一种将需求模型转换为软件体系结构模型的方法

谢仲文[1,2] 李晓燕[3] 李 彤[1,2] 代 飞[1,2] 于 倩[1,2] 张 璇[1,2]
(云南大学软件学院 昆明 650091)[1] (云南省软件工程重点实验室 昆明 650091)[2]
(昆明医科大学外语部 昆明 650500)[3]

摘 要 需求模型到软件体系结构(SA)模型的转换是软件工程领域的一个研究热点。针对DERM所建立的ACP风格的需求模型,以扩展Petri网风格的SA模型为模型转换的目标,以行为映射为模型转换的依据,提出了一种将需求模型转换为SA模型的方法。首先,讨论了模型转换的整体思路;其次,将行为特征模型中的各个结点转换为SA模型中的构件和连接件,并提出了相应的转换规则;再次,讨论了属性特征模型中主动属性特征的转换,并提出了子系统划分的策略;最后,通过案例研究展示了该方法在从需求模型到SA模型的转换中的有效性。

关键词 需求模型,软件体系结构,通信进程代数,Petri网,行为
中图法分类号 TP311.5 **文献标识码** A

Approach of Transformation from Requirements Models to Software Architecture Models

XIE Zhong-wen[1,2] LI Xiao-yan[3] LI Tong[1,2] DAI Fei[1,2] YU Qian[1,2] ZHANG Xuan[1,2]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(Key Laboratory in Software Engineering of Yunnan Province, Kunming 650091, China)[2]
(Department of Foreign Language, Kunming Medical University, Kunming 650500, China)[3]

Abstract The transformation from requirements models to software architecture (SA) models is a hot topic in software engineering. Based on the ACP (Algebra of Communicating Processes) style requirements models set up by DERM (Dynamic Evolution Oriented Requirements Meta-model), an approach of the transformation from requirements models to software architecture models was proposed. The approach takes the Petri nets style SA models as the objective of transformation, and the behavior-mapping as the foundation of transformation. Firstly, the framework of the transformation was discussed. Secondly, the nodes of the behavior feature models were transformed into components and connectors of the SA models, and the counterpoint of the transformation rules was brought forward. Thirdly, the transformation of active property features in property feature models was discussed, and the strategy of sub-systems partition was put forward. Finally, feasibility and effectiveness of the proposed method were exhibited through a case study.

Keywords Requirements models, Software architecture, Algebra of Communicating Processes, Petri nets, Behavior

1 引言

在软件工程领域,需求建模和软件体系结构(software architecture,简称SA)构造是软件生命周期中一脉相承的两个关键活动,对软件产品的质量起着至关重要的决定性作用。但是,在目前的软件工程理论和实践中,从需求模型到SA模型的变换却往往需要依赖于软件设计者的技能、经验和创造力,这种现状往往导致了SA模型的质量难以控制,甚至出现需求模型与SA模型之间的不一致。因此,研究从需求规约到SA模型的转换已经成为当前软件工程领域的一个研究热点和难点[1]。

现有的相关研究成果大多是基于自然语言的需求规约,而没有考虑形式化规约到SA模型的转换,因此很难在实践中得到广泛推广[1],例如:刘冬云等引入一种面向特征的映射方法,并阐述了从需求到软件体系结构的映射步骤[2];Ilic D将自然语言描述的规约按照类型分类,并提出一种将自然语言规约转化为形式化规约的方法[3];Zhang W等引入责任作为从需求模型向SA模型转化的桥梁,通过"特征—责任—构件"的映射关系来维护可追踪性[4];张俊等则通过角色的中介作用,实现了特征模型的构件化[5];Kelleher J等以UML为基础,将用例类作为模板说明需求建模规则,并提出一种基于UML的复杂关系分层结构[6]。目前,也有部分工作研究了

到稿日期:2013-07-15 返修日期:2013-10-27 本文受国家自然科学基金项目(61379032,61262024,60122025),云南省软件工程重点实验室开放基金面上项目(2012SE307,2012SE309),云南省自然科学基金项目(2012FD005)资助。

谢仲文(1982—),男,博士,讲师,CCF会员,主要研究方向为软件工程和形式化方法,E-mail:xiezw56@126.com;李晓燕(1988—),女,硕士,助教,主要研究方向为计算机英语,E-mail:348566634@qq.com(通信作者);李 彤(1963—),男,博士,教授,博士生导师,主要研究方向为软件工程和形式化方法;代 飞(1982—),男,博士,讲师,主要研究方向为软件工程和形式化方法;于 倩(1975—),女,讲师,主要研究方向为软件工程、网络工程;张 璇(1978—),女,副教授,主要研究方向为软件工程、信息安全。

从形式化规约到 SA 模型的转换，例如：Ferreira P 等提出了一种将 CSP 描述的规约自动转换成为 UML-RT 描述的 SA 模型的方法[7]；祝义等进一步基于 TCSP 规约，生成保留了形式化语义的 SA 模型，实现了从进程代数规约到精确的 SA 模型的转换[1]。

目前，随着软件工程学科的不断深入和网络技术的飞速发展，软件演化的重要性和普适性越来越强[8]。软件演化包括静态演化和动态演化，其中，软件动态演化仍面临多方面的挑战[9,10]。为了应对动态演化面临的挑战，本课题组拟提出一种面向软件动态演化的系统建模方法：该方法在需求分析、SA 建模、动态演化分析、动态演化实施等多个阶段管理和支持动态演化，以解决软件动态演化面临的挑战为导向，以需求模型为驱动，以 SA 模型为视图，以支持软件动态演化的平台为支撑，以形式化方法为保障。在整个研究框架内，本文的前期成果是已完成的面向软件动态演化的需求建模[11]。因此，在文献[11]中建立的需求模型的基础上，本文将讨论从需求模型变换生成 SA 模型，SA 模型将是后续工作进行动态演化分析和实施的视图。

梅宏等认为，从需求模型向 SA 模型的转换主要关注两个问题：1) 如何根据需求模型构建 SA 模型；2) 如何保证转换的可追踪性[12]。在文献[11]中建立的需求模型在多个特性上能有效支持动态演化，因此，在从需求模型向 SA 模型的转换过程中，对动态演化的支持特性和机制应能够在变换生成的 SA 模型中得到映射和继承。

本文第 2 节讨论模型转换的思路、目标和依据；由于文献[11]的需求模型包括行为特征模型和属性特征模型，因此第 3 节和第 4 节分别对行为特征模型和属性特征模型的转换进行讨论；第 5 节是案例研究；最后是总结与展望。

2 模型转换的思路、目标和依据

2.1 模型转换的思路

元模型是用来定义模型的工具，是建模的主要依据。为了建模面向动态演化的软件需求，文献[11]设计了一个面向动态演化的需求元模型，简称 DERM(Dynamic Evolution Oriented Requirements Meta-model)。DERM 作为需求元模型，是用来定义需求模型的工具。本文的目标是，将 DERM 定义的需求模型转换为 SA 模型。因此，模型转换将从 DERM 的各个部件展开。

DERM 以特征为基本部件，特征可以分为行为特征和属性特征。相应地，由 DERM 定义的需求模型也包含行为特征模型和属性特征模型。因此，本文在模型转换的过程中，采取分而治之的策略：先对行为特征模型进行转换，得到相应的构件及其构件之间的关系；然后依据属性特征模型，对构件之间的关系做进一步梳理和约束，并进行子系统划分和 SA 模型优化等工作。关于 DERM 及其部件的详细定义，可以参看文献[11]。

