面向业务流程的数据模型异常检测方法

刘之强 李红燕 王 磊 曲 强
(北京大学信息科学技术学院 北京 100871)
(机器感知与智能教育部重点实验室 北京 100871)

摘 要 当前,流程驱动的信息系统构建方式得到了越来越广泛的应用.在流程驱动的方式中,流程模型对数据模型有着不可忽视的影响.但是当前的数据模型异常检测方法都是针对数据模型本身的特点而未考虑流程模型.同样,流程模型的验证方法也缺乏对数据模型的考虑.文中提出并分析了面向业务流程的数据模型的异常问题,并给出了其 3 种基本类型.为了检测这些异常,文中提出了 Data-process Graph(DP-Graph)模型,将数据模型和流程模型放在统一的架构下进行研究.而后,基于 DP-Graph,文中提出了 DPGT 算法,有效地实现了面向业务流程的数据模型异常检测.文章中的实验结果验证了 DPGT 算法对于这些异常的高检出率.

关键词 数据模型异常检测;业务流程管理;工作流模型验证;信息系统构建
中图法分类号 TP18 **DOI 号**: 10.3724/SP.J.1016.2010.01349

A Data Model Anomalies Detection Method for Business Process Model

LIU Zhi-Qiang LI Hong-Yan WANG Lei QU Qiang
(*School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871)
(*Key Laboratory of Machine Perception, Ministry of Education, Beijing* 100871)

Abstract Nowadays, the process-driven method for information system construction is more and more widely used. In the process-driven method, the process model has significantly influence on the data model. Unfortunately, the existing data model anomalies detection methods are only about the anomalies of data model itself. These methods don't take the process model into account. Similarly, the process model verification methods also lack for consideration of data model. This paper analyses those anomalies and gives a basic classification of three classes of them. This paper also proposes a model called Data-process Graph (DP-graph) to build up linkage between the data model and process model. Based on DP-Graph, DPGT method is proposed to detect anomalies of data model for business process model. Experimental results show the high detection rate of the anomalies of the method.

Keywords data model anomalies detection; business process management; workflow verification; information system construction

1 引 言

传统的信息系统构建方式是数据驱动(data driving)的.这种方式主要关注业务数据的作用,数据模型成为信息系统构建的起点,要求业务流程模型完全适应数据模型.

近年来,随着业务流程管理[1](business process

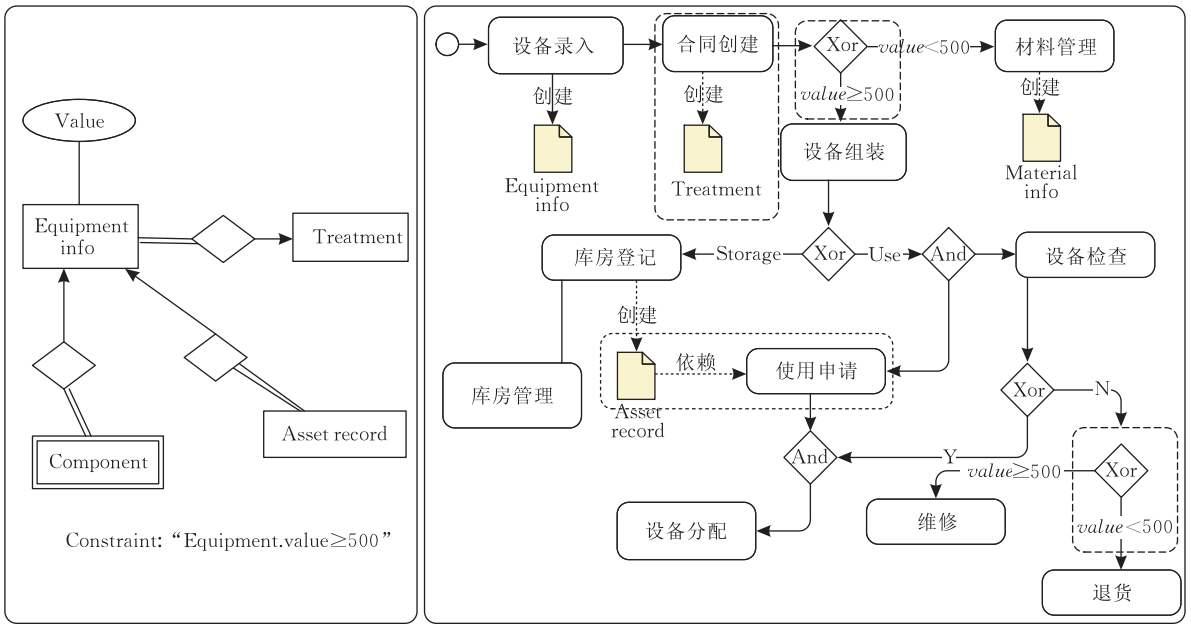
收稿日期:2010-06-11.本课题得到国家自然科学基金(60673113,60973002)和国家"八六三"高技术研究发展计划项目基金(2007AA01Z191,2009AA01Z150)资助.**刘之强**,男,1987 年生,硕士研究生,主要研究方向为信息系统自动化构建与业务流程. E-mail: liuzq@cis.pku.edu.cn. **李红燕**(通信作者),女,1970 年生,教授,主要研究领域为数据管理与挖掘. E-mail: lihy@cis.pku.edu.cn. **王 磊**,男,1983 年生,硕士研究生,主要研究方向为信息系统自动化构建与业务流程. **曲 强**,男,1985 年生,硕士研究生,主要研究方向为查询处理以及大规模图挖掘.

management)越来越受到重视,流程驱动的构建方式得到了广泛的应用.这种构建方式是指将业务流程模型作为整个系统构建的出发点,通过对流程模型的精化,衍生出诸如数据模型等其它的模型.

