a, c) ∈ DEP) or ((a,
        b) ∈DEP and (a, c) ∈DEP) or ((b, a) ∈DEP and (c, a) ∈ DEP)
        or ((b,a) ∈DEP and (c, a) ∈DEP)
        then
          SA = SA∪{a};
        end if
        RL = RL -{a};
        end while
  until SA = temp
  return SA;
end
```

In the algorithm 1, the function GenSubProcesses (A, DEP, T) performs the activity set A, dependence set DEP and threshold T as inputs and generates sub-processes corresponding to each role. The algorithm starts with the activity set consisting of only one activity that has the least dependence degree, and then examines its adjacent activity set according to definition 4. The adjacent activity set here is different from the activity set mentioned in [9], which has direct dependence with the current activity set. That is because sub-processes maybe make sense if they satisfy the principle of order preserving [9], while some potential sub-processes are omitted by considering activity sets, which have direct dependence relation with the current activity set. Algorithm 1 figures out the difference degree of the activity set with at least two activities and updates the adjacent activity set by recursively adding an activity until the difference degree is larger than T. The function DepCheck(SA, A, DEP) is used to check the complementary set RL of the current activity set SA to find if there exists direct dependence between activities in RL and activities in SA. The complexity of the algorithm 1 is about $O(n^3)$.

Let's take an application example. In Figure 1, assume T= 0.33, initially the activity set {a} has the least dependence degree D({a}) = 0.046, and the adjacent activity set of a is adjAS(a) = {b}. So activity b is added and D({a, b}) = 0.27, which is less than T. Then considering {c, d}, the adjacent activity set of {a, b}, D({a, b, c}) = 0.345 and D({a, b, d}) = 0.379 are both larger than T. Thus {a, b} is returned with a process role created. The remainder activity set is denoted as remAS = {c, d, e, f, g}, in which activity d has least difference degree with D({d}) = 0.051 and its adjacent activity set is {c, e, f}. Similarly, c is first added and D({d, c})=0.164, then e is added into the activity set {d, c} and we can get D({d, c, e})=0.219, which is still less than T. We use the function DepCheck to examine the activity set {d, c, e}. Since d→f but f→e, f is added to the activity set to make it satisfy the principle of order preserving. The difference degree of {d, c, e, f} is 0.32, thus {d, c, e, f} is returned with another role found. Now the remainder activity set remAS is {g}, so {g} is returned and role identification is completed while we can get the third role.

Some mined roles may be meaningless so business experts are needed to judge the validity and semantics of roles. In the example above, the role of {a, b} is a planning manager, the role of {d, c, e, f} is a technician and the role of {g} is a worker. Table 2 shows the actors and their corresponding roles. As this table shown, p_{101}, p_{107} and p_{115} are planning managers, p_{105}, p_{114}, p_{115} and p_{120} are technicians and p_{106}, p_{114} and p_{115} are workers.

Table 2 actors and their roles

actors	roles
p_{101}	planning manager
p_{105}	technician
p_{106}	worker
p_{107}	planning manager
p_{114}	technician
p_{114}	worker
p_{115}	planning manager
p_{115}	technician
p_{115}	worker
p_{120}	technician

3.2 RAD modeling

The activity set taken by each role makes up sub-processes in RAD modeling and the sub-processes can be identified by algorithm 1. Furthermore, since many measures have been proposed to describe social relations among actors [7], the interaction between roles can be determined by social network diagrams among actors. Thus in this paper, we model the interaction between roles based on social network analysis, which can be mined by the method similar to reference [7].

Definition 9 Activity dependence between process actors p_i, p_j is denoted as $rel(p_i, p_j) = \{(a, b)|i\in I, i=(a_1a_2\ldots a_b\ldots a_k), p_i, p_j\in P$, a is executed by p_i, b is executed by p_j, and $a\rightarrow b\}$. If there exists direct dependence between activity a and b, and in a workflow instance, activity a is executed by the actor p_i and activity b is executed by actor p_j, then the activity pair (a, b) shows that p_j depends on p_i. Here we deduce from the method presented in reference [7] to find such activity pairs.