由于本文属于面向软件动态演化的系统建模方法的一个组成部分，考虑到动态演化在分析和实施时极其严格，因此需要形式化方法作为基本手段，即所建立的 SA 模型应该是形式化的。另一方面，本文转换生成的 SA 模型将是动态演化分析和实施时的视图，因此，SA 模型还应直观和易于观察。考虑到 Petri 网既具有直观的图形表示，又具有严格的数学基础，本文转换生成的 SA 模型将以 Petri 网为形式化工具，对其适当扩展，以适合 SA 模型的描述需要，保证建立的 SA 模型既能够直观表达系统的结构，又具有精确的语义。

2.2 模型转换的目标

模型转换生成的是 SA 模型，因此首先给出 SA 元模型的主要部件定义，作为模型转换的目标。

定义 1(构件) 构件是 SA 中承担计算功能的单元，用八元组 $N=(P,T;F;E,A,I,O,C)$ 表示：

(1)$P\cup T\neq\emptyset$ 且 $P\cap T=\emptyset$，$F\subseteq(P\times T)\cup(T\times P)$，$dom(F)\cup ran(F)=P\cup T$；

(2)$E\subseteq T$；$A\subseteq P$；$C\subseteq T$；

(3)N 存在一个输入源(简称“源”)$i\in P\cup T$，使得 $^\bullet i=\emptyset$(i 的前集为空)；同时，N 存在一个输出槽(简称“槽”)$o\in P\cup T$，使得 $o^\bullet=\emptyset$(i 的后集为空)；

(4)任意一个节点 $x\in P\cup T$，都属于从 i 到 o 的一个路径上；

(5)$(P,T;F;E,A,C)$ 称为构件的网结构，描述了构件的实现；

(6)I 是 N 的输入源 i，表示构件的输入接口，O 是 N 的输出槽 o，表示构件的输出接口，C 是 N 的交互变迁集，表示构件的端口集，接口和端口统称为构件的接触点(*touch*)。

其中：P 是 N 的库所集，T 是 N 的变迁集；F 是 N 的有向弧集，称为 N 的流关系；E 是易变变迁集(变量变迁集)，A 是主动库所集，C 是交互变迁集；$dom(F)=\{x\in P\cup T\mid \exists y\in P\cup T,(x,y)\in F\}$；$ran(F)=\{x\in P\cup T\mid \exists y\in P\cup T,(y,x)\in F\}$。

需要注意的是：构件定义中区分了接口和端口，与文献[11]中区分组合和复合相对应，接口对应用于组合，端口对应用于复合。

定义 2(连接件) 连接件是 SA 中承担连接和交互功能的单元，用一个四元组 $Con=(Type,C_f,C_b,Q)$ 表示：

(1)$Type$ 表示连接件的类型，其取值范围为 $\{C_T,C_P,C_{TP},C_{PT}\}\cup C^*$，其中 C^* 是自定义的连接件类型；

(2)C_f 表示连接件的源，C_b 表示连接件的槽，C_f 和 C_b 统称为连接件的角色(*role*)；

(3)Q 是连接件的实现，当连接件是自定义的连接件时，Q 为其对应的连接协议；当连接件是基本类型时，Q 为空集。

其中基本连接件类型含义为：C_P 表示库所熔合连接件，C_T 表示变迁熔合连接件，C_{PT} 表示正向弧添加连接件，C_{TP} 表示逆向弧添加连接件。

定义 3(连接关系) 连接关系确定了 SA 中构件和连接件之间的关联关系，描述了 SA 的拓扑结构，用二元组 $l=(Com.touch,Con.role)$ 表示，其中：

(1)$Com.touch$ 表示构件 Com 的一个接触点；

(2)$Con.role$ 表示连接件 Con 的一个角色；

(3)连接关系表明构件的接触点 $Com.touch$ 和连接件的角色 $Con.role$ 相连接。

定义 4(SA) 接下来以递归形式定义 SA：

(1)一个构件是一个 SA；

(2)四元组 (COM,CON,L,C) 是一个 SA，满足条件：1)

COM 是一个构件的集合，其中至少包含一个构件；2）CON 是一个连接件的集合，其中任意一个连接件的两个角色都参与连接关系；3）L 是一个连接关系的集合；4）C 是 SA 的端口的集合；5）以 COM 中构件为顶点、CON 中连接件为边，依据 L 建立连接关系，得到的是一个连通图；

（3）四元组（SA，CON，L，C）是一个 SA，满足条件：1）SA 是一个 SA 的集合，其中至少包含一个 SA；2）CON、L、C 同上；3）以 SA 中元素为顶点，CON 中连接件为边，依据 L 建立连接关系，得到的是一个连通图。

SA 的递归定义使得 SA 的构造具有层次性。

由于关注从需求模型到 SA 模型的转换，因此只给出以上在转换过程中必须的部件，作为转换目标。关于 SA 元模型的设计思路、其它部件及 SA 的分析等内容，我们将另文阐述。

2.3 模型转换的依据

为了保证从需求模型到 SA 模型之间转换的合理性，必须找到模型变换的依据。需求模型描述的是问题空间的知识，SA 模型描述的是解空间的知识，这是两者之间的转换的难点。不过，问题空间有问题空间的行为，解空间有解空间的行为。因此，本文以行为为纽带，以行为映射为依据，以保证模型变换的合理性和可追踪性。

由于需求模型是以 ACP 风格的进程代数为基础来描述的，而 SA 模型则建立在扩展的 Petri 基础之上，因此分别定义基于 ACP 的问题空间行为和基于 Petri 网的解空间行为。

定义 5（问题空间行为） 对于需求模型 M，其行为特征模型 M_B 的每个结点对应着一个进程，每个叶子对应着一个事件。问题空间行为是一个三元组 $d=(p,t,p')$，其中：（1）p 和 p' 表示进程，分别对应 M_B 的一个结点，表示行为的状态；（2）t 表示导致状态发生变化的事件，$t\in Leaf$，$Leaf$ 是叶子集合。

特殊的问题空间行为 $(p,t,\surd)$ 表示在状态 p 下执行事件 t 后，该需求的行为成功地终止。

定义 6（解空间行为） 对于构件 Com，解空间行为是一个三元组 $b=(s,t,s')$，其中：（1）s 和 s' 表示构件的局部状态，其实质是构件 Petri 网中变迁 t 的外延格局，即是一个映射 $N:P_t\rightarrow\{0,1\}$，其中 $P_t={}^{\cdot}t\cup t^{\cdot}$；（2）$t$ 表示导致构件状态发生变化的变迁，$t\in T$，T 是构件的变迁集。

在解空间行为中，变迁代表着 SA 模型的一种行为能力，正是这种能力，使得需求模型中对应的问题空间行为描述的事件可以发生，从而使得解空间的该行为满足问题空间对应的行为要求。因此，在将需求模型转换为 SA 模型的过程中，若能将需求模型的每一个事件变换为 SA 模型的一个对应的变迁，并且使 SA 模型中的变迁之间的关系与需求模型中事件之间的关系也保持一致，则 SA 模型能够满足需求模型的要求。

