基于Petri网的软件动态演化的一致性分析

谢仲文[1,2] 明 利[1] 林 英[1,2] 秦江龙[1,2] 莫 启[1,2] 李 彤[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]

摘 要 在分析软件动态演化面临的挑战的基础上,以扩展的Petri网为主形式化工具,基于面向动态演化的SA元模型DEAM,对如何保证动态演化的一致性问题进行分析。首先,讨论了一致性分析的总体思路和策略,确定以构件作为动态演化实施和分析的基本对象;其次,从构件结构演化的视角对构件的子网类型进行分析,提出了保证结构一致性的方法;再次,从构件行为演化的视角,分别从内部和外部观察构件演化前后的行为,并通过建立模拟关系来分析和判断演化前后构件的行为是否一致;最后,通过案例研究对所提方法的可行性进行验证。

关键词 软件动态演化,软件体系结构,行为模拟,一致性保持
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.11.046

Consistency Analysis of Software Dynamic Evolution Based on Petri Net

XIE Zhong-wen[1,2] MING Li[1] LIN Ying[1,2] QIN Jiang-long[1,2] MO Qi[1,2] LI Tong[1,2]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(Key Laboratory in Software Engineering of Yunnan Province, Kunming 650091, China)[2]

Abstract On the basis of the challenges about analyzing software dynamic evolution, taking extended Petri nets as the main formalism, in the view of dynamic-evolution-oriented software architecture meta-model DEAM, the problem to be analyzed is how to ensure the consistency during dynamic evolution. Firstly, the main strategies of consistency analysis was discussed, and the components were selected as the basic analysis objects of dynamic evolution. Secondly, from the perspective of component structure evolution, the sub-net types of the components were analyzed and the methods were presented to assure the structure consistency. Thirdly, in the view of component behavior evolution, via setting up simulative relation, it was analyzed whether the component behavior consistency is preserved or not after evolution. Finally, a case study verifies the feasibility of the proposed method.

Keywords Software dynamic evolution, Software architecture, Behavior simulation, Consistency preservation

1 引言

构造性和演化性是软件的两个基本特性[1]。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(Software Evolution)的重要性和普适性越来越强[2]。历时22年总结出的Lehman定律论证了软件系统必定是不断演化的[3]。软件演化包括静态演化和动态演化:对于软件静态演化,已有相对完整的理论和可行的技术;与之相比,软件动态演化的相关理论、方法和技术还处于起步阶段[4]。因此,软件动态演化已逐渐成为软件工程领域的研究热点[5]。动态演化的目标在于支持系统在运行时刻以尽可能小的代价实施动态演化,即系统中不受动态演化实施影响的部分仍可以继续提供服务,而受影响的部分则须暂停服务而无法正常工作[4,6]。

软件动态演化仍面临多方面的"挑战"[4,5,7,8],重点关注以下两个方面的挑战:1)必须在保证动态演化实施的可靠性的前提下,尽可能降低动态演化实施的成本,即尽量缩小它对运行中系统的影响范围;2)破坏系统一致性的动态演化实施方案是不可行的,必须对动态演化实施方案进行可动态演化性分析,即方案必须保证动态演化实施前后的系统一致性。从软件体系结构(Software Architecture,SA)的角度,构件以及构件之间的连接是软件系统的基本组成部分[9]。因此,要降低动态演化的成本,必须对构件之间的相关性进行分析,找出与目标构件动态行为相关的最小构件集合。保证系统一致性的难点在于行为一致性,行为一致性包括内部一致性和外部一致性:内部一致性需保证目标构件在动态演化实施之后能够恢复状态,并从断点正确地继续执行下去;外部一致性需保证目标构件在实施演化之后能继续正确地与其它构件交互。

为了应对以上挑战并提高所建立的系统模型的动态演化性,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设计、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

