基于Petri网的数据库日志分析方法研究

景 波 刘 莹 陈 耿
(南京审计学院信息科学学院 南京 210029)

摘 要 为提高对复杂ERP系统的审计效率,提出了通过数据库日志快速发现系统中存在的不合规业务流程的方法。该方法利用数据库中已有的结构关系来确定日志操作中工作流程的次序,并通过Petri网中的a-算法对数据库日志进行转换,将转换后的真实业务流与合规业务流进行差异比较,使审计人员能较直观和准确地判断出系统中存在的异常业务流程。

关键词 数据库日志,流程挖掘,Petri网
中图法分类号 TP301.6 **文献标识码** A

Research on Database Log Based on Petri Nets

JING Bo LIU Ying CHEN Geng
(School of Information Science, Nanjing Audit University, Nanjing 210029, China)

Abstract In order to improve audit efficiency in complex ERP systems, the method of quickly discovering non-compliance of business processes existing in the system through the database log was proposed. The method utilizes the structural relationship already existing in the database to determine the order of the log operation workflow and convert the database log by a-algorithm in Petri nets, and compares the differences between converted real business flow and compliance business flow, so that auditors can more intuitivey and accurately judge the anomalous business processes in the system.

Keywords Database log, Process mining, Petri net

1 引言

随着法国兴业银行雇员凯维埃尔通过系统漏洞进行未经授权的交易导致该行损失49亿欧元的案情曝光,内部流程控制漏洞的审计力度也日益加大。海量的被审数据和有限的审计资源使得审计人员很难在短时间内准确发现不合规的业务流程。业务流程合规性检查主要是针对用户是否按原始系统设计的标准流程进行ERP系统操作的流程检查。近年来数据库日志文件分析成为审计中流程检查的主要手段,但常规方式下获得的数据库系统日志文件只能得到用户操作ERP系统时的一般行为(例如登入、注销、用户计算机名称、IP等),无法完整再现用户操作ERP系统的流程。为此,本文提出利用数据库系统表中已有的数据间结构关系来简化日志中数据操作先后顺序的确定,并利用Petri网建模[1,2]的a-算法[3]将日志中的操作流程进行图形化转换,便于审计人员通过与合规业务流的差异分析来判断日志中的操作流程是否为异常业务流。

2 基本概念

Petri网是由Carl Adam Petri在1962年提出的一种描述并行及并发操作的工具,是一种具有数学性质与图形特性的系统建模工具,可以用来简化多种系统的构建与分析。企业在复杂的业务处理流程中,流程选择具有对实际业务自身特点的依赖性,故可以将工作流程映射为Petri网模型[4]。此外,企业运营中包含诸多业务实例的业务流程的处理在工作流模型中均有一种业务处理实例过程的逻辑事件图与其相对应[5]。本文则利用数据库日志记录的次序关系生成相应的逻辑事件图,通过与合规流程图比对来发现违规流程。

定义1[1] 一个三元组$N=(S,T;F)$是一个Petri网,当且仅当:

(1)$S\cup T\neq\varphi$;

(2)$S\cap T=\varphi$;

(3)$F\subseteq(S\times T)\cup(T\times S)$;

(4)$dom(F)\cup code(F)=S\cup T$。

其中,S为库所,T为变迁,F是N的弧,表示库所与变迁之间的流关系,并且

$dom(F)=\{x\in S\cup T\mid \exists y\in S\cup T:(x,y)\in F\}$

$code(F)=\{x\in S\cup T\mid \exists y\in S\cup T:(y,x)\in F\}$

定义2[1] 设$\Sigma=(S,T;F,M)$是一个Petri网系统,M是Σ的基网$PN=(S,T;F)$上的一个标识,$t\in T$,则t在M有

到稿日期:2013-08-16 返修日期:2013-10-18 本文受国家自然科学基金(70971067,71271117),江苏省公共工程审计重点实验室开放课题(20201201213),江苏省审计信息工程重点实验室开放课题(AIE201205)资助。

景 波(1975—),男,硕士,副教授,主要研究方向为IT审计、数据挖掘,E-mail:jbo@nau.edu.cn;刘 莹(1977—),女,硕士,讲师,主要研究方向为数据挖掘、分布式计算技术;陈 耿(1965—),男,博士,教授,主要研究方向为数据挖掘、审计信息化、知识工程等。