传统的数据模型的异常检测方法主要关注于数据模型本身的静态属性,比如数据模型本身的约束、是否满足范式要求等.但是,这些方法没有考虑业务流程对数据模型的影响.对于业务流程驱动的信息系统中数据模型的异常检测,需要综合考虑数据模型和业务流程模型的特点.因为即使在数据模型和业务流程模型本身均正确的情况下,仍会出现数据模型相对于流程模型的不一致,即出现数据模型的异常.本文立足于面向业务流程的数据模型异常检测,提出了自己的模型和方法并给出了相应的实验结果.

为了直观地说明面向业务流程时数据模型会发生的异常,本文给出图1所示的来自CCTV(China Central Television)物资管理系统的模型片段.图1展示了系统中的一个数据模型片段及其对应的流程模型片段(使用BPMN符号[2]表示).综合考虑数据模型与流程模型之间的交互性,在此例中可以发现以下几种由于数据模型不合理导致流程模型出现的异常:

(1)数据创建失败.在业务流程中的任务"设备录入"创建Equipment info数据对象,但是此时数据模型中Equipment info其依赖的数据对象Treatment还未创建.



(a)数据模型 (b)流程模型

图1 CCTV物资管理系统模型片段

(2)条件不能被满足.由于数据约束条件$value \geqq 500$的存在,任务"材料管理"和"退货"永远不能被执行.

(3)数据前提冲突.任务"使用申请"有一个前提数据对象Asset record.但这个数据对象是由一个不被触发的任务"库房登记"创建的.这样任务"使用申请"就会执行失败.

从以上的应用场景,可以发现在面向业务流程的时候,数据模型会发生一些与流程模型的不一致,即异常.通过日常的项目工作,可以发现这些异常具有如下的3个特点:

(1)这些异常在数据模型中是普遍存在的.在CCTV物资管理系统(业务流程模型包含6个子流程,97个任务,数据模型包含124个实体)中,在数据模型与流程模型各自均正确的前提下,发现上述3种错误43个.在实际的医院信息系统中(Hospital Information System)仅住院管理部分就发现了上述错误101个.可见在实际的信息系统构建过程中,数据模型异常的发生是具有普遍性的.

(2)某些异常会直接导致系统运行时出错.而解决这次错误要涉及代码级的改动.系统也必须在进行代码改动后重新上线.

(3)利用人工的方法来检测这些异常对工作人员素质要求较高.这项工作需要工作人员既熟悉数

据模型又熟悉业务流程模型. 并且从上面的 CCTV 物资管理系统和 HIS 的例子也可以发现，实际项目中数据模型和业务流程模型的庞大也使得人工检测这些异常变得枯燥乏味. 因此，寻求自动化的方法来高效而准确地发现这些异常十分具有现实意义.

解决面向业务流程的数据模型异常检测问题是具有挑战性的. 发生这些异常的时候，数据模型和业务流程模型本身都是正确的. 这些异常的产生从根源上来说是由于数据模型与流程模型之间的不一致性. 因此，想要检测这些异常就必须将数据模型和流程模型二者联系在一起. 但此前并没有建立这两者之间联系的方法. 为了解决这个问题，本文提出了 Data-Process Graph(DP-Graph)模型来联系这两个模型，并给出了 DPGT 方法来检查这些异常.

本文第 2 节主要讨论一些相关工作；第 3 节具体分析并给出面向业务流程的数据模型异常问题的分类；第 4 节提出 DP-Graph 模型来联系数据模型与业务流程模型；第 5 节给出 DPGT 方法来检测面向业务流程模型的数据模型异常；第 6 节给出相应的实验结果. 最后是对全文的总结.

2　相关工作

2.1　数据模型异常检测

对于数据模型本身的正确性检测，已经有非常成熟的工具[3-4]. 常见的数据建模工具都提供针对诸如 ER 模型或者关系模型的检查功能，比如 Sybase PowerDesigner、ERWin、Rational Rose Enterprise 等. 这些针对数据模型的检测主要针对数据模型结构方面的异常，例如数据模型的自洽、约束的一致性、是否满足范式要求等静态特征. 但这些检测完全没有考虑业务流程对数据模型的影响. 在本文后续篇幅中讨论数据模型异常检测问题时，假定所讨论的数据模型已经通过了这些检查，是自身静态结构正确的数据模型.

2.2　业务流程验证

实现面向业务流程的数据模型异常检测，应该了解一些传统的业务流程验证方法. 近年来，业务流程验证在流程模型管理领域引起了极大的关注. 以往有从多个不同的视角来针对业务流程的研究，比如控制流(control flow)、数据流(data flow)和业务流程的时间坐标等.

业务流程模型中的控制流的验证方法主要是针对业务流程模型本身来分析其控制流中的异常问题. 典型的方法有 Petri-net[5-6]、YAWL[7] 和 UML 活动图[8-9]等. 为了分析流程模型中的异常，一系列方法比如线性代数[10](linear algebraic)、有限状态验证[11](finite state verification)以及可覆盖图形分析[12](coverability graph analysis)等方法被提出来. 一些流程模型验证的工具也被设计并实现了，比如 Wofan[13]、WofanYAWL[12]等. 但这类方法的主要问题是仅仅考虑了流程模型本身，并未考虑与流程模型相关的数据模型，从而并不支持数据模型的异常检测.

当前也有一些基于数据流(data flow)的业务流程模型验证方法[14-15]. 这类方法可以检测并发现诸如数据丢失(missing data)、数据冲突(conflicting data)以及数据冗余(redundant data)等. 这类方法仅仅考虑了在工作流模型中的数据，并未对静态的数据模型有所考虑，忽略了数据对象之间的相关性. 例如数据模型中的依赖性问题，这类方法并不能检测出.

在文献[16]中给出了数据模型与业务流程模型之间的相互关系，提出了结合业务流程模型来进行数据模型验证的重要性. 但是并没有给出面向业务流程模型的数据模型异常检测方法.

3　面向业务流程模型的数据模型异常分析

为了分析面向业务流程的数据模型异常，首先要理解数据模型和业务流程模型之间的交互方式. 这种交互方式可以分成两大类：

(1) 数据产品(data products). 业务流程所操作的数据产品是由数据模型定义的.

