doi:10.3969/j.issn.1673-9833.2013.06.010

网络化软件行为过程建模方法研究

谢保良，张文佳，蒋　云，马迎辉，彭　成，满君丰，潘伟强

（湖南工业大学 计算机与通信学院，湖南 株洲 412007）

摘　要：提出了一种基于过程挖掘的行为过程模型的网络化软件行为分析方法和研究途径。该方法采用改进的通用行为过程模型，利用过程挖掘的思想和方法，对监控收集的日志文档进行行为挖掘及petri网建模，在此基础上，对行为进行合理性分析与验证，以确保所建立模型的完备性和通用性；然后，以一个开放网络环境下的应用案例为例，分析验证了系统的复杂交互行为，通过对系统进行监控，可有效预防各种恶意行为，提高系统的安全性和可靠性。

关键词：网络化软件；交互行为；行为模型；过程挖掘

中图分类号：TP311.56　　**文献标志码**：A　　**文章编号**：1673-9833(2013)06-0046-06

Research of Modeling Method for Networked Software Behavior Process

Xie Baoliang, Zhang Wenjia, Jiang Yun, Ma Yinghui, Peng Cheng, Man Junfeng, Pan Weiqiang

（School of Computer and Communication, Hunan University of Technology, Zhuzhou Hunan 412007, China）

Abstract: Presented an analysis method of networked software behavior and research approach based on process mining of behavior process model. The method uses the improved general behavior process model and adopts the concepts and method of process mining to perform the behavior mining and Petri net modeling on the monitoring and collecting log files, applied and verified the behavior reasonability to ensure the built model completeness and versatility; and then applied an open network environment applicable case to analyze and verify the complex interaction of system. Through the monitoring of system, it is effective to prevent malicious behaviors and improve the safety and reliability of system.

Keywords: networked software; interactive behavior; behavior modeling; process mining

1　网络化软件及其行为建模

随着计算机技术和互联网技术的不断发展，软件的发展呈现出明显的网络化趋势。网络化软件已成为软件技术发展的主导模式和软件系统的主要形态[1-3]。网络化软件是一类以因特网（internet）为媒介，以网上信息/服务资源（service resource）为元素，以元素之间的协同和互操作为构造手段，其拓扑结构和行为可动态演变的软件密集型混合系统[3-4]。目前，网络化软件已成为因特网上的应用主流，正在推进整个软件行业及信息产业的快速发展。

软件行为是软件运行时的各种表现形态及状态演变的过程，而网络化软件的行为模型是根据某一层次的行为信息构建的行为状态序列及变迁，以表征软件的行为特征，并可用于行为的各种异常检测[5]。软件行为模型是软件领域研究的热点，比较典型的

收稿日期：2013-10-12

基金项目：国家自然科学基金资助项目（61350011，41362015），湖南省自然科学基金资助项目（12JJ2036，14JJ2115），湖南工业大学研究生创新基金资助项目（CX1308）

作者简介：谢保良（1986-），男，湖南双峰人，湖南工业大学硕士生，主要研究方向为可信软件，E-mail: 552131291@qq.com

动态行为模型有基于对象的软件行为（software behavior based on system objects，SBO）模型[5]、交互行为动态建模（interactive behavioral dynamic modeling，IBDM）模型[6]和有限状态自动机（finite state automaton，FSA）模型[7]，混合模型有混杂有限自动机（hybrid finite automaton，HFA）模型[8]和 Dyck 模型[9]等。软件行为建模有静态建模、动态建模和混合建模。静态建模采用直接分析软件（程序）的源代码或二进制代码的方式，提取系统调用等相关行为信息，建立行为模型。动态建模采用动态训练的方式，监控并记录不同情况下的函数、系统调用等的行为信息，形成行为模型[6,10-11]。目前，软件行为建模的发展趋势是混合建模，即综合考虑系统调用时的控制流信息和数据流信息等，以开发出更精确、更完备的软件行为模型。网络化软件的行为过程是一个动态变化的过程[12]，因此，采用动态建模的方式来描述其行为是符合实际的。