发生权的条件是：

$\forall s\in {}^{*}t, M(s)\geqslant t^{*}, K(s)\geqslant M(s)+W(t,s)$

记作 $M[t>$。

定义 3[1] 设 T 是任务的集合，$\sigma\in T^{*}$ 是一个流程路径，$W\in P(T^{*})$ 是流程日志。其中 $P(T^{*})$ 是 T^{*} 的幂集，即 $W\subseteq T^{*}$。

定义 4[1] Petri 网系统的变迁发生规则满足：

设 $\sum=(S,T;F,M)$ 是一个 Petri 网系统，M 是 $\sum$ 的基网 $PN=(S,T;F)$ 上的一个标识，$t\in T$。若 $M[t>$，则在标识 M 下，变迁 t 可以发生，从标识 M 发生变迁 t 得到一个新的标识 M'（记为 $M[t>M'$），对 $\forall s\in S$：

$$M'(s)=\begin{cases}M(s)-W(s,t) & \text{若 } s\in {}^{*}t-t^{*}\\ M(s)+W(t,s) & \text{若 } s\in t^{*}-{}^{*}t\\ M(s)-W(s,t)+W(t,s) & \text{若 } s\in t^{*}\cap {}^{*}t\\ M(s) & \text{若 } s\in t^{*}\cup {}^{*}t\end{cases}$$

定义 5[1] 设 W 是任务集 T 上的工作流日志，即 $W\in P(T^{*})$，设 $a,b\in T$，则存在如下次序关系：

1. $a>_{w}b$，当且仅当 $\exists\sigma=t_1t_2\cdots t_{n-1}, i\in\{1,2\cdots n-2\}$，使得 $\sigma\in W$ 且 $t_i=a$ 且 $t_{i+1}=b$；

2. $a\rightarrow_{w}b$，当且仅当 $a>_{w}b$ 且 $b\leqslant_{w}a$；

3. $a\#_{w}b$，当且仅当 $a\leqslant_{w}b$ 且 $b\leqslant_{w}a$；

4. $a\parallel_{w}b$，当且仅当 $a>_{w}b$ 且 $b>_{w}a$。

3 基于 Petri 网的转换过程

3.1 日志数据的预处理

首先，通过 ApexSQL Log 数据库工具获取 ERP 系统的数据库操作日志；但是由于系统中存在多种会造成系统产生异常导致应用实例中断并伴随产生不完整的日志文件的因素，因此需要对日志进行预处理。通过将顺序化的流程日志进行形式化转换，然后进行异常日志过滤处理，除去冗余信息和噪音的数据。过滤规则如下：设 $N=(P,T,F)$ 是一个合规的工作流网，$W=(T_w,O_w,t_{wi},t_{wo})$ 是 N 的执行日志。

规则 1 $\forall\sigma\in WL$，则以 t_i 为起始节点，以 t_o 为结束节点；

规则 2 $\forall t\in T_w$，则 $\exists\sigma\in WL$，且 $t\in\sigma$；

规则 3 $\exists W'$ 是 N 的执行日志，$\forall\sigma\in W'$，则 $\sigma\in W$。

规则 1 保证日志中的工作流执行的踪迹是完整的，σ 以 t_i 为起始节点，以 t_o 为结束节点；规则 2 过滤掉不存在于 T_w 的噪音节点；规则 3 去除了重复的工作日志。处理后的日志中包含 *LogID*, *BeginTime*, *Operation*, *Table*, *EndTime* 等信息，*LogID* 为每笔操作的次序，*BeginTime* 和 *EndTime* 分别为该笔操作的开始时间和结束时间，*Operation* 代表进行何种操作行为，*Table* 为对应操作的数据库表。

以 ERP 系统的采购流程为例，其包括标准的采购流程、先采购后付款的流程、紧急采购即无申购直接采购的流程。合规流程的子流程包含（申购，采购），（采购，报账），（申购，询价），（询价，议价），（议价，采购）等环节。假设审计中获取的被审数据库操作日志如表 1 所列。