定义 7（行为映射） 行为映射是一个函数 $f:D\rightarrow B$，f 是一个一一映射，其中集合 D 是由问题空间行为组成的集合，B 是由解空间行为组成的集合。

行为映射的定义为模型转换提供了依据。

3 行为特征模型的转换

在文献[11]中，行为特征模型被组织成一个树形，因此，接下来，自底向上地、从最底层的叶子结点开始，对行为特征模型进行转换。

3.1 叶子的转换

叶子结点集合是原子计算行为特征的集合。其中，每个原子计算行为特征可以是一个行为特征常量，也可以是一个行为特征变量。

由于特征常量指代稳定的行为特征，而特征变量指代易变的行为特征，因此特征变量往往需要进一步被替换或细化。考虑到构件模型中专门区分出易变变迁集（对应特征变量集），即定义 1 中八元组 $N=(P,T;F;E,A,I,O,C)$ 中的 E 集合，接下来分别给出行为特征常量和变量的转换规则。

规则 1（行为特征常量的转换） 行为特征常量 a，变换为一个基本构件 $Com_a=(P,T;F;E,A,I,O,C)$，其中：（1）$P=\emptyset$；（2）$T=\{a\}$；（3）$F=\emptyset$；（4）$E=\emptyset$；（5）$A=\emptyset$；（6）$I=\{a\}$；（7）$O=\{a\}$；（8）$C=\emptyset$。

规则 1 的转换如图 1 所示，行为特征常量 a 被转换成为对应的稳定变迁 a，封装成为构件之后，变迁 a 既是构件的输入接口，也是输出接口。图中虚线表示变迁被提升为构件的接口。

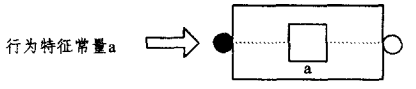

图 1 行为特征常量的转换

规则 2（原子特征变量的转换） 行为特征变量 x，变换为一个基本构件 $Com_x=(P,T;F;E,A,I,O,C)$，其中：（1）$P=\emptyset$；（2）$T=\{x\}$；（3）$F=\emptyset$；（4）$E=\{x\}$；（5）$A=\emptyset$；（6）$I=\{x\}$；（7）$O=\{x\}$；（8）$C=\emptyset$。

规则 2 的转换如图 2 所示，行为特征变量 x 被转换成为对应的易变变迁 x，需注意的是，易变变迁用黑盒变迁表示，以区别于表示稳定变迁的白盒。黑盒变迁往往需要进一步被替换或细化。

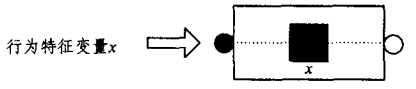

图 2 行为特征变量的转换

3.2 组合结点的转换

文献[11]中的组合与复合具有不同的含义：首先，组合对应 $\cdot$，$+$，$*$ 3 个算子，而复合对应 $\|_{[C]}$，$\llcorner_{[C]}$，$\sigma_{[H]}$ 3 个算子。其次，在进行组合时，可以把组合对象当成黑盒，只需关注其接口类型；而在进行复合时，由于涉及内部动作的交互，因此必须在明确复合对象的内部构造的基础上，才可以进行复合。

3.2.1 接口类型的转换

本节先考虑组合结点的转换，由于组合结点通过连接件组合多个构件，因此，连接件的类型和组合构件的接口类型必须匹配。所以在某些情况下，需要转换构件的接口类型，以保证连接的匹配。

规则 3（接口类型转换） 一个构件可以通过引入虚库所或虚变迁的方式转换构件的接口类型，虚库所或虚变迁是指没有实际意义的结点，它只能作为构件的输入源的前提或输出槽的后提被添加进网结构，从而成为构件的输入或输出接口。

由于接口类型转换的方式有多种，而且都比较简单，因

此，这里仅给出一个接口类型转换的例子。

例 1　将规则 1 中的构件 $Com_a=(P,T;F;E,A,I,O,C)$，通过引入虚库所，使其输入接口和输出接口都是库所类型，即进行接口类型转换，转换为构件 $Com_a'=(P',T';F';E',A',I',O',C')$，其中：(1)$P'=\{v_1,v_2\}$；(2)$T'=\{a\}$；(3)$F'=\{(v_1,a),(a,v_2)\}$；(4)$E'=\emptyset$；(5)$A'=\emptyset$；(6)$I'=\{v_1\}$；(7)$O'=\{v_2\}$；(8)$C'=\emptyset$。

在图 3 中通过引入虚库所 v_1 和 v_2，使得构件的接口类型由原来的变迁类型变换为库所类型，虚结点用虚线的库所或变迁表示。

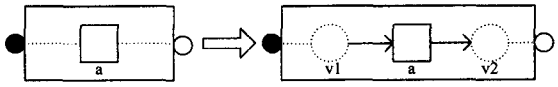

图 3　接口类型转换的一个例子

3.2.2　顺序组合结点的转换

组合结点对应着一个组合算子，其子树是参与组合的子构件。首先考虑顺序组合算子。

两个行为特征 α 和 β，通过顺序组合算子 $\cdot$，组合成一个组合特征，用二元组 $\langle id,\alpha\cdot\beta\rangle$ 表示，组合特征 $\langle id,\alpha\cdot\beta\rangle$ 先按特征 α 规定的行为操作，α 成功终止后再按 β 规定的行为操作。α 和 β 分别对应成为子构件 Com_α 和 Com_β，将它们顺序组合成为一个构件，如图 4 所示，其中库所 p_{23} 代表库所 p_2 和 p_3 熔合之后的库所，子构件的内部结构并不影响顺序组合，因此用星形结构抽象地表示。

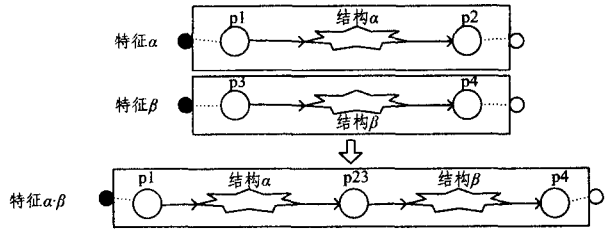

图 4　顺序组合转换成一个构件

规则 4(顺序组合转换成构件)　特征的顺序组合 $\alpha\cdot\beta$ 对应成两个子构件 Com_α 和 Com_β 的顺序组合，组合成为一个父构件 $Com_{\alpha\cdot\beta}=(P,T;F;E,A,I,O,C)$，$F_*$ 对应熔合后的库所 p_{23} 的弧关系，$F_\#$ 对应被熔合库所 p_2 和 p_3 的弧关系，其中：(1)$P=P_\alpha\cup P_\beta\cup\{p_{23}\}-\{p_2,p_3\}$；(2)$T=T_\alpha\cup T_\beta$；(3)$F=F_\alpha\cup F_\beta\cup F_*-F_\#$；(4)$E=E_\alpha\cup E_\beta$；(5)$A=A_\alpha\cup A_\beta$；(6)$I=I_\alpha$；(7)$O=O_\beta$；(8)$C=C_\alpha\cup C_\beta$。