到稿日期:2015-10-06 返修日期:2015-12-24 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(2012SE308,2012SE309),云南省教育厅科学研究基金(2014Y012)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;明 利(1989-),女,硕士生,主要研究方向为软件工程与形式化方法;林 英(1973-),女,博士,副教授,主要研究方向为软件工程和信息安全,E-mail:xiezw56@126.com(通信作者);秦江龙(1984-),男,讲师,主要研究方向为软件工程与形式化方法。

挑战为导向，以需求模型为驱动，以 SA 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，作者的前期成果是已完成的面向软件动态演化的需求建模[10]和软件体系结构建模[28]。因此，本文将在文献[10, 28]的基础上，对软件动态演化实施过程中的一致性进行分析。

本文第 2 节简介面向软件动态演化的 SA 建模，将其作为一致性分析的基础；第 3 节对所采取的一致性分析思路进行概述；第 4 节从结构的视角对构件的结构一致性进行分析；第 5 节从行为的视角对构件的行为一致性进行分析；第 6 节是案例研究；第 7 节是相关工作；最后总结与展望。

2 面向动态演化的 SA 建模

本文作者在文献[28]中提出了面向动态演化的 SA 元模型 DEAM，作为本文工作的重要基础，其主要部件定义如下。

定义 1(构件网结构)[28] 满足以下条件的六元组 $N=(P,T;F;E,A,C)$称作一个构件网结构：

(1)$P\cup T\neq\emptyset$ 且 $P\cap T=\emptyset$；

(2)$F\subseteq(P\times T)\cup(T\times P)$；

(3)$\mathrm{dom}(F)\cup ran(F)=P\cup T$；

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

(5)N 存在一个输入源(简称：源)$i\in P\cup T$，使得 $i=\emptyset$(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 $o^{\cdot}=\emptyset$(o 的后集为空)；

(6)$\forall x\in P\cup T$，都在从 i 到 o 的一个路径上。

其中，P 是库所集；T 是变迁集；F 是有向弧集，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；$\mathrm{dom}(F)=\{x\in P\cup T\mid\exists y\in P\cup T,(x,y)\in F\}$；$\mathrm{ran}(F)=\{x\in P\cup T\mid\exists y\in P\cup T,(y,x)\in F\}$。

定义 2(构件)[28] 构件是 SA 中承担计算功能的单元，用四元组 $Com=(N,I,O,S)$表示，其中：

(1)$N=(P,T;F;E,A,C)$是一个构件网结构，描述了构件的实现；

(2)I 是 N 的输入源 i，表示构件的输入接口；

(3)O 是 N 的输出槽 o，表示构件的输出接口；

(4)接口和端口统称为构件的接触点(*touch*)；

(5)S 是构件的依赖规约，用一个进程项表示(来源于文献[10]中的需求建模)，是从构件外部对构件的观察描述：表达构件必须实现的功能，否则构件将无法满足与环境之间的交互要求。

定义 3(连接件)[28] 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，其中：

(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^* 是自定义的复杂连接件类型；

(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复杂连接件时，Q 为其对应的连接件网结构；当连接件是基本类型时，Q 为空集。

定义 4(连接关系)[28] 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 5(子系统)[28] 子系统是一个三元组 $Sub=(COM,CON,L)$，满足以下条件：