(2) 基于数据的控制(data-based control). 数据作为流程的控制条件. 主要有两种情况：

① 任务的数据前提(tasks' data preconditions). 流程中任务的执行常常依赖特定的数据对象[16].

② 数据驱动的流程控制(data-based control). 业务流程的控制结构可以被数据影响[16].

面向业务流程模型的时候，数据模型之所以后发生异常正是由于这两者之间的交互性，图 2 从流程模型影响数据模型和数据模型影响流程模型两个维度来分析面向业务流程时数据模型中可能出现的异常.

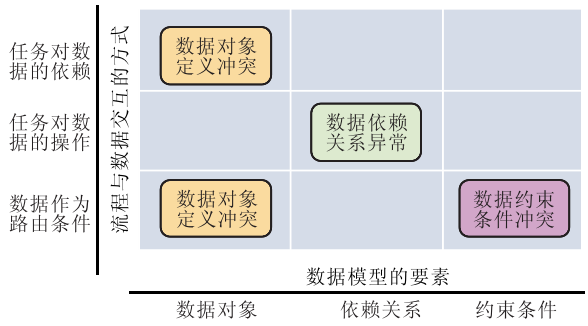


图 2 面向流程模型的数据模型异常

从图 2 的分析,可以得到基于业务流程模型的数据模型 3 种基本的异常:

(1) 数据依赖关系异常 DDA(Data Dependence Anomaly). 对应第 1 部分实例中的第 1 个错误. 由于数据对象所依赖的数据对象未创建,流程模型中的任务创建的数据对象不能成功插入数据模型. 根本原因是该数据对象所在的数据模型中的依赖关系设计不合理.

(2) 数据对象定义冲突 DDC(Data Definition Contradiction). 对应第 1 部分的第 3 个错误. 流程中的某个任务的前提数据对象在任务触发的时候仍然未创建,导致任务不能继续执行. 这个异常的出现是由于有些数据对象在定义时未考虑到创建该数据对象的任务和将该数据对象作为前提的任务之间的关系. 该数据对象必须根据流程的实际情况进行重新定义.

(3) 数据约束条件冲突 DCC(Data Condition Contradiction). 定义第 1 部分中的第 2 个错误. 该错误发生的主要原因是数据约束条件的不合理. 由于某些不合理的数据约束条件,流程中的某些分支将会永远无法达到.

为了检测这 3 种基本异常,下文提出了 DP-Graph 以建立数据模型和流程模型之间的联系并给出了 DP-Graph 的构造方法. 而后基于 DP-Graph,本文提出了面向业务流程的数据模型异常检测方法 DPGT.

4 Data-Process Graph(DP-Graph)

在实际的信息系统构建过程中,数据模型与业务流程模型是两个分离的部分. 为了对数据模型中的面向流程模型的异常进行检测,必须给出数据模型与业务流程模型之间关系的统一描述模型. 通过这个模型,才能把数据模型和流程模型放在一个统一的框架下研究,从而检测出面向业务流程的数据模型中的异常.

本文给出了一种 DP-Graph(Data-Process Graph)模型来描述数据模型与业务流程之间的关系.

DP-graph 由一组顶点集和一组边集构成,顶点集包含了从数据模型和业务流程模型中抽取的对象,边集用来表达各种对象之间的关系.

图 3 是一个根据图 2 中的数据模型和流程模型构建的 DP-Graph 实例.

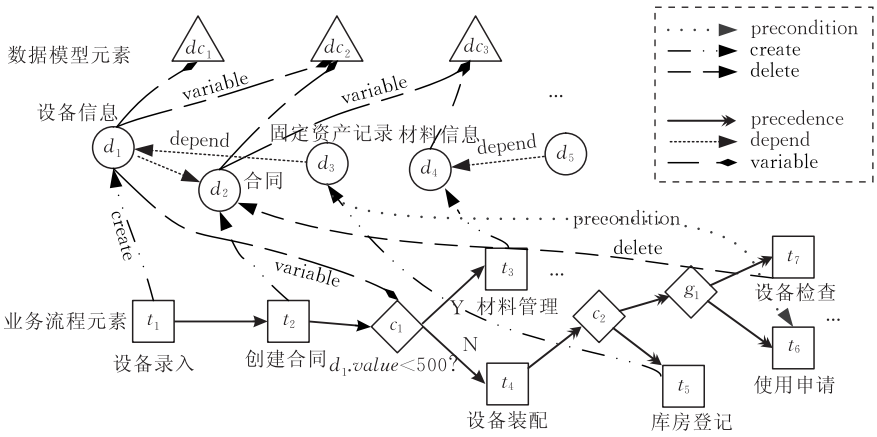


图 3 Data-process Graph 实例

定义 1(DP-Graph). DP-Graph 是一个有向图,DP-Graph=$\langle V,E\rangle$,其中 V 为顶点集:

$$V=Dataos\cup D\_Constrains\cup Task\cup Gateways\cup Conditions,$$

$$E=D_{\text{depend}}\cup DC_{\text{variable}}\cup P_{\text{precedence}}\cup CT_{\text{true}}\cup CT_{\text{false}}\cup DP_{\text{preconditions}}\cup PD_{\text{create}}\cup PD_{\text{delete}}\cup C_{\text{variable}}.$$

顶点集 V 由 5 个子集组成:

$Dataos$, $D\_Constrains$ 这两个集合包含了从数据模型中抽象出来的元素.

(1) $Dataos=\{d_1,d_2,\cdots,d_v\}(v\geq 0)$,是数据对

象集合，其中 d_i 是数据对象.

(2) $D\_constrains=\{dc_1,dc_2,\cdots,dc_p\}(p\geq 0)$，是数据约束条件集合，其中 dc_p 是一个数据约束条件，由关于数据对象的谓词表示.