Algorithm 2 analyzing the interaction between roles
Input: role set R, activity dependence pairs ADP
Output: connecting activity set of all roles CON

```
function GenCon(R, ADP)
begin
  CON = ∅;
  for each role Ri in R(0<i<|R|)
    for each role Rj in R(i<j<=|R|)
      con(Ri, Rj) = ∅;
      for each activity ap Ri is in charge
        for each activity aq Rj is in charge
          if there exists pu, pv who are assigned to Ri and
            Rj respectively such that (ap, aq)□rel(pu, pv) or
            (aq, ap)□rel(pv, pu)
          then  add ap or aq into con(Ri, Rj);
          end if
        next
      next
```

```
        CON = CON □ all nonempty con(Ri, Rj);
      next
   next
   return CON;
end
```

Algorithm 2 analyzes the interaction between roles. The function GenCon(R, REL) tries to find the interaction between roles based on the activity dependence between actors. If (a_p, a_q) is the activity pair, and p_v are assigned to R_i and R_j to perform activities a_p, a_q respectively, then a_p is the activity, which connects the role R_i with R_j. The function GenCon(R, REL) finally generates the activity set connecting related roles, which describe the interaction between roles. For example, if b→c and actor p_{115}, p_{105} perform activity b and activity c respectively in the workflow instance i_3, it is obvious that (b, c) ∈ rel(p_{115}, p_{105}). Since p_{115} plays the role "planning manager" ,which is responsible for activity b, and p_{105} plays the role "technician" to take charge of activity c, it seems that activity b ∈ con(planning manager, technician). Algorithm 2 finds that con(planning manager, technician) = {b } and con(technician, worker) = {c, e}. Figure 2 shows RAD modeling of the clothing production process. There are three blue rectangles, which represents roles and the sub-processes in the rectangles correspond to the roles while the arrows between the rectangles describe the interaction between roles.

We have done some experiments on many other workflow logs besides the clothing production logs using the proposed method. Table 3 shows the result of a role-mining example. By comparison, it seems that we can get better role identification results when T is set between 0.3 and 0.4. It needs further probative work.

Table 3 an example

roles	difference degree	activities	actors
0	0.270	a, b	p_{101}, p_{107}, p_{115}
1	0.304	d, e, f	p_{105}, p_{114},p_{115},p_{120}
2	0.092	e	p_{105}, p_{114},p_{115}, p_{120}
3	0.410	g	p_{106}, p_{114}, p_{115}

4. Conclusions

How to identify roles and their interactions is necessary for RAD modeling but most of methods for addressing the issue seem to be subjective. Roles are often identified by business experts without definite criteria. In this paper, we use workflow mining to discover process roles based on the assumption that actors who play the same role are responsible for the sub-process whose difference degree is less than the predefined threshold, and then we analyze their interactions with the aid of social network analysis. Thus RAD modeling is finished. The experimental results of applying our RAD modeling method based on workflow mining to some workflow logs seem to be satisfactory except that the threshold T is not easy to choose. In the future, more research should be done to mine workflow models with the loop-structure.

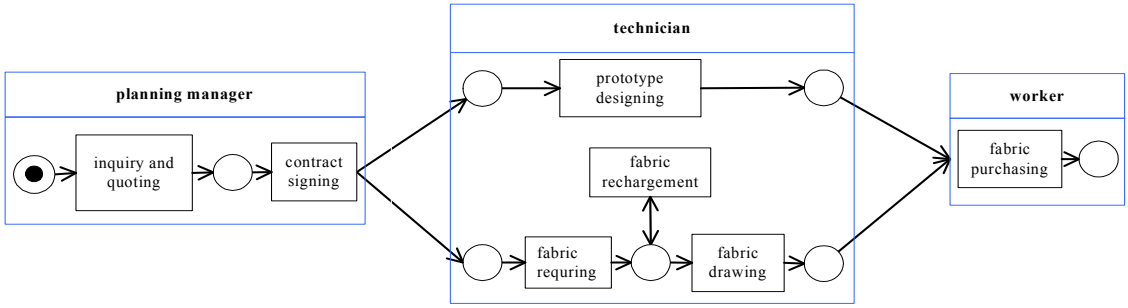

Figure 2 mined RAD model

Acknowledgement

This paper is supported by the National High-Tech. R&D Program for CIMS, China（2008AA04Z127）.

References

[1] Zhao Weidong ， Huang Lihua(2004). Role-based multi-agent workflow systems. Chinese Journal of Management Science, vol7, no2, pp 55-62(in Chinese)

[2] Mentzas G, Christos, Kavadias S(2001). Modeling business process with workflow systems: an evaluation of alternative approaches. International Journal of Information Management,vol 21,no.2 ,pp123-135

[3] Phalp K T, Henderson P, Walters R J, et al(1998). RolEnact: role-based enactable models of business processes. Information and Software Technology, vol40,no.3, pp123-133

[4] Ould M. Business Processes: Modeling and analysis for reengineering and improvement, New York：John Wiley & Sons, 1995

[5] Wil M.P V, Weijters A J(2003). Process mining: a research agenda, Computers in Industry, vol.53, no.3, pp231-244

[6] Wil M.P V, Song M(1998). Discovering models of software processes from event-based data], ACM Transactions on Software Engineering and Methodology, vol.7,no.3, pp215-249

[7] Wil M.P V, Song M(2004). Mining social networks: Uncovering interaction patterns in business processes, Business Process Management,pp244-260

[8] Schlegelmich J, Steffens U(2005). Role mining with ORCA. Proceedings of the tenth ACM symposium on Access control models and technologies. Stockholm, Sweden: SACMAT，pp168-176

[9] Liu D R, Shen M(2003). Workflow modeling for virtual processes: An order-preserving process-view approach. Information Systems, vol.28,no.6,pp505-532