(1)*COM* 是一个构件的集合，其中至少包含一个构件；

(2)*CON* 是一个自定义的复杂连接件的集合；

(3)L 是一个连接关系的集合；

(4)对于 *CON* 中的每一个连接件，其两个角色都参与连接关系，或者都不参与连接关系；

(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，得到一个连通图。

定义 6(静态视图)[28] SA 的静态视图是一个三元组 $SA_S=(SUB,CON,L)$，满足以下条件：

(1)*SUB* 是一个子系统集合，其中至少包含一个子系统；

(2)*CON* 是一个自定义的复杂连接件的集合；

(3)L 是一个连接关系的集合；

(4)对于 *CON* 中的每一个连接件，其两个角色都参与连接关系，或者都不参与连接关系；

(5)以 *SUB* 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。

定义 7(状态)[28] 对于一个构件 $Com=(P,T;F;E,A,I,O,C,S)$，映射 $M:P\to\{0,1,2,\cdots\}$称为构件的一个状态，记为状态 s；对于一个静态视图 SA_S 的构件集合 *COM*，*COM* 中每一个构件的状态记为 M_1，M_2，M_3，…，M_n，则 $M=M_1\cup M_2\cup M_3\cup\cdots\cup M_n$ 称为 *SA* 的一个状态，M_1，M_2，M_3，…，M_n 分别称为 *SA* 的一个构件子状态。

定义 8(动态视图)[28] 动态视图是相对于静态视图而言的，是一个四元组 $SA_D=(SUB,CON,L,M)$，满足以下条件：

(1)(SUB,CON,L)是对应的静态视图；

(2)M 是 *SA* 的初始状态；

(3)*SA* 的状态在运行规则和连接件的动态作用下，由于主动库所的执行、变迁的点火而不断改变其自身的状态。

关于面向动态演化的 SA 建模的细节请参考文献[28]。

3 一致性分析思路

对于软件动态演化的实施，必须保证其一致性。可动态演化性分析是指分析一个动态演化方案是否可行，用它来实施动态演化是否会破坏软件系统的一致性。可动态演化性分析是保证动态演化实施一致性的一个重要手段和基础。对于一个动态演化方案，若在可动态演化性分析时就得出“将导致系统不一致”的结论，那么该演化方案是不能被实施的。

对于一致性的定义，目前存在着定义不统一和概念混乱等问题[7,8]。本文基于文献[28]所建立的 SA 模型，从结构和行为两个角度进行可动态演化性分析，并分别定义结构一致性、基于行为的内部一致性和外部一致性作为可动态演化性分析的标准。由于动态演化一般以构件为基本单位，因此接下来分别从结构和行为两个角度进行构件的动态演化的一致性分析。

4 结构一致性分析

在对构件实施动态演化时，需要改变构件的部分实现。

基于本文的构件模型，需要对构件网结构的一个局部实施演化，先给出构件子网的概念。

定义 9(构件子网) 设 $N=(P,T;F;E,A,C)$是构件 X 的构件网结构，若 $P_1\subseteq P,T_1\subseteq T,F_1=F\cap(P_1\times T_1\cup T_1\times P_1),E_1\subseteq E,A_1\subseteq A,C_1\subseteq C$，则 $N_1=(P_1,T_1;F_1;E_1,A_1,C_1)$是 N 的一个子网。

在定义构件子网之后，对于构件的动态演化，从结构的视角，它就变成用“新子网”代替“旧子网”。

为了建立可动态演化性分析的结构一致性条件，必须对子网的边界和结构类型进行分析。

定义 10(子网的边界) 设 $N=(P,T;F;E,A,C),N_1=(P_1,T_1;F_1;E_1,A_1,C_1)$是 N 的子网，则 $\Omega(N_1):=\{x\in(P_1\cup T_1)\mid(\exists z\notin(P_1\cup T_1))\wedge(z\in\cdot x)\}$称为 N_1 的上边界，$\Psi(N_1):=\{x\in(P_1\cup T_1)\mid(\exists z\notin(P_1\cup T_1))\wedge(z\in x\cdot)\}$称为 N_1 的下边界，$\Omega(N_1)\cup\Psi(N_1)$称为 N_1 的边界。

定义 10 中 x 的后提 $\cdot x=\{y\in P\cup T\mid(y,x)\in F\}$；$x$ 的后提 $x\cdot=\{y\in P\cup T\mid(x,y)\in F\}$。