$Tasks$，$Gateways$，$Conditions$ 这 3 个集合包含了从业务流程模型中抽象出来的元素.

(3) $Tasks=\{t_1,t_2,\cdots,t_m\}(m\geq 1)$，是业务流程中任务的集合，其中 t_i 是任务.

(4) $Gateways=\{g_1,g_2,\cdots,g_n\}(n\geq 0)$，是流程控制节点集合，其中 g_i 是控制节点对象.

(5) $Conditions=\{c_1,c_2,\cdots,c_u\}(u\geq 0)$，是业务流程中控制条件的集合，其中 c_i 是一个控制条件，表示为关于数据对象的谓词.

边集 E 由 9 个子集组成：

D_{depend}、DC_{variable} 这两个集合包含了数据模型元素之间的关系：

(1) $D_{\text{depend}}\subseteq Dataos\times Dataos$，表示数据对象之间的依赖关系，$\langle d_1,d_2\rangle\in D_{\text{depend}}$，表示数据对象 d_1 依赖于 d_2.

(2) $DC_{\text{variable}}\subseteq D\_Constrains\times Dataos$，如果 $\langle dc,d\rangle\in DC_{\text{variable}}$，表示数据对象 d 是数据约束 dc 的一个变量.

$P_{\text{precedence}}$、CT_{true}、CT_{false} 这 3 个集合表示业务流程模型元素之间的关系：

(3) $P_{\text{precedence}}\subseteq(Tasks\cup Gateways)\times(Tasks\cup Gateways\cup Conditions)$，是流程元素的前驱-后继关系.

(4) $CT_{\text{true}}\subseteq Conditions\times(Tasks\cup Gateways\cup Conditions)$，是触发关系，指的是 $Condition$ 满足时要触发的流程元素.

(5) $CT_{\text{false}}\subseteq Conditions\times(Task\cup Gateways\cup Conditions)$，是触发关系，指的是 $Condition$ 不满足时要触发的流程元素.

$DP_{\text{preconditions}}$、$PD_{\text{create}}$、$PD_{\text{delete}}$ 和 C_{variable} 这 4 个集合表示了业务流程模型元素与数据模型元素之间的关系.

(6) $DP_{\text{preconditions}}\subseteq(Tasks\cup Gateways)\times Dataos$，是业务流程元素对数据对象的依赖关系.

(7) $PD_{\text{create}}\subseteq Tasks\times Dataos$，如果 $\langle t,d\rangle\in PD_{\text{create}}$，任务 t 创建数据对象 d.

(8) $PD_{\text{delete}}\subseteq Tasks\times Dataos$，如果 $\langle t,d\rangle\in PD_{\text{delete}}$，任务 t 删除数据对象 d.

(9) $C_{\text{variable}}\subseteq Conditions\times Dataos$，如果 $\langle c,d\rangle\in C_{\text{variable}}$ 表示数据对象 d 是数据约束 c 的一个变量.

Gateway 定义了流程控制流的逻辑节点，与流程的基本控制结构模式相对应，Gateway 能表达各种控制流的逻辑分支以及合并的语义.

定义 2(Gateways). Gateway 是一个 5 元组：$\text{Gateway}=\langle Type,ControlRules\rangle$，其中

$$Type\in\{andjoin,ordiv,orjoin\}.$$

$ControlRules$ 是一组控制流的触发规则，采用 Datalog 形式定义，型如

$$Trigger(t_b):-IsCom(t_a)\&c_{ab}.$$

这条规则表示“如果 t_a 完成而且条件 c_{ab} 满足，则触发任务 t_b”. 控制流的路由通过触发规则表示.

以上给出了 DP-Graph 的定义，现在分析给出 DP-Graph 的特点：

(1) DP-Graph 是一个形式化的图定义. 它将数据模型与业务流程模型之间的关系通过图的边抽象地表示出来.

(2) DP-Graph 是一个面向分析的图. 它的定义是为了检测数据模型中相对于业务流程模型的异常，忽略了对于该分析没有价值的信息.

(3) DP-Graph 独立于特定的数据模型和流程模型描述语言. 只要有关于数据模型和流程模型足够多的信息，就可以创建出 DP-Graph.

为了实现基于 DP-Graph 的异常检测，首先要构建 DP-Graph. 下面讨论 DP-Graph 的构建算法. 对于一种业务流程构建方式而言，需要根据它的数据模型和流程模型来构建 DP- Graph.

DP-Graph 本质上是一个有向图，可以用邻接表和邻接矩阵的数据结构来表示. 根据流程模型和数据模型来构建 DP-Graph，可以首先通过对数据模型中 D_{depend} 的分析，生成 DP-Graph 中数据模型对应的邻接表 D-Graph. 然后再对流程模型中的 $P_{\text{precedence}}$、CT_{true}、CT_{false} 元素进行分析，生成 DP-Graph 与流程模型中对应的邻接表 P-Graph. 而后 DC_{variable} 和 C_{variable} 两种元素可以用邻接矩阵表示. 最后再用 3 组共 6 个邻接表来表示 D-Graph 和 P-Graph 之间的关系 $DP_{\text{preconditioins}}$、$PD_{\text{create}}$、$PD_{\text{delete}}$，便构建出了完整的 DP-Graph. 可见一个 DP-Graph 共包括 8 个邻接表和两个邻接矩阵. 具体的构建算法如算法 1 所示.

算法 1. DP-Graph Construction.