可以通过 *sys. indexes* 和 *sys. index_columns* 这两张表实现主键的自动获取。*sys. indexes* 表中的 *is_primary_key* 字段表示是否为主键的信息，但还要通过表 *id* 和表 *sys. index_columns* 关联来找到具体字段。即先通过 *sys. index_columns* 的 *object_id* 和 *column_id* 筛选出具体的字段，然后通过 *ic. index_id* 到表 *sys. indexes* 中找到相应的记录来确定是否为主键，同时通过匹配名字是否以 *PK_* 开头来过滤其他索引。实现的 SQL 语句如下：

```
Select col. name as ColumnName,
       col. max_length as DataLength,
       col. is_nullable as IsNullable,
       t. name as DataType,
       ep. value as Description,
       (select top 1 ind. is_primary_key
       from sys. index_columns ic
              left join sys. indexes ind
                 on ic. object_id=ind. object_id
              and ic. index_id=ind. index_id
          and ind. name like 'PK_%'
           where ic. object_id=obj. object_id
       and ic. column_id=col. column_id
       ) as IsPrimaryKey
from sys. objects obj
       inner join sys. columns col
       on obj. object_id=col. object_id
       left join sys. types t
       on t. user_type_id=col. user_type_id
       left join sys. extended_properties ep
       on ep. major_id=obj. object_id
       and ep. minor_id=col. column_id
       and ep. name='MS_Description'
where obj. name=@TableName
```

表 1 ERP 系统的数据库操作日志

1	12-08-16 11:21:01 Insert T1 12-08-16 11:23:50
2	12-08-16 11:23:05 Insert T2 12-08-16 11:25:40
3	10-06-01 11:24:27 Insert T1 12-08-16 11:25:51
4	12-08-16 11:25:11 Update T1 12-08-16 11:26:13
5	12-08-16 11:25:42 Update T1 12-08-16 11:27:54
6	12-08-16 11:26:57 Insert T3 12-08-16 11:28:29
7	12-08-16 11:27:33 Update T2 12-08-16 11:31:15
8	12-08-16 11:29:51 Insert T3 12-08-16 11:33:29
9	12-08-16 11:31:24 Insert T3 12-08-16 11:34:47
10	12-08-16 11:31:26 Insert T4 12-08-16 11:33:19
11	12-08-16 11:32:37 Update T3 12-08-16 11:35:39
12	12-08-16 11:33:39 Insert T4 12-08-16 11:35:29
13	12-08-16 11:34:28 Update T3 12-08-16 11:36:19
14	12-08-16 11:34:39 Insert T4 12-08-16 11:36:21
15	12-08-16 11:35:28 Update T4 12-08-16 11:36:19
16	12-08-16 11:36:33 Update T4 12-08-16 11:37:29
17	12-08-16 11:36:48 Update T4 12-08-16 11:38:19

数据库中的外键关系可以利用 SQL Server 中的系统表 *INFORMATION_SCHEMA. KEY_COLUMN_USAGE*、*sysforeignkeys* 来获得。通过系统表取得 4 张表的主键、外键关系为 $T1(PK(Key1))$、$T2(PK(Key2), FK(Key1))$、$T3(PK(Key3), FK(Key2))$、$T4(PK(Key4), FK(Key3))$。