过程模型的本质是描述并发系统中可观察事件间的偏序关系，而对日志文档进行挖掘的主要工作体现在对其反映的软件行为的异常性、交互性及性能分析等方面。如 M. L. Jacchero 等人[13]提出了一种新的面向对象的建模语言，用来支持系统软件过程的提取。M. T. Baldassarre 等人[14]基于过程模式视角，研究了如何支持过程模型的定制，以适应过程运行时所面对的具体环境。对于软件过程建模研究的目的，Li Mingshu 等人[15]将之概括为：文档化（documentation）、执行（execution）、分析（analysis）和演化（evolution）。对于模型推断方面的研究，研究者们广泛使用 K-tail 算法[16]，这种算法具有在无开发人员监督的情况下，也可推断小型或简易软件系统模型的优势。此外，相关研究者通过编写规约条件来推断系统模型，如 Zhai Jian 等人[17]使用扩展的 TRISO/ML 语言图形方式，来描述和刻画软件过程组件的内部行为，它通过标准的软件过程组件和组件化建模操作，可最大限度地避免软件过程建模中的主观因素，建立的软件过程模型可被严格、准确地记录和客观重复回溯，但它仍需进一步建立组件间的关系和约束等限制条件，以规范建模过程，从而达到对可信软件过程建模的目的。

基于以上研究成果，本文采用过程挖掘的方式，对网络化软件的行为过程进行描述与分析，使用的建模工具为 Prom6.2。首先，对监控收集到的日志文档的软件行为进行 petri 网建模，在此基础上，再进行行为的合理性分析与验证，以确保所建模型的完备性和通用性。

2　行为过程模型的建立

2.1　相关定义

定义 1　事件为系统（程序）函数序列的集合。设定事件为一多元组 $e=(m, n, s, q, v)$，其中，$m\in F$（函数的有限集），$n\in D_{\mathrm{p}}$（参数值域），$s\in D_{\mathrm{v}}$（变量值域），$q\in T$（时间戳的集合），$v\in C$（系统所有类的集合）是事件类型的子集。

定义 2　事件类型 E 指具有相同行为属性（参数）的事件的集合。

定义 3　事件实例指完成某一具体操作的活动，是过程实例的元素。

定义 4　过程实例 $P_i=\left(e_{i_1}, e_{i_2}, e_{i_3}, \cdots, e_{i_n}\right)$，$i\in \mathbf{N}$ 为完成某操作所包括的事件实例序列。

定义 5　偏序关系。首先定义 $\alpha\in\beta\times\gamma$ 为事件实例偏序关系集合，如果〈a, b〉$\in\alpha$, $a\neq b$，事件实例 a 和事件实例 b 相邻，且事件实例 a 位于事件实例 b 之前，称两事件实例具有偏序关系（偏序符号为$\succ$），记为 $a\succ b$。

定义 6　事件实例状态 S 即系统运行时程序（函数）执行调用的函数序列，也即事件的集合，包括初始状态和终止状态。S_{c} 为初始状态的集合，S_{f} 为终止状态的集合。

定义 7　相对发生率 rel 和绝对发生率 abs。相对发生率 rel 为行为模型中所有的过程实例所包含的发生的事件实例所占百分比，绝对发生率 abs 为行为模型中所有的过程实例所包含的发生的事件实例数量。

定义 8　通用行为过程模型（the general behavior model process，GBMP）为一个 5-*tuple*（$S, E, T, S_{\mathrm{c}}, S_{\mathrm{f}}$）。其中，$S$ 为模型中所有状态的集合，数学描述为 $S=\{m|m\in F$，m 为程序执行时调用的函数$\}$；E 为模型中所有事件类型的集合，数学描述为 $E=\{e|e$ 具有相同的属性或参数$\}$；T 为模型中所有事务（变迁）的集合，即所有事件实例的集合；$S_{\mathrm{c}}\subseteq S$ 为模型中初始状态的集合；$S_{\mathrm{f}}\subseteq S$ 为模型中终止状态的集合。