输入：数据模型 dm，流程模型 pm

输出：DP-Graph DPG

变量定义：邻接表 DG，PG，$DP1$，$DP2$，$PDC1$，$PDC2$，$PDD1$，$PDD2$

邻接矩阵 $\boldsymbol{CV}$，$\boldsymbol{DCV}$

1. 创建邻接表 D-Graph DG，DG 的每个节点对应于一个 Dataos；

2. For every D_{depend} 中的元素〈D_m,D_n〉
3. $DG[m]$链接 n;
4. 创建邻接表 P-Graph PG,
PG 的每个节点对应于一个 Task 或者 Gateway;
5. For every $P_{precedure}$ 中的元素〈P_m,P_n〉 和 CT_{true} 中的元素〈P_m,P_n〉
6. $PG[m]$ 链接 n;
7. For every CT_{false} 中的元素〈P_m,P_n〉
8. $PG[m]$ 链接 n;
9. 创建邻接表 $DP1$ 和 $DP2$ 来表示 $DP_{preconditions}$,两个表分别从 DG 到 PG 及 PG 到 DG 两个方向建立;
10. 同理创建邻接表 $PDC1$ 和 $PDC2$ 来表示 PD_{create};
11. 同理创建邻接表 $PDD1$ 和 $PDD2$ 来表示 PD_{delete};
12. 创建邻接矩阵 ***CV*** 和 ***DCV***;
13. ***CV*** 矩阵的每一行对应一个 Conditions,每一列对应一个 Dataos;
14. ***DCV*** 矩阵的每一行对应一个 Constraints,每一列对应一个 Dataos;
15. 将 ***CV*** 和 ***DCV*** 的所有元素初始化为 0;
16. For every $DC_{variable}$ 的元素〈DC_m,DO_n〉
17. ***DCV***$[m][n]\leftarrow 1$;
18. For every $C_{variable}$ 的元素〈C_m,DO_n〉
19. ***CV***$[m][n]\leftarrow 1$;
20. DG,PG,$DP1$,$DP2$,$PDC1$,$PDC2$,$PDD1$,$PDD2$,***CV***,***DCV*** 共同构成一个完整的 DP-Graph DPG

下面分析 DP-Graph 构建算法的复杂度. 该算法共需要构建 8 个邻接表和两个邻接矩阵. 设流程模型任务数为 t,流程控制条件数为 k,控制变量数为 l. 数据模型数据对象数为 d,数据约束条件数为 m,约束变量数为 n. 复杂度为 $O(4t+4d+kl+mn)$.

5 基于 DP-Graph 的数据模型异常检测

本文提出的异常检测方法的主要思想是基于 DP-Graph,应用一组状态转移规则,搜索系统的状态空间,从而发现异常状态. 首先,本文给出 DP-Graph 的状态空间. 这些状态空间可以看作是真实系统状态的仿真. 然后,相关的异常状态的定义也会被给出. 最后,本文给出了一些推动算法进程的状态转移规则.

5.1 状态空间

DP-Graph 的状态由任务、数据对象和控制条件的状态构成. 本节给出了任务、数据对象和控制条件状态定义,并以此为基础定义了 DP-Graph 的状态空间.

定义 3(DP-Graph 状态空间). DP-Graph 的状态空间定义为笛卡尔积:$\boldsymbol{S}_{dpg}=TS\times DS\times CS$,其中 TS,DS 和 CS 分别是 Tasks,Dataos 和 Conditions 的状态空间. 其中

$$TS=Tasks\mapsto\{initial,ready,waiting,running,suspended,complete,over\},$$
$$DS=Dataos\mapsto\{defined,undefined,waiting\},$$
$$CS=Conditions\mapsto\{undefined,waiting,uncertain,true,false\}.$$

图 4 展示了任务、数据对象和控制条件的状态转移图. 其中灰色表示的是与数据和流程一致性异常有关的状态,这将在定义 6 中详细解释. 状态之间的转移即推理规则将会在 5.2 节中给出详细定义.

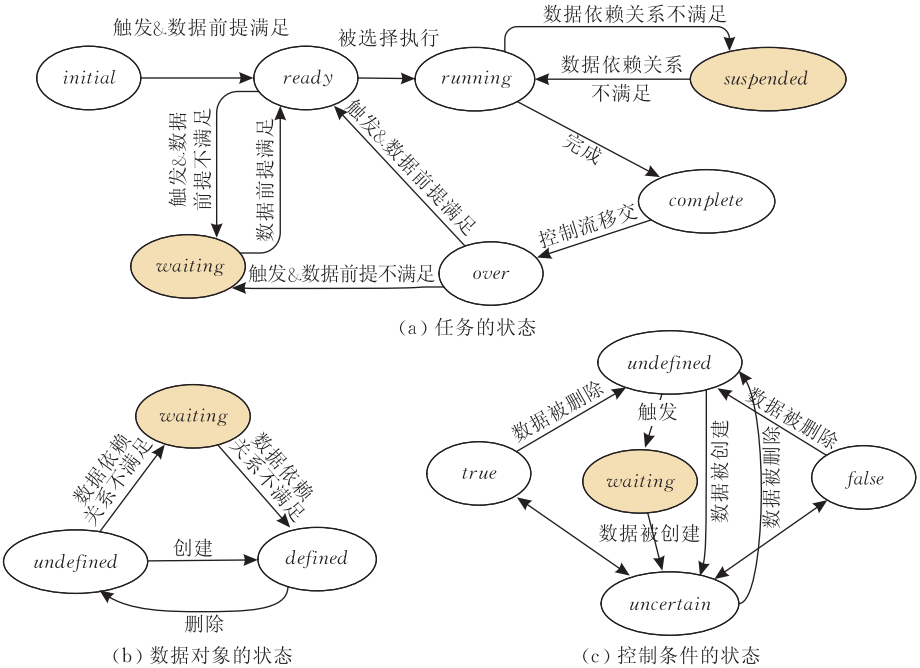


图 4 DP-Graph 状态空间

在系统状态中，有两个特殊的状态需要专门定义，分别是初始状态和结束状态. 初始状态标志着系统执行最初的状态，结束状态标志着系统执行的完成.