顺序组合也可以转换成为 SA，如图 5 所示，图中 C_P 是库所熔合连接件。

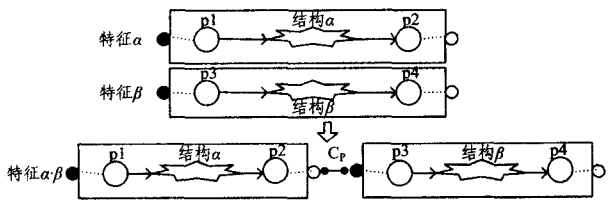

图 5　顺序组合转换成 SA

规则 5(顺序组合转换成 SA)　特征的顺序组合 $\alpha\cdot\beta$ 对应成两个构件 Com_α 和 Com_β 的顺序组合，组合成为 $SA_{\alpha\cdot\beta}=(COM,CON,L,C)$，其中：(1) $COM=\{Com_\alpha,Com_\beta\}$；(2) $CON=\{C_P\}$；(3) $L=\{(Com_\alpha.\,o,C_P.\,C_f),(Com_\beta.\,i,C_P.\,C_b)\}$；(4) $C=Com_\alpha\cup Com_\beta$；其中：$Com_\alpha.\,o$ 和 $Com_\beta.\,i$ 分别是 Com_α 的输出接口和 Com_β 的输入接口，$C_P.\,C_f$ 和 $C_P.\,C_b$ 是连接件 C_P 的角色。

需要注意两点：(1) 连接件 C_P 的两个角色虽然从功能上看是对称的，但是还是应该明确区分，否则连接的时候有可能把两个接口连接到同一个角色；(2) 如果子构件的接口类型不同，连接时有可能需要使用其它类型的连接件(如 C_{TP}，C_{PT})。

3.2.3　选择组合结点的转换

两个行为特征 α 和 β，通过选择组合算子 $+$，组合成一个组合特征，用二元组 $\langle id,\alpha+\beta\rangle$ 表示，组合特征 $\langle id,\alpha+\beta\rangle$ 在特征 α 和 β 中选择其中之一，然后按所选特征规定的行为操作。α 和 β 分别对应成为子构件 Com_α 和 Com_β，将它们选择组合成为一个父构件，如图 6 所示，其中库所 p_{13} 代表库所 p_1 和 p_3 熔合之后的库所，库所 p_{24} 代表库所 p_2 和 p_4 熔合之后的库所。

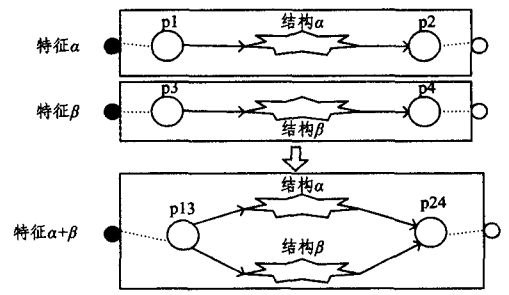

图 6　选择组合转换成一个构件

规则 6(选择组合转换成构件)　特征的选择组合 $\alpha+\beta$ 对应两个子构件 Com_α 和 Com_β 的选择组合，组合成为一个父构件 $Com_{\alpha+\beta}=(P,T;F;E,A,I,O,C)$，其中：(1)$P=P_\alpha\cup P_\beta\cup\{p_{13},p_{24}\}-\{p_1,p_2,p_3,p_4\}$；(2)$T=T_\alpha\cup T_\beta$；(3)$F=F_\alpha\cup F_\beta\cup F_*-F_\#$；(4)$E=E_\alpha\cup E_\beta$；(5)$A=A_\alpha\cup A_\beta$；(6)$I=\{p_{13}\}$；(7)$O=\{p_{24}\}$；(8)$C=C_\alpha\cup C_\beta$。

选择组合也可以转换成为 SA，如图 7 所示。

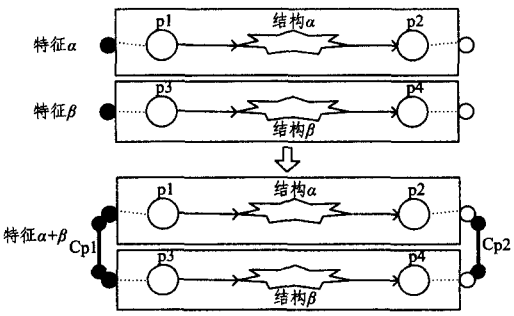

图 7　选择组合转换成 SA

规则 7(选择组合转换成 SA)　特征的选择组合 $\alpha+\beta$ 对应成两个构件 Com_α 和 Com_β 的选择组合，组合成为 $SA_{\alpha+\beta}=(COM,CON,L,C)$，其中：(1)$COM=\{Com_\alpha,Com_\beta\}$；(2)$CON=\{C_{P1},C_{P2}\}$；(3)$L=\{(Com_\alpha.\,i,C_{P1}.\,C_f),(Com_\beta.\,i,C_{P1}.\,C_b),(Com_\alpha.\,o,C_{P2}.\,C_f),(Com_\beta.\,o,C_{P2}.\,C_b)\}$；(4)$C=Com_\alpha\cup Com_\beta$。

规则 7 中用到两个库所熔合连接件。

3.2.4　迭代组合结点的转换

对于行为特征 α，通过迭代组合算子 $*$，组合成一个组合特征，用二元组 $\langle id,(\alpha)^*\rangle$ 表示，组合特征 $\langle id,(\alpha)^*\rangle$ 重复特征 α 规定的行为操作 n 次($n\geqslant 1$)，然后终止。α 对应的子构件为 Com_α，将其迭代组合成为一个父构件，如图 8 所示。

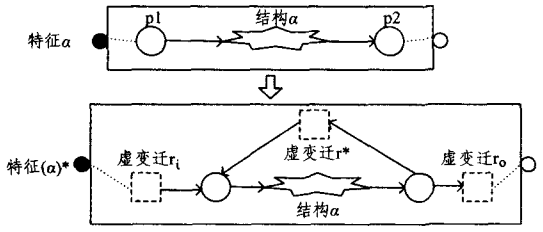

图 8　迭代组合转换成一个构件

规则 8(迭代组合转换成新构件)　特征的迭代组合 $(\alpha)^*$,对应子构件 Com_α 的迭代组合,组合成一个父构件 $Com_{(\alpha)^*}=(P,T;F;E,A,I,O,C)$,$F_*$ 为引入虚变迁增加的弧关系,即 $F_*=\{(r_i,p_1),(p_2,r_o),(p_2,r^*),(r^*,p_1)\}$,其中:(1)$P=P_\alpha$;(2)$T=T_\alpha\cup\{r^*,r_i,r_o\}$;(3)$F=F_\alpha\cup F_*$;(4)$E=E_\alpha$;(5)$A=A_\alpha$;(6)$I=\{r_i\}$;(7)$O=\{r_o\}$;(8)$C=C_\alpha$。

规则 8 中引入 3 个虚变迁和相应的弧,r^* 是迭代虚变迁,r_i 是输入虚变迁,r_o 是输出虚变迁(若不引入 r_i 和 r_o,得到的结构不满足构件定义的要求)。

3.3　复合结点的转换

复合结点对应 $\|_{[C]}$,$\llcorner_{[C]}$,$\sigma_{[H]}$ 3 个算子其中的一个,其子树是参与复合的构件,下面分别讨论。