2.2　构建 GBMP 模型算法

采用动态训练的方式来构建 GBMP 模型。

1）将所提取的日志文档，依据下述正则表达式 R：(?<timestamp>).+(?<TYPE>.+)，将其进行分割，得到过程实例；

2）在程序的执行过程中捕获系统调用，分析系统调用对象，再将系统调用路径转化为对象的行为路径；

3）通过系统调用，将同一系统内的软件行为对

象即事件实例互联起来；

4）根据所得对象的行为路径，建立所需的GBMP模型。

GBMP模型构建算法描述如下。

输入：使用正则 R 进行分割日志文档 L 所得到的事件实例序列。

输出：GBMP模型。

具体步骤如下：

1）将模型的起始状态 S_c 进行初始化；

2）将 S_c 加入模型中，使初始化完成之后模型中仅有 S_c 状态；

3）将事件实例序列进行多次训练，其中，每个事件实例进行单次训练。

4）每个单次训练完成后，将当前的事件实例状态 S'_c 初始化为模型新的 S_c 状态；

5）根据当前状态和新的训练生成的事件实例，继续产生下一个新的状态；

6）判断新的状态是否存在于当前模型中，若不存在，则将其加入模型中；

7）循环上述整个过程，直至穷尽所有产生的事件实例。

2.3 建立行为过程模型图

对建立的在线系统运行时所产生的交互日志文档进行收集，采用日志生成工具XESame，对收集到的日志文档转换成后缀为.XES格式，转换后的部分文档截图如图1所示。

在Windows XP操作系统下，采用建模工具Prom6.2，对这种按时间戳有序排列的特殊日志文档采用基于alpha的petri网进行过程挖掘，建立的行为过程模型如图2所示。

```
22  <classifier name="Event Name" keys="concept:name"/>
23  <classifier name="Resource" keys="org:resource"/>
24  <string key="source" value="Rapid Synthesizer"/>
25  <string key="concept:name" value="excercise.mxml"/>
26  <string key="lifecycle:model" value="standard"/>
27  <trace>
28    <string key="concept:name" value="Case2.0"/>
29    <event>
30      <string key="org:resource" value="UNDEFINED"/>
31      <date key="time:timestamp" value="2013-05-04T17:21:03.471+02:00"/>
32      <string key="concept:name" value="login"/>
33      <string key="lifecycle:transition" value="complete"/>
34    </event>
35    <event>
36      <string key="org:resource" value="UNDEFINED"/>
37      <date key="time:timestamp" value="2013-05-04T17:22:03.471+02:00"/>
38      <string key="concept:name" value="search-goods"/>
39      <string key="lifecycle:transition" value="complete"/>
40    </event>
41    <event>
42      <string key="org:resource" value="UNDEFINED"/>
43      <date key="time:timestamp" value="2013-05-04T17:23:03.471+02:00"/>
44      <string key="concept:name" value="order-items"/>
45      <string key="lifecycle:transition" value="complete"/>
46    </event>
47    <event>
48      <string key="org:resource" value="UNDEFINED"/>
49      <date key="time:timestamp" value="2013-05-04T17:24:03.471+02:00"/>
50      <string key="concept:name" value="invaid-order"/>
51      <string key="lifecycle:transition" value="complete"/>
52    </event>
53    <event>
54      <string key="org:resource" value="UNDEFINED"/>
55      <date key="time:timestamp" value="2013-05-04T17:25:03.471+02:00"/>
56      <string key="concept:name" value="get-card"/>
57      <string key="lifecycle:transition" value="complete"/>
58    </event>
59    <event>
60      <string key="org:resource" value="UNDEFINED"/>
61      <date key="time:timestamp" value="2013-05-04T17:25:03.471+02:00"/>
```