定义 4(初始状态). 初始状态(start state)是一个状态向量 $\boldsymbol{SV}_{\text{start}}=\langle TSV,DSV,CSV\rangle\in\boldsymbol{S}_{\text{dpg}}$，其中

$$TSV(t)=\begin{cases}ready, & t=T_{\text{start}},\\ initial, & t\neq T_{\text{start}}.\end{cases}$$

并且

$$DSV\in Dataos\mapsto\{undefined\},$$
$$CSV\in Conditions\mapsto\{undefined\}.$$

定义 5(结束状态). 结束状态(successful state)是一个状态向量 $\boldsymbol{SV}_{\text{successful}}=\langle TSV,DSV,CSV\rangle\in S_{\text{dpg}}$，其中

$$TSV(t)=\begin{cases}complete, & t=T_{\text{end}}\\ initial\ \text{or}\ over, & t\neq T_{\text{end}}\end{cases}.$$

DP-Graph 可以看作是对业务流程模型和数据模型构成的系统的一种抽象描述. 系统执行过程中所能到达的状态是整个 DP-Graph 状态空间的子集. 在第 3 部分所给出的基于流程模型的数据模型 3 种异常反应在 DP-Graph 的状态空间中，就是本文定义 6 要定义的异常状态.

定义 6(异常状态).

下面 3 种状态即为基于流程模型的数据模型异常状态：

(1) 数据依赖关系异常 DDA(Data Dependence Anomaly) 状态，当且仅当 $(\wedge D\_Constrains\rightarrow true(c))\vee(\wedge D\_Constrains\rightarrow false(c))$，其中“$\wedge D\_Constrains$”代表 $D\_Constrains$ 集合中所有元素的合取.

(2) 数据对象定义冲突 DDC(Data Definition Contradiction)状态，当且仅当 $\exists d\ waiting(d)\vee\exists t\ suspended(t)$.

(3) 数据约束条件冲突 DCC(Data Condition Contradiction)状态，当且仅当 $\exists c\ waiting(c)\vee\exists t\ waiting(t)$.

在图 4 中，灰色标出的状态是与基于流程模型的数据模型异常有关的状态.

5.2 推理规则定义

本节定义了一组状态转移规则，用来表达 DP-Graph 的状态转换. 应用这组规则，可以进行推理，从而检测可能到达的异常状态. 一条状态转移规则表示为一条逻辑语句，形如：$a\Rightarrow b$，其中 a 是条件，b 表示应用规则的结果，下面用 t,c,d,g 来分别表示 $task,condition,dataos$ 和 $gateway$ 4 种元素.

状态转移规则定义如下，根据它们实际语义的不同分为 5 个规则组，分别与流程执行的不同环节对应.

规则组 1(选择). 规定处在当前状态的流程可以选择执行的任务.

$ready(t)\wedge P_{\text{dpd}}(t)\Rightarrow running(t)$，其中 $P_{\text{dpd}}(t):-\forall d(\langle t,d\rangle\in DP_{\text{precondition}}\rightarrow defined(d))$.

如果任务 t 的全部前提数据都存在，则 t 进入运行状态.

$ready(t)\wedge\neg P_{\text{dps}}\Rightarrow waiting(t)$.

规则组 2(操作). 表示任务执行数据操作引起的状态转移.

$undefined(d)\wedge\exists t\ (running(t)\wedge\langle t,d\rangle\in PD_{\text{create}}\wedge P_{\text{ddd}}(d))\Rightarrow defined(d)$，其中

$P_{\text{ddd}}(d):-\forall d'(\langle d,d'\rangle\in D_{\text{depend}}\rightarrow defined(d'))$.

如果数据 d 所依赖的数据全部存在，则任务 t 可以创建数据 d.

$undefined(d)\wedge running(t)\wedge\langle t,d\rangle\in PD_{\text{create}}\rightarrow\neg P_{\text{ddd}}(d)\Rightarrow waiting(d)\wedge suspended(t)$.

数据 d 所依赖的数据未创建，则 t 进入 suspended 状态，d 进入 waiting 状态，发生异常.

$running(t)\wedge defined(d)\wedge\langle t,d\rangle\in PD_{\text{delete}}\Rightarrow undefined(d)$；

$defined(d)\wedge\exists d',\exists t(\langle d,d'\rangle\in D_{\text{depend}}\wedge running(t)\wedge defined(d')\wedge\langle t,d'\rangle\in PD_{\text{delete}})\Rightarrow undefined(d)$.

删除数据 d 的同时删除所有依赖 d 的数据对象.

规则组 3(唤醒). 表示数据状态的改变触发相应流程元素状态的变化.

$waiting(t)\wedge\forall d(\langle t,d\rangle\in DP_{\text{preconditions}}\rightarrow defined(d))\Rightarrow ready(d)$；

$waiting(c)\wedge\forall d(\langle c,d\rangle\in C_{\text{variable}}\rightarrow defined(d))\Rightarrow uncertain(c)$；

$waiting(d)\wedge\forall d'(\langle d,d'\rangle\in D_{\text{depend}}\rightarrow defined(d'))\Rightarrow defined(d)$；

$suspended(t)\wedge\forall d(\langle t,d\rangle\in PD_{\text{create}}\rightarrow\neg waiting(d))\Rightarrow running(t)$.

规则组 4(完成). 表示任务完成时状态的改变.

$running(t)\wedge\forall d((\langle t,d\rangle\in PD_{\text{create}}\rightarrow defined(d))\wedge(\langle t,d\rangle\in PD_{\text{delete}}\rightarrow undefined(d)))\Rightarrow complete(t)$.

规则组 5(触发). 表示对后续任务的触发.

$complete(t) \wedge (over(t') \vee initial(t')) \wedge \langle t,t' \rangle \in P_{precedence} \Rightarrow over(t) \wedge ready(t')$;

t 执行完成后，开始执行 t 的后续任务 t'.