对于子网的边界，若其中的元素都是库所，则其是库所类型边界，简称 P 型边界；若其中的元素都是变迁，则其是变迁类型边界，简称 T 型边界。

定义 11(子网替换) 设 $N=(P,T;F;E,A,C)$是构件 X 的构件网结构，$N_1=(P_1,T_1;F_1;E_1,A_1,C_1)$是 N 的一个子网，$N_2=(P_2,T_2;F_2;E_2,A_2,C_2)$是演化方案定义的子网，用子网 N_2 替换 N_1 得到的新构件网结构 $N'=(P',T';F';E',A',C')$为：

(1)$P'=(P-P_1)\cup P_2,T'=(T-T_1)\cup T_2$；

(2)$F'=(F-F_1-F(\Omega_1)-F(\Psi_1))\cup F_2\cup F(\Omega_2)\cup F(\Psi_2)$，其中 $F(\Omega_i)$是从子网外指向子网的上边界的弧的集合，$F(\Psi_i)$是从子网的下边界指向子网外部的弧的集合；

(3)$E'=(E-E_1)\cup E_2,A'=(A-A_1)\cup A_2,C'=(C-C_1)\cup C_2$。

结构一致性分析要求在分析演化方案中的子网替换之后，得到的新的构件网结构依然满足构件网结构的定义。

定义 12(结构一致性) 若使用动态演化方案中的子网替换原构件中的子网，得到的新构件的实现满足构件网结构的要求，则称该动态演化方案满足结构一致性。

定义 13(子网类型) 根据子网边界类型的不同，可以将子网划分为以下 4 种类型：1)P 型子网，其上边界和下边界都是 P 型；2)T 型子网，其上边界和下边界都是 T 型；3)PT 型子网，其上边界是 P 型，下边界是 T 型；4)TP 型子网，其上边界是 T 型，下边界是 P 型。

子网类型只是对子网的边界进行了分析，为了使子网替换满足结构一致性分析，还要参照构件网结构对子网的内部结构进行限制。

定义 14(良性子网) 若子网 $N_1=(P_1,T_1;F_1;E_1,A_1,C_1)$中的任意一个节点 $x\in P_1\cup T_1$ 都属于从上边界到下边界的一个路径，则称该子网是良性的。

定理 1 对于构件 X 的构件网结构 $N=(P,T;F;E,A,C),N_1=(P_1,T_1;F_1;E_1,A_1,C_1)$是 N 的子网，$N_1'=(P_1',T_1';F_1';E_1',A_1',C_1')$是演化方案中用于替换 N_1 的子网。该演化方案满足结构一致性，当且仅当：

(1)N_1 和 N_1'的子网类型相同；

(2)N_1'是一个良性子网。

证明：由构件网结构的定义，显然有：构件网结构的子网演化前后类型相同，并且其子网是个良性子网。考虑到子网替代时可能破坏流关系和路径关系两个条件，其它条件显然未受破坏。因此，只需分别证明满足以下两个条件的子网替换不会破坏结构一致性。

(1)由于 N_1 和 N_1'的子网类型相同，因此 $F'=(F-F_1-F(\Omega_1)-F(\Psi_1))\cup F_1'\cup F(\Omega_1')\cup F(\Psi_1')\subseteq(P'\times T')\cup(T'\times P')$，即边界的流关系得到保持；

(2)由于 N_1'是一个良性子网，因此任意一个节点 $x\in(P_1'\cup T_1')$都属于从上边界到下边界的一个路径，由于 N 满足构件网结构模型的定义，因此 x 属于从 i'到 o'的一个路径；又任意一个节点 $y\in(P\cup T-P_1\cup T_1)$，在 N 中必定属于从 i 到 o 的一个路径 l，在 N_1'中必定存在一条从上边界到下边界的路径，替代 N_1 中对应的路径之后得到 l'，l'是属于从 i'到 o'的一个路径，且 $y\in l'$；由于 $P'\cup T'=(P_1'\cup T_1')\cup(P\cup T-P_1\cup T_1)$，因此任意一个节点 $z\in P'\cup T'$ 都属于从 i'到 o'的一个路径，即所有节点的路径关系得到保持。