3.3.1　并行复合结点的转换

两个行为特征 α 和 β,通过并行复合算子 $\|_{[C]}$,复合成一个复合特征,用二元组 $\langle id,\alpha\|_{[C]}\beta\rangle$ 表示,参与复合的两个特征 α 和 β,在执行到处于 C 规定的交互动作集之中的元素时,必须按交互动作协同完成任务;否则,可按 α 与 β 各自的定义方式,并行地执行。α 和 β 分别对应构件 Com_α 和 Com_β,设其交互动作集 C 中有一个元素 $\langle id,u,v\rangle$,其中:u 和 v 分别为构件 Com_α 和 Com_β 内部结构的变迁。构件 Com_α 和 Com_β 复合成为体系结构,如图 9 所示。

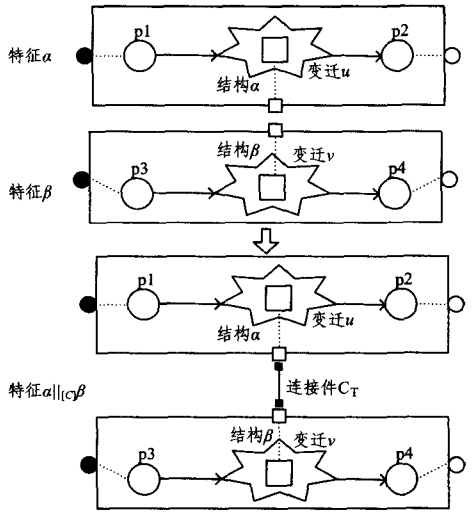

图 9　并行复合转换成 SA

规则 9(并行复合转换成 SA)　特征的并行复合 $\alpha\|_{[C]}\beta$ 对应成两个构件 Com_α 和 Com_β 的并行复合,$C=\{\langle id,u,v\rangle\}$ 是交互动作集,复合成为 $SA_{\alpha\|_{[C]}\beta}=(COM,CON,L,C)$,其中:(1)$COM=\{Com_\alpha,Com_\beta\}$;(2)$CON=\{C_T\}$;(3)$L=\{(Com_\alpha.c_1,C_T.C_f),(Com_\beta.c_2,C_T.C_b)\}$;(4)$C=Com_\alpha\cup Com_\beta$。其中 $Com_\alpha.c_1$ 是构件 Com_α 的端口 c_1,$Com_\beta.c_2$ 是构件 Com_β 的端口 c_2。

在图 9 中构件 Com_α 的结构 α 内部的变迁 u 被提升为端口 c_1,构件 Com_β 的结构 β 内部的变迁 v 被提升为端口 c_2,端口 c_1 和 c_2 用连接件 C_T 连接。

3.3.2　左并行复合结点的转换

两个行为特征 α 和 β,经左并行复合算子 $\llcorner_{[C]}$,复合成一个复合特征,用二元组 $\langle id,\alpha\llcorner_{[C]}\beta\rangle$ 表示,左并行复合规定必须先执行左元 α 的第一个动作,因此右元 β 实际上是受 α 的第一个动作控制的。因此左元 α 常分解为:$\alpha=x\cdot\alpha'$。这样,即要求首先执行左元的第一个动作 x,且 x 不参与交互,然后按并行算子 $\|_{[C]}$ 继续执行 $\alpha\llcorner_{[C]}\beta$ 剩余的部分(即 $\alpha'\|_{[C]}\beta$)。可见,$\alpha\llcorner_{[C]}\beta=x\cdot(\alpha'\|_{[C]}\beta)$,因此,左并行复合结点的转换可以按此顺序完成:(1)首先根据规则 9 完成 $\alpha'\|_{[C]}\beta$ 的转换;(2)根据规则 4 或规则 5 完成顺序组合的转换。

3.3.3　封装复合结点的转换

对于一个行为特征 $\alpha\|_{[C]}\beta$(不考虑 $\alpha\llcorner_{[C]}\beta$,因其可变换为 $\alpha'\|_{[C]}\beta$),对其进行封装复合后得到一个新的行为特征,用二元组 $\langle id,\sigma_{[H]}(\alpha\|_{[C]}\beta)\rangle$ 表示;H 是一个只含交互动作的原子计算行为特征的集合,表明 H 集合中的元素只在复合行为特征的内部发生交互,封装的外部对集合 H 中的元素不可见。

封装复合的结果是 H 中的端口从端口集中被去除,无法参与到高层的交互之中,同时降低了高层的复杂度。封装复合的转换如图 10 所示,其中 $H=\{u,v\}$,图中有色的端口表示被封装于内部。

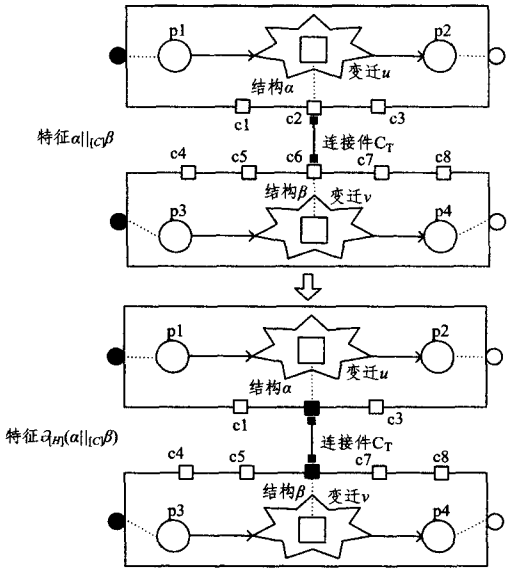

图 10　封装复合转换成 SA

规则 10(封装复合转换成 SA)　特征的封装复合 $\partial_{[H]}(\alpha\|_{[C]}\beta)$ 对应子体系结构 $SA_{\alpha\|_{[C]}\beta}$ 的封装复合,$H=\{u,v\}$ 是封装动作集,复合成为 $SA_{\partial[H](\alpha\|_{[C]}\beta)}=(COM,CON,L,C)$,其中:(1)$COM=SA_{\alpha\|_{[C]}\beta}.COM$;(2)$CON=SA_{\alpha\|_{[C]}\beta}.CON$;(3)$L=SA_{\alpha\|_{[C]}\beta}.L$;(4)$C=SA_{\alpha\|_{[C]}\beta}.C-\{c_u,c_v\}$。其中 c_u 是 u 对应的端口,c_v 是 v 对应的端口。

在图 10 中 c_u 和 c_v 分别对应 c_2 和 c_6,因此,封装复合之后的端口集不再包含 c_2 和 c_6,即 c_2 和 c_6 无法再参与高层交互(即无法再被高层连接件所连接),而其它端口(图中 c_1,c_3,c_4,c_5,c_7,c_8)既可以参加内部交互也可参加高层交互。

4 属性特征模型的转换

4.1 主动属性特征的转换

在文献[11]的需求模型中，既包含行为特征，也包含属性特征。为了支持动态演化的行为相关性分析，文献[11]中引入了一个重要的属性特征——主动属性特征，用于区分主动行为特征和被动行为特征。主动行为特征是需求模型中引起行为相关性的源头元素，它们主动地创建任务或者接受系统外部提交给系统的任务。在定义 1 的构件定义中，八元组 $N=(P,T;F;E,A,I,O,C)$ 的 A 集合用于区别主动元素和被动元素，A 是 P 的一个子集，即属于库所集的一部分。因此，库所集也相应地被分为两类：灰盒库所和白盒库所。灰盒库所是系统与外界的临界点，外界可以向灰盒库所中加入托肯，分配给系统任务；白盒库所是系统内部的元素，只能由系统中与之相连的变迁来改变其状态。