$complete(t) \wedge \langle t,g \rangle \in P_{precedence} \Rightarrow g.ControlRules$;

$P_{ttc}(t,c) \wedge true(c) \wedge \langle c,t' \rangle \in CT_{true} \wedge (over(t') \vee initial(t')) \Rightarrow over(t) \wedge ready(t')$，其中 $P_{ttc}(t,c) :- complete(t) \wedge \langle t,c \rangle \in P_{precedence}$;

$P_{ttc}(t,c) \wedge false(c) \wedge \langle c,t' \rangle \in CT_{false} \wedge (over(t') \vee initial(t')) \Rightarrow over(t) \wedge read(t')$;

$P_{ttc}(t,c) \wedge uncertain(c) \wedge P_{canBeTure}(c) \wedge \langle c,t' \rangle \in CT_{true} \wedge (over(t') \vee initial(t')) \Rightarrow over(t) \wedge ready(t') \wedge true(c)$，其中

$P_{canbeture} :- \neg(\wedge D\_Constrains \rightarrow false(c))$;

$P_{ttc}(t,c) \wedge uncertain(c) \wedge P_{canBeFalse}(c) \wedge \langle c,t' \rangle \in CT_{false} \wedge (over(t') \vee initial(t')) \Rightarrow over(t) \wedge ready(t') \wedge false(c)$，其中

$P_{canbefalse} :- \neg(\wedge D\_Constrains \rightarrow ture(c))$.

检测是否存在不可达分支.

$P_{ttc}(t,c) \wedge undefined(c) \Rightarrow waiting(c)$.

5.3 DPGT：基于 DP-Graph 的数据模型异常检测方法

根据前面定义的推理规则，本文提出了面向业务流程模型的数据模型异常检测算法 DPGT(DP-Graph based Detection). DPGT 的基本思想是在 DP-Graph 的基础上，应用前面定义的状态转换规则，搜索 DP-Graph 的状态空间，检查前面定义的异常状态，从而发现面向流程模型的数据模型异常.

DPGT 基本过程是从初始状态开始，应用状态转换规则，得出所有可能到达的状态，当所有可达状态搜素完毕时算法结束. 过程中记录下发现的异常状态.

下面是基本的 DPGT 算法.

算法 2. DP-Graph Based Detection.

输入：DP-Graph dpg，状态向量 ***sv***

输出：错误状态

变量定义：*Stack SVstack*，状态向量 ***tempsv***，***tempsv2***，任务集 *rs*

```
1. SVstack.push(SV_start);
2. while SVstack 非空
3.    sv=SVstack.pop();
4.    检查 sv 是否异常状态;
5.    if sv 不是"结束状态";
6.       rs←所有"就绪"状态的任务;
7.       if(rs 非空)
8.       报告非正常终止(死锁);
9.    else
10.      for rs 中的每个任务
11.         Tempsv←sv;
12.         应用"choose"(dpg,tempsv);
13.         应用"operation"(dpg,tempsv);
14.         应用"awaken"(dpg,tempsv);
15.         应用 "complete"(dpg,tempsv);
16.         for 每个 "trigger" 规则
17.            tempsv2←tempsv;
18.            应用 "trigger"(dpg,tempsv2);
19.            if tempsv2 未访问过
20.               SVstack.push(tempsv2);
21.         end for
22.      end for
23. end while
```

以下是对 DPGT 算法进行复杂度分析. DPGT 算法的本质是按照多个不同的执行序列来依次模拟地执行流程中的各个任务，在执行的过程中发现数据模型的异常. 不妨设执行一个任务的平均时间为 1，算法复杂度的关键便是检测并行分支中的不同任务执行序列的个数. 假如有 m 个并行分支，每个分支有 n 个任务，可能的执行序列个数是 $C_n^n \times C_{2n}^n \times C_{3n}^n \times \cdots \times C_{mn}^n$，执行每个序列需要检查 mn 个任务的状态，故总的复杂度为 $O(C_n^n \times C_{2n}^n \times C_{3n}^n \times \cdots \times C_{mn}^n \times mn)$.

6 实 验

本次实验是基于 CCTV 物资管理系统来进行的. 中央电视台物资管理系统包含了复杂的业务流程模型和数据模型，业务流程模型包含 6 个子流程，97 个任务，数据模型包含 124 个实体. 业务流程的构建以及任务操作的描述在 PASE[17] 中完成. 该系统是一个典型的信息系统.

首先用文献[14]中提出的工作流模型中的 data-flow 验证方法进行实验. 在相关工作部分已经分析过该方法是基于流程模型中的动态数据，并不针对于数据模型. 这种方法对于数据对象定义冲突(DDC)异常(对应于该方法中的 missing data 情况)在流程模型中可以检测出来. 但由于其并未考虑数据模型中的依赖关系和约束条件，其对于其它两类异常是无能为力的. 表 1 是这种方法在 CCTV 物资管理系统中的实验结果. 该表列出了异常发现的数量. 这些异常便是第 3 部分提出来的数据依赖关系异常(DDA)、数据对象定义冲突(DDC)以及数据约束条件冲突(DCC).

表 1　data-flow 验证方法在 CCTV 物资管理系统上的实验结果

检测项目	实际数量	检测数量	异常检出率/%
DDA 数量	10	0	0
DDC 数量	21	19	90.5
DCC 数量	12	0	0
异常总数	43	19	44.2

下面便对 DPGT 方法实现的模型验证器在系统设计中进行了试用. 表 2 是 DPGT 方法的实验结果.

表 2　DPGT 在 CCTV 物资管理系统上的实验结果

检测项目	实际数量	检测数量	异常检出率/%
DDA 数量	10	9	90.0
DDC 数量	21	19	90.5
DCC 数量	12	12	100.0
异常总数	43	40	93.0

DPGT 方法对于这 3 种异常均有较高的检出率,即其在检测面向业务流程的数据模型异常方面是非常有效的. 这些异常有很多是隐蔽的,在模型设计中经常被忽略. DPGT 方法的应用减少了这种错误,使得系统模型更加规范,增强了系统健壮性,减轻了系统测试和修改的工作.

有些异常 DPGT 方法并没有检出,但是可以通过系统测试发现,经过分析仍然划归流程-数据一致性导致的数据模型异常. 异常的检出率并未达到 100%,原因在于 CCTV 物资管理系统流程中包含一些现阶段的 DPGT 算法尚未完全支持的高级流程模式[18]和即席流程[17]定义.