综上：N_1'必定满足构件网结构模型的定义，即演化方案满足结构一致性。□

5 行为一致性分析

接下来从行为的视角分析动态演化方案是否满足行为一致性。采用分而治之的策略，分别对构件本身行为的内部一致性和构件之间行为的外部一致性进行分析。

5.1 行为的内部一致性分析

保证构件行为的内部一致性的目标在于：确保构件在实施动态演化之后能够恢复状态，并从断点正确地继续执行下去。可见，对于演化方案中描述的演化后的构件，若其行为具有继续完成演化之前的构件被中断的任务的能力，则该演化方案将满足行为的内部一致性。

考虑到本文基于构件网结构和动态构件系统描述构件，接下来从构件的变迁执行序列的角度形式定义构件行为的内部一致性。

定义 15(内部一致性) 假设演化前的构件为 Com_1，演化方案描述的构件为 Com_2；在动态演化实施前的构件状态为 $s_1(Com_1)$，演化方案中对应的构件状态为 $s_2(Com_2)$；在 $s_1(Com_1)$状态下，其可能的后续变迁序列的集合为 Σ_1，在 $s_2(Com_2)$状态下其可能的后续变迁序列的集合为 Σ_2。若满足 $\Sigma_1\subseteq\Sigma_2$，则称该演化方案满足行为的内部一致性。

由定义 15 可知，内部一致性要求：对于被中断的任务而言，演化方案的实施必须保证构件执行未完成的任务的能力不被削弱；当然，若构件中没有被中断的任务，那么 Σ_1 为空集，则不受此限制。

由于本文的构件模型基于扩展的 Petri 网，而 Petri 网作为一种操作类的形式化方法，其每一个变迁的执行描述了构件的每一步行为。正是变迁的点火执行使得构件从一个状态转变为另一个状态，因此首先给出变迁行为的定义。

定义 16(变迁行为) 对于构件 Com 而言，其变迁行为是一个三元组 $b=(s,t,s')$，其中：1)s 和 s'分别表示变迁 t 执行前和执行后的构件状态；2)t 表示导致构件状态发生变化的变迁，$t\in T$。

定义 17(构件行为图) 构件 Com 在初始状态 s_1 下，其行为图是 $G=(s_1,S,E)$，其中：1) s_1 是初始状态，是行为图的顶点；2) S 是 s_1 可达的构件状态的集合，是行为图的节点集，且满足 $s_1\in S$；3) E 是变迁行为的集合，其中每一个元素 b 满足 $b.s\in S$ 和 $b.s'\in S$，E 是 G 的有向边的集合。

构件行为图的本质是 Petri 网的可达图，因此可以采用可达图的构造算法来构造行为图，具体算法可参考文献[14]，本文不再重复给出。

对于被中断的任务和构件而言，有了行为图之后，就可以以中断状态为顶点，分别构造演化前构件的行为图和演化方案描述的行为图，并通过行为图比较来判断是否满足行为的内部一致性。

从行为图的角度，要求演化方案的构件描述从该特定状态开始的行为，必须能够包含演化之前的构件从对应状态开始的行为，即演化方案的构件从该状态开始的行为模式至少跟演化前的构件的行为模式一样丰富。接下来，参照进程代数中的强模拟关系[11]，定义构件的强模拟关系。

定义 18(构件的强模拟关系) 假设构件 Com_1 的状态 s_1 和演化方案描述的构件 Com_2 的状态 s_2 相对应，G_1 和 G_2 分别是 Com_1 和 Com_2 在 s_1 和 s_2 状态下的行为图，称 Com_2(在 s_2 状态下)强模拟 Com_1(在 s_1 状态下)，记为 $Com_2(s_2)\Rightarrow Com_1(s_1)$，满足：