主动属性特征作用域中的叶子对应的结点，对应主动态原子行为特征，在转换的时候，被转换为一个前集库所为灰盒库所的变迁；非主动属性特征作用域中的叶子对应的结点，对应被动原子行为特征，则无需对其前集库所进行着色。

规则 11(主动库所着色转换)　对于构件 $Com=(P,T;F;E,A,I,O,C)$，集合 L 对应主动属性特征作用域中的叶子对应的结点，即 $L\subseteq T$，对其主动库所着色之后构件转换为 $Com'=(P',T';F';E',A',I',O',C')$，其中：(1)$P'=P$；(2)$T'=T$；(3)$F'=F$；(4)$E'=E$；(5)$A'=A\cup^{\cdot}L$；(6)$I'=I$；(7)$O'=O$；(8)$C'=C$。

在转换时需注意一种特殊情况：$L\cap I\neq\emptyset$，此时构件的输入接口也是主动属性特征作用域中的叶子对应的变迁。由于输入接口没有前集，因此，需先对该构件进行接口类型转换，引入一个虚库所，将该构件的输入接口转换为库所类型，然后再对包括该虚库所在内的 L 中变迁的前集进行着色。

4.2 根据属性特征的依赖和互斥关系划分子系统

属性特征用于约束或限制系统的服务或行为。对于一个目标系统而言，各个属性特征之间并非孤立地存在着，属性特征之间包含着依赖和互斥的关系。由于属性特征通过其作用域，指定其在行为特征模型中的作用范围，从而将行为特征模型和属性特征模型统一为需求模型，因此各个不同的属性特征，其作用范围之间也会有多种关系，如包含关系、交叠关系、分离关系等。

划分子系统是增加 SA 的构造性、降低其复杂性的有效手段。在工程实践中，软件设计人员往往从系统处理的业务出发，基于对业务的理解划分子系统，如销售子系统、生产子系统、财务子系统等。实际上，子系统这个名词应该是针对计算机系统而非业务系统而言的，划分子系统最重要的依据就是依赖关系[13]。

因此，本文根据属性特征的依赖和互斥关系，提出划分子系统的两条策略：(1) 具有依赖关系的属性特征的作用域，将其对应的构件和连接件尽量划分在同一个子系统内；(2) 具有互斥关系的属性特征的作用域，将其对应的构件和连接件尽量划分在不同的子系统内。

需要说明的是，以上两条策略只是划分子系统时的宏观建议，与之前的转换规则的机械转换不同。因此，不同的 SA 设计者应用以上策略会可能得到不同的子系统划分结果。

此外，值得一提的是，我们在文献[11]中提出了需求模型的规范化方法，亦即将需求模型从第一范式规范化到第三范式。规范化的需求模型满足依赖一致性和互斥一致性，从而使得本文提出的子系统划分策略更容易操作，划分的结果结构性更好。

5 案例研究

5.1 案例说明

为了说明本文提出的方法之可行性，将以文献[11]的案例为基础进行案例建模和分析。该案例是一个简化的网上银行系统，背景如下：某市商业银行为加强自身的竞争力，准备推出其网上银行系统。要求系统实现用户信息管理、余额查询、历史记录查询、网上转账等功能；对于转账功能，要求执行转账之前，系统自动向客户手机发送验证码，验证成功后才执行转账；此外，目前银行正与该市电力营销系统商讨通过网上银行代收电费事宜，但尚未确认和签约。文献[11]已建立了该案例的需求模型，接下来，给出模型转换过程示例。

5.2 模型转换过程示例

作为示例，有必要从网上银行系统的需求模型中选择典型部件，将其转换为对应的构件和连接件。接下来，对短信验证转账特征进行转换。

从文献[11]的需求建模知：短信验证转账特征对应的进程项：$(t_0\cdot(t_1+(t_2\cdot(t_3\cdot(t_4+t_5)))))\|_{[C]}((e_0+e_1)^*)$，其中 $C=\{\langle 0.01,t_2,e_0\rangle,\langle 0.02,t_3,e_1\rangle\}$。因此，该特征可对应细分为两个子构件：转账构件(对应进程项为 $t_0\cdot(t_1+(t_2\cdot(t_3\cdot(t_4+t_5)))))$和验证构件(对应进程项为$(e_0+e_1)^*$)，并通过两个子构件的并行复合和交互构成父构件或 SA(这里先视为构件，后视为 SA)。

首先，不考虑子构件的内部结构(即将子构件内部视为黑盒)，因此，构件 Com 对应的特征被表示为 $Com=Com_1\|_{[C]}Com_2$，其中 Com_1 对应进程项为 $t_0\cdot(t_1+(t_2\cdot(t_3\cdot(t_4+t_5)))))$，$Com_2$ 对应进程项为$(e_0+e_1)^*$。通过并行复合转换规则，得到短信验证转账构件的框架，如图 11 所示。

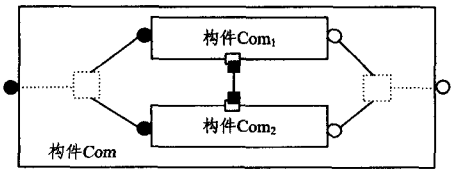

图 11　短信验证转账构件的框架

图 11 中构件 Com_1 和 Com_2 以变迁熔合连接件相连接，并通过引入的两个虚变迁组成父构件；两个虚变迁分别被提升为构件的输入和输出接口。

接下来，为了以尽量少的篇幅给出 SA 转换的示例，将短信验证转账构件视为短信验证转账 SA。首先，通过转换规则详细建模转账子构件的内部结构，即框架中的构件 Com_1 的内部结构。Com_1 对应进程项为 $t_0\cdot(t_1+(t_2\cdot(t_3\cdot(t_4+t_5)))))$，因此其转换过程为：(1)对 t_4+t_5 进行选择组合转换；(2)对 $t_3\cdot(t_4+t_5)$进行顺序组合转换；(3)对 $t_2\cdot(t_3\cdot(t_4+t_5))$再次进行顺序组合转换；(4)对 $t_1+(t_2\cdot(t_3\cdot(t_4+t_5)))$进行选择组合转换；(5)对 $t_0\cdot(t_1+(t_2\cdot(t_3\cdot(t_4+t_5)))))$再次进行顺序组合转换；(6)根据主动属性特征，$t_0$ 在其作用范围内，对其前集 p_1 进行着色。

经过以上转换步骤，得到其内部结构，如图 12 所示，由于 t_0 是主动特征，因此建模得到的库所 p_1 是主动库所；p_1 和 p_5 分别被提升为输入和输出接口，t_2 和 t_3 被提升为端口。

进一步，给出转换得到的转账构件的形式化定义，Com_1

$=(P_1,T_1;F_1;E_1,A_1,I_1,O_1,C_1)$，其中：$P_1=\{p_1,p_2,p_3,p_4,p_5\}$；$T_1=\{t_0,t_1,t_2,t_3,t_4,t_5\}$；$F_1=\{(p_1,t_0),(t_0,p_2),(p_2,t_1),(p_2,t_2),(t_1,p_5),(t_2,p_3),(p_3,t_3),(t_3,p_4),(p_4,t_4),(p_4,t_5),(t_4,p_5),(t_5,p_5)\}$；$E_1=\emptyset$；$A_1=\{p_1\}$；$I_1=\{p_1\}$；$O_1=\{p_5\}$；$C_1=\{t_2,t_3\}$。