利用获取到的主键和外键关系可得到如图 1 所示的表间关系。

主、外键的表间关系可以分为两种情形：

(1)只有主键而无外键，如表 1 中第 1 行操作日志涉及表 $T1$，而 $T1$ 只有主键值 $T1(Key1)$。

(2)含有主键和外键，如第 2 行操作日志涉及表 $T2$，由于表 $T2$ 同时具备主键和外键，因此该日志记录的操作因

T2(PK(Key2),FK(Key1))而会影响 T1,T2 这两张表。

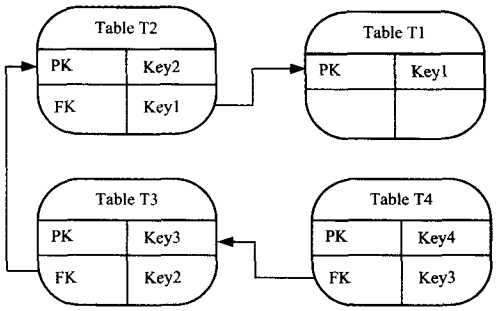

图 1　主键外键的表间关系

利用这两种主、外键关系对表 1 进行形式化扩充,得到表 2。

表 2　结合主、外键关系的日志文件

Log ID	Begin Time	Operation	Table	End Time	Table(PK value)	Table(FK value)
1	11:21:01	Insert	T1	11:23:50	T1(Key1)	
2	11:23:05	Insert	T2	11:25:40	T2(Key2)	T1(Key1)
3	11:24:27	Insert	T1	11:25:51	T1(Key1)	
4	11:25:11	Update	T1	11:26:13	T1(Key1)	
5	11:25:42	Update	T1	11:27:54	T1(Key1)	
6	11:26:57	Insert	T3	11:28:29	T3(Key3)	T2(Key2)
7	11:27:33	Update	T2	11:31:15	T2(Key2)	T1(Key1)
8	11:29:51	Insert	T3	11:33:29	T3(Key3)	T2(Key2)
9	11:31:24	Insert	T3	11:34:47	T3(Key3)	T2(Key2)
10	11:31:26	Insert	T4	11:33:19	T4(Key4)	T3(Key3)
11	11:32:37	Update	T3	11:35:39	T3(Key3)	T2(Key2)
12	11:33:39	Insert	T4	11:35:29	T2(Key2)	T1(Key1)
13	11:34:28	Update	T4	11:36:19	T4(Key4)	T3(Key3)
14	11:34:39	Update	T3	11:36:21	T3(Key3)	T2(Key2)
15	11:35:28	Insert	T4	11:36:19	T4(Key4)	T3(Key3)
16	11:36:33	Update	T4	11:37:29	T4(Key4)	T3(Key3)
17	11:36:48	Update	T4	11:38:19	T4(Key4)	T3(Key3)

3.2　a-算法分析

将数据库的每项行为定义为一个事件,再利用 a-算法生成基于 Petri 网的工作流网(P_w,T_w,F_w)。为便于描述,将表 1 中的数据操作 *Insert* T1、*Update* T1、*Insert* T2、*Update* T2、*Insert* T3、*Update* T3、*Insert* T4、*Update* T4 依次记为 A、B、C、D、E、F、G、H;并设 T_w 为日志中所有操作的变迁集,T_i 为日志中所有操作的初始变迁集,T_o 为结束变迁集,X_w 为包含秩序关系的变迁集,Y_w 为在 X_w 中去除重复变迁集,P_w 代表在 Y_w 中,将各个变迁加入库所并加入输入库所与输出库所,F_w 为以箭头连接起来的库所与变迁。

a-算法转换过程如下:

步骤 1　$T_w=\{t\in T\mid \exists_{\sigma\in W^t}\in\sigma\}$

首先将所有事件归纳为一个集合,t 代表所有可能发生的事件,因此 t 的所有集合 T 为 $T_W=\{A,B,C,D,E,F,G,H\}$。

步骤 2　$T_i=\{t\in T\mid \exists_{\sigma\in W^t}=first(\sigma)\}$

确认起始事件 T_i,由于表间存在主、外键的关联关系,因此,数据操作时必须要先满足这种关系,一般而言,主、外键关系可以分为 4 种类型:

(1)在同表下,*Table* 1_(*PK*) →*Table* 1_(*PK*)

如 *Insert* T1 和 *Update* T1,都在同一张表 T1 上,并且其 *PK* 都是 T1,此种关联方式类似进行更新操作,必须完成新增操作后,再进行更新。

(2)在不同的表下,*Table*_1(*PK*)→*Table*_2(*FK*)

表 1 的主键为表 2 的外键,如 *Update* T1_(*PK*)→*Insert* T2_(*FK*)。

(3)在不同的表下,*Table* 1_*PK*→*Table* 2_*PK*1(*FK*)

如在确立发票主表与发票明细表之间的关联时,若主键字段相同,则可以通过发票明细表的其它项目来判别两表间的关联,确定主键和外键。

(4)在不同的表下,*Table* 1_*PK*→*Table* 2_*PK*(*FK*)

如公司员工表,经理的身份同时为经理及员工,此关联方式本身就是主键与外键。

以表 2 为例,编号为 1 时的 *Insert* T1,不需要任何的外键,它的主键为它本身;而编号为 4 的 *Update* T1,关联方式为上面描述的第一种方式(*Insert* T1→*Update* T1);而第 2 行的 *Insert* T2,它的外键则为 T1 的主键(*Update* T1→*Insert* T2),因此为第二种关联方式,藉由这两种关联方式,可以将此示例中的关联都串连起来,根据示例,*Insert* T1 为起始事件,因此可以确定某个流程的起始事件会为事件 A;此外,编号为 6 时的 *Insert* T3,它的外键必为 T2 的主键,而编号为 1—5时并没有 *Update* T2 这个行为,因此可以知道此时的 *Insert* T3 没有与其它的流程相关联,所以可以判定它为起始流程;故在步骤 2 的起始事件 T_i 里,$T_i=\{A,E\}$。

步骤 3　$T_o=\{t\in T\mid \exists_{\sigma\in W^t}=last(\sigma)\}$确认最后一个事件,由步骤 2 的描述可以确认,示例中最后的事件为 *Update* T4(事件 H),因此 $T_o=\{H\}$。