(1)对于 G_1，若存在变迁行为 $b_1=(s_1,t,s_1')$，则对于 G_2，存在变迁行为 $b_2=(s_2,t,s_2')$与之对应；

(2)$Com_2(s_2')\Rightarrow Com_1(s_1')$。

构件的强模拟关系意味着：无论构件 Com_1 选择哪条变迁行为路径，构件 Com_2 都能找到一条相应的变迁行为路径，且该路径上保留了 Com_1 的所有选择。需要注意的是，本文的强模拟关系是单向的，即要求构件 Com_2 能强模拟 Com_1，但不要求构件 Com_1 也能强模拟 Com_2，否则就成了强互模拟，这也是由动态演化的单向性决定的。

定理 2 假设构件 Com_1 的状态 s_1 和演化方案描述的构件 Com_2 的状态 s_2 相对应，若满足条件：$Com_2(s_2)\Rightarrow Com_1(s_1)$，则称该演化方案保持了行为的内部一致性。

证明：只需证明对于在 $s_1(Com_1)$状态下的变迁序列集合 Σ_1 中的任意一个元素 λ，若 $Com_2(s_2)\Rightarrow Com_1(s_1)$，则 λ 必定属于 $s_2(Com_2)$状态下的变迁序列集合 Σ_2。归纳证明如下：

第 1 步 对于 Σ_1 中的任意一个变迁序列 λ，假设 λ_1 是序列 λ 的第一个变迁，λ_n 是序列 λ 的前 n 个变迁序列，依此类推。由于 $Com_2(s_2)\Rightarrow Com_1(s_1)$，因此对于 G_1，若存在变迁行为 $b_1=(s_1,\lambda_1,s_1')$，则对于 G_2，必然也存在变迁行为 $b_2=(s_2,\lambda_1,s_2')$，且使得 $Com_2(s_2')\Rightarrow Com_1(s_1')$。可见，在 Σ_2 中必然存在一个变迁序列 λ'，使得 λ_1 是 λ'的第一个变迁。

第 2 步 对于 Σ_1 中的任意一个变迁序列 λ，假设在 Σ_2 中存在一个变迁序列 λ'，使得 λ'的前 n 个变迁为 λ_n。由于 $Com_2(s_2)\Rightarrow Com_1(s_1)$，因此在 n 个相同变迁情况下，必定存在 n 组对应的变迁行为，故有：$Com_2(s_2^n)\Rightarrow Com_1(s_1^n)$。进一步，对于 G_1，若存在变迁行为 $b_1^{n+1}=(s_1^n,t_{n+1},s_1^{n+1})$，其中 t_{n+1}是 λ 的第 $n+1$ 个元素，则对于 G_2，必然也存在对应的$b_2^{n+1}=(s_2^n,t_{n+1},s_2^{n+1})$，且使得 $Com_2(s_2^{n+1})\Rightarrow Com_1(s_1^{n+1})$。因此，对于 Σ_1 中的任意一个变迁序列 λ，在 Σ_2 中必定存在一个对应的变迁序列 λ'，使得 λ'的前 $n+1$ 个变迁为 λ_n。

第 3 步 依此类推，必定存在 $\lambda'=\lambda$，即对于任意的变迁序列 $\omega\in\Sigma_1$，必有 $\omega\in\Sigma_2$，即满足 $\Sigma_1\subseteq\Sigma_2$。

因此，若 $Com_2(s_2)\Rightarrow Com_1(s_1)$，则演化方案 B_{Com} 满足行为的内部一致性。□

5.2 行为的外部一致性分析

保证构件行为的外部一致性的目标在于：确保构件在实施动态演化之后，能够继续正确地与环境(即其他依赖于它的构件)交互，以继续协作完成任务。考虑到环境对于待演化构件的要求是通过构件的依赖规约的形式描述，因此对于演化方案中描述的演化后的构件，若其行为依然能够满足依赖规约的行为要求，则该演化方案将满足行为的外部一致性。进一步，依赖规约来自于文献[10]中的需求建模，是通过使用通信进程代数 ACP[15]的进程项的形式进行描述的。