图1 监测日志的部分文档截图

Fig.1 The partial document screenshot of the monitoring log

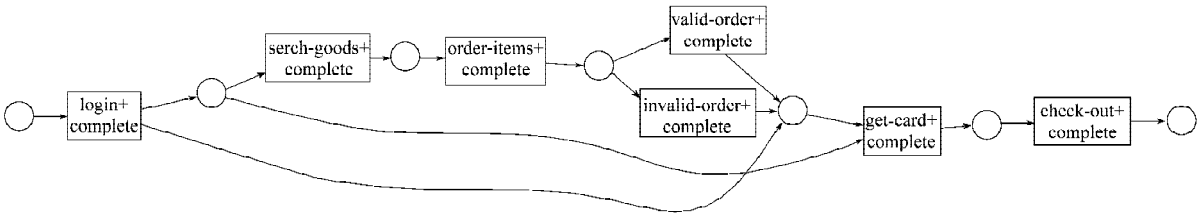

图2 采用petri网表示的监测日志行为过程模型图

Fig. 2 The behavior process model diagram of the monitoring log using petri net

采用Prom6.2进行建模，运行时的截图见图3。

图3 运行时的截图

Fig. 3 The running screenshot

2.4 原理及过程分析

2.4.1 petri网形式化描述

petri网为一种图形化的建模和分析工具。petri网能表达并发事件，通过描述变迁之间的因果关系，构造时序。因此，软件过程模型可以使用经典的petri网来描述。petri网由一个四元组（库所、变迁、输入函数、输出函数）组成，任何图形均可以映射到这个四元组上来，反之亦可。

经典的petri网包含两种元素：变迁（用方形节点表示）和库所（用圆形节点表示），而库所与变迁之间的有向弧表示两者之间的关系，有向弧只能连接异种节点，其中，库所中包含着动态对象令牌

（token），它可以从一个库所移动到另一个库所。petri 网的状态由令牌所在库所的分布所决定。如果一个变迁的每个输入库所都拥有令牌，该变迁即被允许，此时，输入库所所在的令牌将被消耗，同时为输出库所产生令牌，此为 petri 网的行为规则。

使用 petri 网来描述软件行为过程，可以较好地表示网络化软件系统的工作流和构成软件系统的各个组件以及组件与组件之间的相互关系等。

2.4.2　alpha 算法描述

alpha 算法由科学家 Wil van der Aalst 提出，其具体描述如下，其中，W 为 T 上的工作流日志：

1）$T_W=\{t\in T|\exists\sigma\in W,t\in\sigma\}$，

2）$T_I=\{t\in T|\exists\sigma\in W,t=\text{first}(\sigma)\}$，

3）$T_O=\{t\in T|\exists\sigma\in W,t=\text{last}(\sigma)\}$，

4）$X_W=\{(A,B)|A\subseteq T_W\wedge B\in T_W\wedge\forall a\in A\forall b\in Ba\to_W b\wedge\forall a_1a_2\in A\ a_1\#_W a_2\wedge\forall b_1b_2\in B\ b_1\#_W b_2\}$，

5）$Y_W=\{(A,B)\in X_W|\forall(A',B')\in X_W\ A\subseteq A'\wedge B\subseteq B'\Rightarrow(A,B)=(A',B')\}$，