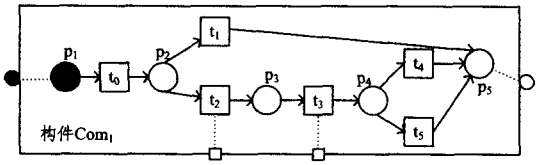

图 12　转账构件

接下来，经过多次的顺序组合转换和一次迭代组合转换，可以得到验证构件的内部结构，即框架中的构件 Com_2 的内部结构。Com_2 对应进程项为$(e_0+e_1)^*$，根据变换规则，其内部结构如图 13 所示。

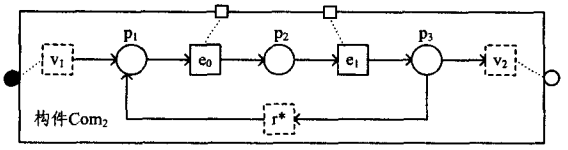

图 13　验证构件

由于 p_1 和 p_3 不符合接口的定义（分别有多于 1 个的前集和后集），所以添加虚变迁 v_1 和 v_2（若要与框架中保持接口类型匹配，还需添加两个虚库所，因本文在此将短信验证转账部件视为 SA，因此，不需要再添加虚库所）；此外，r^* 是迭代组合变换引入的虚变迁。进一步，给出验证构件的形式化定义，$Com_2=(P_2,T_2;F_2;E_2,A_2,I_2,O_2,C_2)$，其中：$P_2=\{p_1,p_2,p_3\}$；$T_2=\{e_0,e_1,v_1,v_2,r^*\}$；$F_1=\{(v_1,p_1),(p_1,e_0),(e_0,p_2),(p_2,e_1),(e_1,p_3),(p_3,v_2),(p_3,r^*),(r^*,p_1)\}$；$E_1=\emptyset$；$A_1=\emptyset$；$I_1=\{v_1\}$；$O_1=\{v_5\}$；$C_1=\{e_0,e_1\}$。

在此基础上，最后，给出短信验证转账 SA 的详细结构，如图 14 所示。

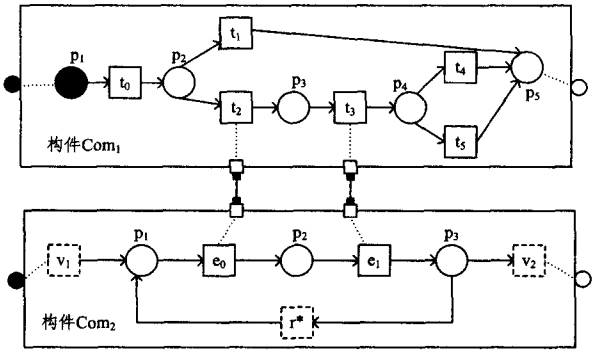

图 14　短信验证转账 SA

在图 14 中，构件 Com_1 的端口 t_2 和构件 Com_2 的端口 e_0 通过变迁熔合连接子连接，类似地，构件 Com_1 的端口 t_3 和构件 Com_2 的端口 e_1 通过变迁熔合连接子连接；两个构件并行复合成为短信验证转账 SA，记为 SA_{Com}。接下来，给出其形式化定义，$SA_{Com}=(COM,CON,L,C)$，其中：(1) $COM=\{Com_1,Com_2\}$；(2) $CON=\{C_{T1},C_{T2}\}$；(3) $L=\{(Com_1.t_2,C_{T1}.C_f),(Com_2.e_0,C_{T1}.C_b),(Com_1.t_3,C_{T2}.C_f),(Com_2.e_1,C_{T2}.C_b)\}$；(4) $C=Com_1\cup Com_2$。

限于篇幅，对于其它构件和子系统，不再详细给出转换过程。

结束语　从需求模型到 SA 模型的转换是软件工程领域的一个研究热点和难点。本文在文献[11]建立的需求模型的基础上，提出一种从需求模型到 SA 模型的转换方法。本文的工作具有以下特点：(1)本文转换的双方——需求模型和 SA 模型分别建立在 ACP 和 Petri 网的基础上，因此，不存在二义性而使得两类模型具有精确性；(2)本文的转换方法和规则以行为映射为依据，问题空间的每一种行为都通过规则映射为解空间的对应行为，保证了模型变换的可追踪性；(3)需求模型中对软件动态演化的主要机制（如用组合与复合的区分以实现计算和交互的相对隔离、稳定特征和易变特征的区分、主动需求和被动需求的区分以支持动态演化的相关性分析和管理等）在转换得到的 SA 模型中都有相应的机制进行支持，从而使得需求模型对动态演化的支持在 SA 模型中得到继承和延续。

随着软件工程研究的深入，SA 已超出对软件设计阶段的支持，逐步扩展到整个软件生命周期[12]。从需求模型到 SA 模型的转换方面的研究，其主要目标还局限于对软件设计的支持。因此，与本文密切相关的另一项工作是，针对软件动态演化面临的挑战，我们提出了一种面向动态演化的 SA 元模型 DEAM。在本文变换得到的 SA 模型基础上，DEAM 引入动态视图，使得 SA 能够反映软件系统的运行情况，将 SA 模型的作用由设计阶段延拓到动态演化分析和实施阶段，进而使 SA 模型不仅是设计阶段的蓝图，也是动态演化分析和实施时的视图；而且，DEAM 与本文变换得到的 SA 模型完全兼容。限于篇幅，这部分工作我们将另文阐述。

下一步工作是，以现有工作为基础，分析动态演化实施时的构件行为相关性。行为相关性分析作为动态演化的理论基础之一，对实施动态演化起着至关重要的作用。一方面，只有找到与待演化构件行为相关的构件集合，驱动相关构件进入静止状态，才能保证动态演化实施的可靠性；另一方面，与待演化构件行为不相关的构件必须被排除在该集合之外，这样有利于控制实施动态演化时的波及范围。本文对计算和交互的相对隔离、对主动库所和被动库所的区分等机制提供了管理相关性的手段，为行为相关性问题的研究做好了准备工作。

参 考 文 献

[1] 祝义，黄志球，周航，等. 基于进程代数规约生成软件体系结构模型的方法[J]. 计算机研究与发展，2011，48(2)：241-250

[2] 刘冬云，梅宏. 从需求到软件体系结构：一种面向特征的映射方法[J]. 北京大学学报：自然科学版，2004，40(3)：372-378

[3] Ilic D. Deriving formal specification from informal requirements [C]// Proc of the 31st Annual Int Computer Software and Application Conference. Los Alamitos, CA: IEEE Computer Society, 2007: 145-452

[4] Zhang Wei, Mei Hong, Zhao Hai-yan, et al. Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach [C]// Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems. Berlin Heidelberg: Springer, 2005: 248-263