步骤 4

$$X_W=\{(A,B)\mid A\subseteq T_W\wedge B\subseteq T_W\wedge\forall_{a\in A}\forall_{b\in B^a\to W^b}\wedge\forall_{a_1,a_2\in A^{a_1}\neq W^{a_2}}\wedge\forall_{b_1,b_2\in B^{b_1}\neq W^{b_2}}\}$$

将集合中相关联的事件取出,根据步骤 2 的描述,如 A(*Insert* T1)与 B(*Update* T1)之间有关联产生,因此(A,B)有关联性;B(*Update* T1)与事件 C(*Insert* T2)同理,会产生(B,C)的关联。因此,利用不同关联方式可以将流程提取出来,所有的关联关系表示为:

{(A,B),(B,C),(C,D),(D,E),(E,F),(F,G),(G,H),(E,G),(G,H),(A,B),(B,E),(E,F),(F,G),(G,H)}

把相同关系事件的集合去除之后会得到下列集合:{(A,B),(B,C),(C,D),(D,E),(E,F),(F,G),(G,H),(E,G),(B,E)},进而得到 3 条流程:(A,B,C,D,E,F,G,H)、(E,G,H)、(A,B,E,F,G,H),依次用集合 X_{W_1}、X_{W_2}、X_{W_3} 来表示。

步骤 5　$P_W=\{p_{(A,B)}\mid (A,B)\in Y_W\}\cup\{i_W,o_W\}$

P_W 集合表示为 $P_W=\{1,2,3,4,5,6,7,8,9\}$,编号只表示其放置的位置与事件的相对位置。要加入库所至变迁,通常会有 3 种加入的方式:

(1)*Input Place* 方式:在起始事件前加入库所,如 X_{W_2} 则在起始事件 E 之前加入一个库所。

(2)*Output Place* 方式:在结束事件后加入库所,如 X_{W_2} 则在结束事件 H 之后加入一个库所。

(3)*Insert Place* 方式:在流程里具有关联关系的事件里加入库所,如 X_{W_2} 则在 E 和 G 之间、G 和 H 之间分别加入库所。按上述方式加入库所后的下列集合:

$PX_{W_1}=\{1(A2B),(B3C),(C4D),(D5E),(E6F),(F7G),(G8H)9\}$

$PX_{W_2}=\{1(E2G),(G3H)4\}$

$PX_{W_3}=\{1(A2B),(B3E),(E4F),(F5G),(G6H)7\}$

数字表示库所与变迁之间的相对位置。步骤 6 将流程中加入有向连接弧(*arc*)。

步骤 6

$F_w=\{(a,p_{(A,B)})\mid(A,B)\in Y_W\wedge a\in A\}\cup\{(p_{(A,B)},b)\mid(A,B)\in Y_W\wedge b\in B\}\cup\{(i_w,t)\mid t\in T_i\}\cup\{(t,o_w)\mid t\in T_o\}$

将库所与变迁加入有向连接弧，用来表示使用者行为的流程方向。

$F_{W_1}=\{(1,A),(A,2),(2,B),(B,3),(3,C),(C,4),(4,D),(D,5),(5,E),(E,6),(6,F),(F,7),(7,G),(G,8),(8,H),(H,9)\}$

$F_{W_2}=\{(1,E),(E,2),(2,G),(G,3),(3,H),(H,4)\}$

$F_{W_3}=\{(1,A),(A,2),(2,B),(B,3),(3,E),(E,4),(4,F),(F,5),(5,G),(G,6),(6,H),(H,7)\}$

根据步骤 5 的描述，将得到的 3 个流程图合并绘制成一张图形，如图 2 所示。

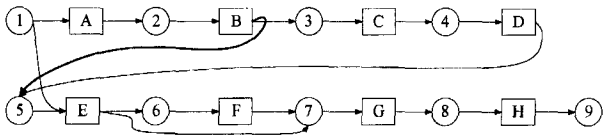

图 2　转换后的 3 条流程

对应原设计合规流程是将事件 A－H 执行一遍，并包含一些常见的特殊流程(例如，紧急采购)，故合规流程的事件集合为：

$F_{W_{原设计}}=\{(A,B),(B,C),(B,E),(C,D),(D,E),(E,F),(F,G),(G,H)\}$

将 X_W 与 $F_{W_{原设计}}$ 进行差运算后，若结果为空，则表示该流程被包含在规范流程里，否则为不规范流程。X_{W_1} 差运算结果为空集，该流程与规范流程一致；X_{W_2} 差运算结果为{(E,G)}，该流程为异常流程；X_{W_3} 差运算结果为空集，该流程虽形式上与规范流程不一致，但属于规范流程。