6）$P_W=\{P_{(A,B)}|(A,B)\in Y_W\}\cup\{i_W,o_W\}$，

7）$F_W=\{(a,P_{(A,B)})|(A,B)\in Y_W\wedge a\in A\}\cup\{(P_{(A,B)},b)|(A,B)\in Y_W\wedge b\in B\}\cup\{(i_W,t)|t\in T_I\}\cup\{(t,o_W)|t\in T_O\}$，

and $a(W)=(P_W,T_W,F_W)$。

2.4.3　过程挖掘工具 Prom6.2 工作机制

由于过程挖掘目标创建的是实时过程模型，而非静态模型，因此，寻找一种允许将当前系统软件行为交互状态和最新相关活动映射到新的、直观的用户界面成为迫切需求，过程挖掘工具 Prom6.2 即能满足这种需求。Prom6.2 是一项由科学家 Wil van der Aalst 领衔开发的开源过程挖掘工具，它输入后缀为.XES 或.MXML 格式的日志文档，输出为以各种建模语言表示的行为过程模型图，如 petri 网、UML 活动图、数据流图和有限状态机等。Prom6.2 能将复杂的过程挖掘算法隐藏到可以自动设置参数并提供适当的过程分析类型的用户界面后面。

2.4.4　交互行为合理性分析

网络化软件中，事件实例之间的交互行为是一个随机过程，单个实体的行为不能被孤立地看待，因此，必须对行为的合理性进行整体研究。

本文采用基于 alpha 算法的 petri 网建模方法，图 2 中圆形图形表示库所，方形图形表示变迁，也即事件实例，箭头代表有向弧，表示事件实例发生的先后次序。图 2 中共含有 3 个过程实例，用偏序关系表示为：

1）start=login ≻search-goods ≻order-items ≻valid-order ≻get-card ≻check-out=end，

2）start=login ≻search-goods ≻order-items ≻invalid-order ≻get-card ≻check-out=end，

3）strat=login ≻get-card ≻check-out=end。

这 3 个过程实例具有相同的起点和结束点。模型图中共含有 7 个事件实例，即通用过程行为模型元组中的 T 元素，每个事件实例的相对和绝对发生率如图 4 所示。

Class	Occurrences (absolute)	Occurrences (relative)
login+complete	3	20.0%
check-out+complete	3	20.0%
get-card+complete	3	20.0%
order-items+complete	2	13.333%
search-goods+complete	2	13.333%
valid-order+complete	1	6.667%
invalid-order+complete	1	6.667%

图4　事件实例的相对和绝对发生率

Fig. 4　The relative and absolute rate of the event instances

由图 4 可知，事件实例 login, check-out, get-card 发生的概率最高，而 valid-order 和 invalid-order 发生的概率最小。如果发生的概率高，表明该系统中此事件实例与其他事件实例交互频繁，行为关系密切，由于系统调用作为了程序函数访问资源的接口，故其整体系统调用的频度也高，如果发生的概率小则反之。同时，从图 4 中还可以看出，valid-order 和 invalid-order 两个事件实例之间是并行的关系，但是图中出现了 invalid-order → get-card 与实际操作不相符合的行为，此即所谓的反例路径，它是理论上要求不能发生但实际存在的一种异常行为，即交互行为不合理。因此，通过搜寻系统行为模型中的反例路径并加以分析，可以检测出系统的漏洞和异常。当然，该模型无法表达程序行为所有的状态，将模型再泛化，以建立精确动态的行为过程模型将是下一步的研究重点。

3　实例分析

本文采用事件实例及其偏序关系来描述系统运行时的交互行为状态，本研究组之前已验证了该方法的合理性[7]。事件实例即为 petri 网模型中所示的变迁（transition），它描述了系统运行时行为的交互关系。下面以湖南省株洲市公共自行车租赁系统为例，对系统行为的合理性加以分析验证。