7　总结和展望

随着流程驱动的建模方式的广泛应用,数据模型面向业务流程的异常也越来越频繁地出现,而并没有现成方法来自动化地检测这些异常.

本文提出并分析了基于流程模型的数据模型异常并将这些异常分成数据依赖关系异常、数据对象定义冲突、数据约束条件冲突三类. 为了实现自动化地对这些异常进行检测,本文提出了 DP-Graph 模型来联系数据模型和流程模型并基于 DP-Graph 提出了 DPGT 异常检测方法. 文章给出了 DPGT 算法在 CCTV 物资管理系统项目中应该的实验结果,验证了 DPGT 算法检测面向流程模型的数据模型异常的高检出率,证明了该方法在实际模型构建中的实用性.

在以后的工作中,会增加 DPGT 检测算法对即席流程[15]的支持.

参　考　文　献

[1] van der Aalst W, ter Hofstede A, Weske M. Business process management: A survey//Proceedings of the 2003 International Conference on Business Process Management (BPM 2003). Eindhoven, The Netherlands, 2003: 1-12

[2] OMG: Business process modeling notation (BPMN) specification. final adopted specification. Technical Report: Object Management Group (OMG) 06-02-01, 2006

[3] Moody D, Shanks G. Improving the quality of data models: Empirical validation of a quality management framework. Information Systems, 2003, 28(6): 619-650

[4] Moody D, Shanks G. What makes a good data model? Evaluating the quality of entity relationship models//Proceedings of the 13th International Conference on the Entity Relationship Approach. Manchester, England, 1994. LNCS 881. 1994: 94-111

[5] van der Aalst W. Verification of workflow nets//Proceedings of the 18th International Conference on Application and Theory of Petri Nets (ICATPN 97). Toulouse, France, 1997. LNCS 1248. 1997: 407-426

[6] van der Aalst W. Challenges in business process management: Verification of business processes using Petri nets. Bulletin of the European Association for Theoretical Computer Science, 2003, 80: 174-198

[7] van der Aalst W, ter Hofstede A. YAWL: Yet another workflow language. Information Systems, 2005, 30(4): 245-275

[8] Eshuis H. Semantics and verification of UML activity diagrams for workflow modeling [Ph. D. dissertation]. University of Twente, Enschede, The Netherlands, 2002

[9] Wohed P, van der Aalst W, Dumas M et al. Pattern-based analysis of the control-flow perspective of UML activity diagrams//Proceedings of the 24th International Conference on Conceptual Modelling. Klagenfurt, Austria. LNCS 3716. 2005: 63-78

[10] Crampton J. An algebraic approach to the analysis of constrained workflow systems//Proceedings of the 3rd Workshop on Foundations of Computer Security. Turku, Finland, 3004: 61-74

[11] van Dongen B, Jansen-Vullers M, Verbeek H et al. Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. Computers in Industry, 2007, 58(6): 578-601

[12] Wynn M, Verbeek H, van der Aalst W, ter Hofstede A. Business process verification: Finally a reality! Business Process Management Journal, 2009, 15(1): 74-92

[13] van der Aalst W. Business process management demystified: A tutorial on models, systems and standards for workflow management//Lectures on Concurrency and Petri Nets, Eichstätt, Germany. LNCS 3098. 2004: 1-65

[14] Sun S, Nunamaker J, Zhao J, Sheng O. Formulating the data-flow perspective for business process management. Information Systems Research, 2006, 17(4): 374-391

[15] Sadiq S, Orlowska M, Sadiq W, Foulger C. Dataflow and validation in worflow modeling//Proceedings of the 15th Australasian Database Conference. Dunedin, New Zealand, 2004: 207-214

[16] Russell N, ter Hofstede A, Edmond D. Workflow data patterns. Queensland University of Technology, Brisbane: QUT Technical Report FIT-TR-2004-01, 2004

[17] Zhou Bin, Li Hong-Yan, Wang Lei, Zhang Hua-Qiang, Li Mei-Mei. Pase: A prototype for Ad-hoc process-aware information system declaratively constructing environment//Proceedings of the Web-Age Information Management. Zhangjiajie, China, 2008: 473-474

[18] Russell N, ter Hofstede A H M, van der Aalst W M P et al. Workflow control-flow patterns: A revised view. BPMcenter. org, BPM Center Report: BPM-06-22, 2006

LIU Zhi-Qiang, born in 1987, master candidate. His main research interests include information system auto-construction and business process.

LI Hong-Yan, born in 1970, professor. Her research interests include data management and data mining.

WANG Lei, born in 1983, master candidate. His main research interests include information system auto-construction and business process.

QU Qiang, born in 1985, master candidate. His main research interests include query processing in database community and large-scale graph mining.

Background

The data model anomalies detection is a problem in the verification of the data model and business process model in the information system construction. However, the existing methods for anomalies detection only focused on either the features of the data model or the business process model. When detecting the data model anomalies, they did not take business process model into account. Since the process model in the information system construction is more and more important in the process-driven methods, the process model must be considered when detecting anomalies of the data model.

This paper proposed a model of Data-process Graph (DP-graph) and gave a method called DPGT based on DP-graph to detect those anomalies. And the experiment results show the high detection rate of the anomalies of data model for business process model.

This work was supported by the Natural Science Foundation of China (NSFC) program titled "Process-Aware Web Information System Constructing Environment (PASE)" under grant number 60673113. The ideas of PASE were published in WAIM 2008. An early version of this work was published in the International Joint Conference on Artificial Intelligence 2009. That work was focus on the verification between the business process model and data model. This work was mainly focus on the data model. It detected the anomalies of data model for the business process. And we developed a tool based on the DPGT method for data model anomalies detection which discovers the anomalies. This tool is implemented in the PASE so as to provide diagnosis information to help designers to improve the data models in the information system construction process.