[5] 张俊，刘淑芬，姚志林. 一种基于角色的特征模型构件化方法[J]. 电子学报，2011，39(2)：304-308

[6] Kelleher J, Simonsson M. Utilizing use case classes for requirement and traceability modeling[C]// Proc of the 17th IASTED Int Conf on Modelling and Simulation. Montreal: Acta Press, 2006: 609-617

[7] Ferreira P, Sampaio A, Mota A. Viewing CSP specifications with

UML-RT diagrams[J]. Electronic Notes in Theoretical Computer Science,2008,195:57-74
[8] Li Tong. An Approach to Modelling Software Evolution Processes[M]. Berlin:Springer-Verlag,2008
[9] 徐洪珍,曾国荪,陈波. 软件体系结构动态演化的条件超图文法及分析[J]. 软件学报,2011,22(6):1210-1223
[10] 李长云. 基于体系结构的软件动态演化研究[D]. 杭州:浙江大学,2005
[11] 谢仲文,李彤,代飞,等. 面向软件动态演化的需求建模及其模型规范化[J]. 计算机科学与探索,2012,6(6):557-576
[12] 梅宏,申峻嵘. 软件体系结构研究进展[J]. 软件学报,2006,17(6):1257-1275
[13] 谭云杰. 大象:Thinking in UML[M]. 北京:中国水利水电出版社,2009

(上接第 181 页)

表 2 经过 MATLAB 数据采集之后的部分数据

ID \ 属性	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	1	0	1	6.22	5.27	4	10.13	1368	34.67	89.41
2	0	0	0	1	1	1	7.89	5.64	5.15	9.33	1260	44.67	82.35
3	0	0	0	1	1	1	7.56	4.55	2.46	3.6	486	21.33	31.76
4	1	0	1	1	1	1	14	5.64	8.54	1.4	189	74	12.35
5	1	0	1	1	1	1	2.67	2.36	6.69	1.27	171	58	11.18
…	…	…	…	…	…	…	…	…	…	…	…	…	…
119996	1	1	1	1	1	1	7	10	6.15	5.33	720	53.33	47.06
119997	1	1	1	1	1	1	13.56	8.18	5.30	5.53	747	46	48.82
119998	1	1	1	1	1	1	5.56	5.36	1.54	5.13	693	13.33	45.29
119999	1	1	1	1	1	1	7.22	6.09	1.92	1.4	189	16.67	12.35
120000	1	1	1	1	1	1	1.67	7.27	1.77	2.67	360	15.33	23.53

通过之前算法中的数值属性的数据预处理之后的数据如表 3 所列。

表 3 数值属性数据预处理之后的部分数据

ID \ 属性	7	8	9	10	11	12	13
1	−0.6402	−0.6432	−0.6470	−0.6284	3.5000	−0.5538	−0.3874
2	−0.6427	−0.6502	−0.6518	−0.6379	3.5000	−0.5210	−0.3963
3	−0.6205	−0.6465	−0.6644	−0.6546	3.5000	−0.5019	−0.4120
4	−0.5883	−0.7547	−0.6970	−0.8390	2.8942	0.6057	−0.6201
5	−0.7484	−0.7551	−0.6584	−0.7796	3.0122	0.4877	−0.5582
…	…	…	…	…	…	…	…
119996	−0.6680	−0.6504	−0.6729	−0.6777	3.5000	−0.3971	−0.4338
119997	−0.6265	−0.6568	−0.6729	−0.6717	3.5000	−0.4440	−0.4281
119998	−0.6262	−0.6274	−0.6503	−0.6288	3.4999	−0.5796	−0.3877
119999	−0.5920	−0.6175	−0.7113	−0.7231	3.5000	−0.3794	−0.4765
120000	−0.6651	−0.5999	−0.6640	−0.6535	3.4999	−0.5063	−0.4110

为了测试算法的性能,本文将其与组态软件 WinCC 自带的 Alarm 模块的报警信息进行比较。实验结果显示当 $\beta=-0.03$时,阈值半径 $r=2.60391$,此时聚类个数为 5 个,DB 指数为 14.6865 的聚类结果最优。将 WinCC 自带的 Alarm 变量报警信息对该算法得到的结果进行检验,结果显示已知的异常检验率为 98.57%,未知的离群数据为 19 条。对未知的异常数据进行分析发现这些离群数据确实属于异常行为,例如:(1、0、1、1、1、1、14、5.64、8.54、1.4、189、74、12.35),该数据被挖掘出属于未知异常,经分析,该数据的 CO、O2、CO2、H2 比例明显不协调,但每一个属性值又都没有达到 WinCC 变量报警的阈值,因此 WinCC 变量报警没有该条记录。

实验证明,该方法在较好的空间复杂度与时间复杂度下,能有效地发现在高维属性的工业控制系统中的异常属性。

结束语 本文提出了一种基于自适应聚类的离群点挖掘方法,并将其应用到工业控制系统异常检测中,通过对聚类时半径阈值的自适应选择策略的改进来提高算法的聚类效果,并通过两个实验分别验证了该方法的有效性与性能。实验结果表明,该方法能够很好地发现工业控制系统中已知的、未知的异常,通过发现这些异常操作可以极大地提高对工业控制系统安全性的掌控。

由于工业控制系统的数据具有时间序列的特性,今后的研究工作主要是完善该算法,对工业控制系统数据所特有的时间属性进行研究,在时序性上对算法进行优化。

参 考 文 献

[1] IEC 62443-2-1 ED. 1.0 EN:2010, "Industrial communication networks-Network and system security-Part 2-1:Establishing an industrial automation and control system security program" [R]. International Electrotechnical Commission,2010
[2] 张帅. 工业控制系统安全现状与风险分析[J]. 计算机安全,2012(01):15-19
[3] Han Jia-wei, Micheline K. Data Mining: Concepts and Techniques (2nd Edition)[M]. San Francisco: Morgan Kauffmann Publishers,2006
[4] Haw kins D. Identification of Outliers[M]. London: Chapman and Hall,1980
[5] 唐成龙,王石刚. 基于数据间内在关联性的自适应模糊聚类模型[J]. 自动化学报,2010,36(11):1544-1556
[6] 薛安荣,姚林,鞠时光. 离群点挖掘方法综述[J]. 计算机科学,2008,35(11):13-17
[7] 徐翔,刘建伟,罗雄麟. 离群点挖掘研究[J]. 计算机应用研究,2009,26(1):34-39
[8] 王欣. 基于聚类和距离的大数据集离群点检测算法[J]. 制造业自动化,2010,33(4):101-104
[9] 王茜,唐锐. 基于频繁模式的离群点挖掘在入侵检测中的应用[J]. 计算机应用研究,2013,30(4):1208-1211
[10] 唐成龙,王石刚,徐威. 基于数据加权策略的模糊聚类改进算法[J]. 电子与信息学报,2010,32(6):1277-1283
[11] 杨鹏. 离群检测及其优化算法研究[D]. 重庆:重庆大学,2010
[12] 王茜,杨正宽. 一种基于加权 KNN 的大数据集下离群检测算法[J]. 计算机科学,2011,38(10):177-180
[13] Davies, David L, Bouldin, et al. A Cluster Separation Measure [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,1979,PAMI-1 (2):224-227
[14] 杨斌. 基于聚类的异常检测技术的研究[D]. 长沙:中南大学,2008
[15] 蒋盛益. 基于聚类的入侵检测算法研究[M]. 北京:科学出版社,2008:152-159