株洲市公共自行车租赁系统为一个实际应用的复杂网络化软件系统，它由分布在株洲市 5 个行政区

的5个板块组成，每个区下面均设有一级站点，一级站点下又设有二级站点，一、二级站点共500余个，共同构成整个系统。每个站点相当于上述行为过程模型图中的一个事件实例，系统中各个站点之间通过刷卡借车和还车来进行交互，一次正常的（排除锁卡等异常行为）租借归还过程构成系统中的一个事件过程。这一过程是在系统的控制下完成的。此外，在系统的控制下，还涉及车辆的调度行为，如当某站点车辆达到饱和状态（实际设置为80%）或空缺状态（实际设置为20%）时，站点会自动触发控制中心来进行车辆调度，这种调度行为实质上就是一种网络化过程行为。当然，系统内部也存在着反例路径，如还车刷卡→死锁。这种调度行为可以通过建立行为过程模型图（见图5）来体现，图中变迁分别表示设定初始站点、初始站点与可达站点间按距排序、删除不可达站点、新站点按距排序、为所搜索的最后一层站点、不是搜索的最后一层站点、距离最大站点为下一站点、有车未运完、将新站点设置为初始站点和所有车运完。

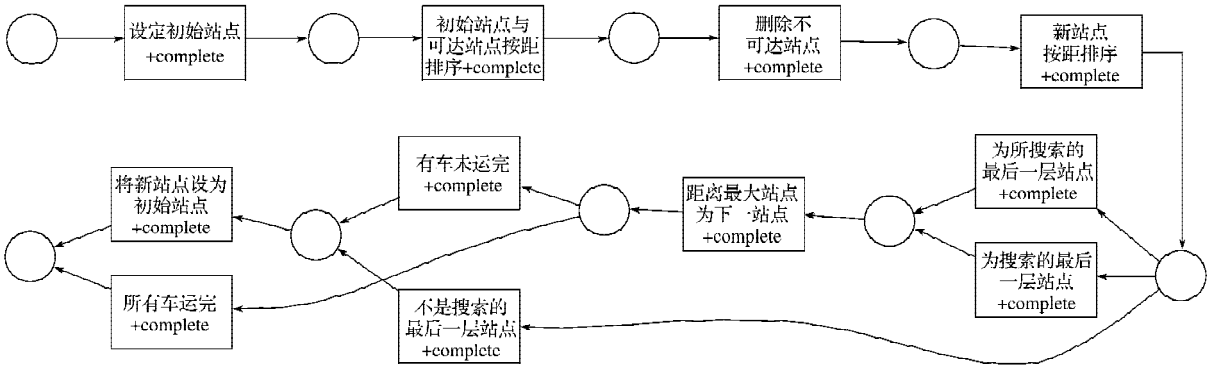

图5 株洲市公共自行车租赁系统车辆调度模型图

Fig. 5 The vehicle scheduling model diagram of Zhuzhou public bike rental system

从上述模型图可以看出，系统内部组件之间交互行为较复杂，故对网络化软件系统进行过程模型的挖掘和分析是非常必要的。

通过对株洲市公共自行车租赁系统进行分析，可以得知，系统的交互行为复杂。对系统进行监控，可有效预防各种恶意行为，提高系统的安全性和可靠性。

4 结语

基于过程挖掘的网络化软件行为过程模型为网络化软件行为的分析提供了新的方法和研究途径。本文通过对所构建系统产生的日志文档进行行为挖掘，将这种规模庞大且内容复杂的记录文本抽象提取为直观、简练的软件行为过程模型，可以直观形象地描述系统（程序）在运行时的内在行为规律。通过分析模型中各种事件实例的交互行为是否符合预期的行为要求，可以揭示系统的异常行为或潜在威胁和漏洞，以便系统管理者提前预防与有效监控。鉴于此，通过对株洲市公共自行车租赁系统进行分析，验证系统的复杂交互行为，以对恶意行为防微杜渐，提高系统的安全性和可靠性。当软件交互行为受到外界因素的干扰而产生非预期行为时，还可以利用网络牵制技术，选择最优化的牵制策略，对系统中相关的节点进行适当的反馈控制，以保证系统行为的可信性。

通过对网络化软件行为过程模型的挖掘，以及分析行为过程模型的普适性、鲁棒性、牵制控制，以使整个系统行为维持在一个正常稳态的范围内，这将是下一步研究的重点。

参考文献：

[1] Li Bing，Ma Yutao，Liu Jing，et al. Advances in the Studies on Complex Networks of Software Systems[J]. Advances in Mechanics，2008(6)：805-814.