3.3　图形化显示

将得到的流程转换为 CPN TOOLS 软件可以识别的 XML 格式，进行 Petri 网的图形绘制。转换 XML 的格式标签含义如表 3 所列。将转换后 XML 文档用 CPN TOOLS 软件打开即可生成流程图，如图 3 所示。

表 3　XML 中主要格式标签的含义

事件	Pos attr	坐标
	Text	名称
	Box	文本框坐标
	Fillattr	事件颜色、位置及栏位属性
	Pattern	
	Lineattr	
	Thick	
	Type	
	Textattr	
库所	Pos attr	坐标
	Text	名称
	Ellipse	文本框坐标
	token	坐标
	Marking	状态的坐标
	Type	形态
	intmark	起始状态
有向弧	Pos attr	坐标
	Transcend	连接的事件
	Placeend	连接的库所
	annot	文字及坐标
	orientation	连接方向

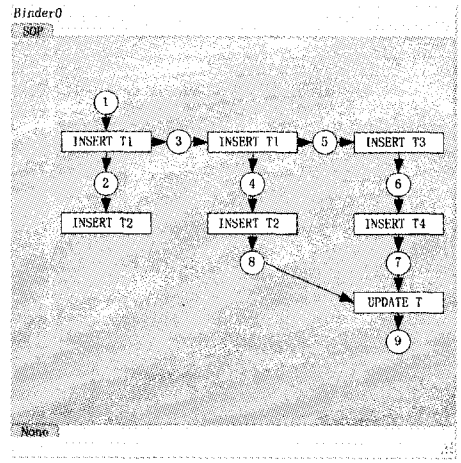

图 3　由 CPN TOOLS 生成的流程图

结束语　针对审计活动中的业务流程分析，本文提出的基于 Petri 网的合规流程分析方法能够建立快速有效的业务流程分析模型，实现业务流程自动比对和不合规业务流程的自动发现，该方法具有普遍适用性。本文通过 ERP 系统在数据库中的表间关系来简化 Petri 网模型，有利于分析效率的提高，未来的工作是进一步改进算法以降低其时间复杂度，并考虑如何快速转换存在多种不同形态的非规则结构的日志数据。

参 考 文 献

[1] 吴哲辉. Petri 网导论[M]. 北京：机械工业出版社，2006

[2] 刘培顺，何大可. Petri 网的分享合成操作[J]. 系统仿真学报，2006，18(11)：3313-3319

[3] van der Aalst W，Weijters T，Maruster L. Workflow Mining：Discovering Process Models from Event Logs[J]. IEEE Transactions on Knowledge and Data Engineering，2004，16(9)：452-460

[4] Reijers H A. Design and control of workflow processes[M]. Berlin：Springer-Verlag，2003：32-59

[5] Girault C，Valk R. Petri nets for systems engineering：a guide to modeling，verification，and applications[M]. Berlin：Springer-Verlag，2003：473-566

[6] van der Aalst W M P，Weijters A J M M，Maruster L. Workflow Mining：Discovering Process Models from Event Logs[J]. IEEE Transactions on Knowledge and Data Engineering，2004，16(9)：1128-1142

[7] Aybar A，Iftar A. Deadlock avoidance controller design for timed petri nets using stretching [J]. Systems Journal，2008，2(2)：178-188

[8] 万欣，刘强. 工作流平台中动态流程模型的研究[J]. 计算机应用研究，2006，23(9)：69-71

[9] 张佩云，黄波，孙亚民. 基于 Petri 网的 Web 服务组合模型描述和验证[J]. 系统仿真学报，2007，19(12)：2872-2876

[10] 钱柱中，陆桑璐，谢立. 基于 Petri 网的 Web 服务自动组合研究[J]. 计算机学报，2006，29(7)：1057-1066

[11] Xiong Peng-cheng，Fan Yu-shun，Zhou Meng-chu. A Petri Net Approach to Analysis and Composition of Web Services [J]. IEEE Transactions on Systems Man and Cybernetics Part A-Systems and Humans，2010，40(2)：376-387

[12] Wen L J，van der Aalst W M P，Wang J M，et al. Mining process models with non-free-choice constructs[J]. Data Mining and Knowledge Discovery，2007，15：145-180