[2] Wang C W. Thoughts on Innovative Development of Computer Technologies[J]. Communications of the CCF，2007，3(1)：8-13.

[3] 何克清，彭 蓉，刘 玮，等. 网络式软件[M]. 北京：科学出版社，2008：103-105.
He Keqing，Peng Rong，Liu Wei，et al. Networked Software[M]. Beijing：Science Press，2008：103-105.

[4] He Keqing，Peng Rong，Liu Jing，et al. Design Methodology of Networked Software Evolution Growth

Based on Software Patterns[J]. Journal of Systems Science and Complexity，2006，19(2)：157-181.

[5] Gao Debin，Reiter Michael K，Song Dawn. Gray-Box Extraction of Execution Graphs for Anomaly Detection[C]// Proc. of the ACM Conf. on Computer and Communications Security. Washington：Research Collection School of Information Systems，2004：318-329.

[6] 彭　成，杨路明，满君丰. 网络化软件交互行为动态建模[J]. 电子学报，2013，41(2)：314-320.
Peng Cheng，Yang Luming，Man Junfeng. Dynamic Modeling of Networked Software Interactive Behavior[J]. Acta Electronics Sinica，2013，41(2)：314-320.

[7] Sekar R，Bendre M，Dhurjati D，et al. A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors [C]//Proc. of the IEEE Symp. on Security and Privacy. Oakland：IEEE Press，2001：144-155.

[8] Li Wen，Dai Yingxia，Lian Yifeng，et al. Context Sensitive Host-Based IDS Using Hybrid Automaton：Context Sensitive Host-Based IDS Using Hybrid Automaton[J]. Journal of Software，2009，20(1)：138-151.

[9] Giffin J T，Jha S，Miller B P. Efficient Context-Sensitive Intrusion Detection[C]//Proc. of the 11th Network and Distributed Systems Security Symp. San Diego：University of Wisconsin Press，2004：45-60.

[10] Hofmeyr S A，Forrest S，Somayaji A. Intrusion Detection Using Sequences of System Calls[J]. Journal of Computer Security，1998，6(3)：151-180.

[11] Li Peng，Park H，Gao Debin，et al. Bridging the Gap Between Data-Flow and Control-Flow Analysis for Anomaly Detection[C]//Proc. of the 2008 Annual Computer Security Applications Conf. Las Vegas：[s.n.]，2008：392-401.

[12] Humphrey W S. A Discipline for Software Engineering[M]. Boston：Addison-Wesley Longman Publishing Co. Inc，1995：11-13.

[13] Jaccheri M L，Picco G P，Lago P. Eliciting Software Process Models with the E3 Language[J]. ACM Transion Software Engineering Methodology，1998，7(4)：368-410.

[14] Baldassarre M T，Caivano D，Visaggio C A，et al. ProMisE：A Framework for Process Models Customization to the Operative Context[C]//Proceeding of the 2002 International Symposiumn on Empirical Software Engineering（ISESE 2002）. Japan：IEEE Computer Society，2002：103-110.

[15] Li Mingshu，Yang Qiusong，Zhai Jian. Systematic Review of Software Process Modeling and Analysis[J]. Journal of Software，2009，20(3)：524-545.

[16] Biermann A W，Feldman J A. On the Synthesis of Finite-State Machines from Samples of Their Behavior[J]. IEEE Transactions Computers，1972，21(6)：592-597.

[17] Zhai Jian，Yang Qiusong，Xiao Junchao，et al. Formalized Approach for Componentized Software Process Modeling [J]. Journal of Software，2011，22(1)：1-16.

（责任编辑：徐海燕）