与行为的内部一致性分析时需要考虑构件所处的状态不同，外部一致性只需考虑构件具备与环境交互的能力，因此考虑构件在初始输入状态下具备的行为能力。首先定义构件的初始输入状态。

定义 19(构件的初始输入状态) 对于构件 Com，输入外延 I^* 为其输入接口 I 和输入接口 I 的后提 $I^\cdot$ 的集合，即$I^*=I\cup I^\cdot$，其初始输入状态是指：输入外延中的库所包含有一个托肯，除此之外，构件中的其它库所都不包含托肯的状态。

定义 19 中 I 的后提 $I^\cdot=\{L\in P\cup T\mid(I,L)\in F\}$。

接下来，分别从构件的变迁执行序列、构件规约的行为迹两个角度定义构件行为的外部一致性。

定义 20(外部一致性) 假设对于演化方案描述的构件 Com，其依赖规约为 S，S 包含的事件集记为 $E(S)$，S 的迹的集合记为 $T(S)$。对于初始输入状态 $s_0(Com)$，其可能的后续变迁序列的集合记为 Σ，若满足条件：$T(S)\subseteq(\Sigma\backslash E(S))$(其中 $\Sigma\backslash E(S)$表示序列集合中的每个序列在集合 $E(S)$中的投影的序列集合)，则称该演化方案满足行为的外部一致性。

由定义 20 可知，内部一致性要求：演化后的构件在构件外部环境看来其行为保持依赖规约的约束，即对构件环境而言演化方案的实施应是透明的。

由于依赖规约使用进程项描述，而演化方案描述的构件使用扩展 Petri 网来描述，因此为了描述进程项所规定的行为，并与变迁行为(定义 16)相对应，提出进程事件行为(简称事件行为)的概念。

定义 21(事件行为) 对于构件的依赖规约 S 而言，其事件行为是一个三元组 $d=(p,t,p')$，其中：1) p 和 p'表示进程，对应于变迁行为中定义的状态；2) t 表示导致进程发生变化的事件，对应于变迁行为的变迁，$t\in E$，E 是事件集。

事件行为 $d=(p,t,p')$表示进程 p 执行事件 t 后变为进程 p'；特殊的事件行为$(p,t,\surd)$表示进程 p 执行事件 t 后成功地终止。

在定义事件行为的基础上，接下来定义依赖规约的行为规约图，以便与构件的行为图相比较，进而判断是否满足行为的外部一致性。

定义 22(行为规约图) 对于构件的依赖规约 S 而言，其行为规约图是三元组 $G=(p_0,P,E)$，其中：1) p_0 是依赖规约进程项，是行为规约图的顶点，对应于行为图的初始状态；2) P 是 p_0 的后续进程集(包含 p_0)，是行为规约图的节点集，对应行为图的状态集合；3) E 是事件行为的集合，其中每一个元

素 d 满足 $d.p\in P$ 和 $d.p'\in P$，是行为规约图的有向边的集合，对应行为图的变迁行为集合。

接下来，给出行为规约图的构造算法。

算法 1 构造依赖规约 S 的行为规约图

输入：依赖规约的进程项 p_0

输出：行为规约图 $G(p_0)$

(1)初始化有向图 $G(p_0)=(P,E)=(\{p_0\},\varnothing)$，$p_0$ 未做标记；

(2)while 集合 P 中还存在未标记的节点 do

(2.1)从 P 中任选一个未做标记的节点 p_1 并标记它；

(2.2)for 每个在 p_1 中可执行的事件 t do

(2.2.1)计算 p_2，得到事件行为(p_1,t,p_2)使得 t 发生；

(2.2.2)P:=P∪$\{p_2\}$，其中 p_2 未做标记；

(2.3.3)E:=E∪$\{(p_1,t,p_2)\}$；

(3)P 中不存在未做标记的节点，输出 $G(p_0)$，算法结束。

算法中步骤 2.2 中 p_2 的计算依据参见文献[16]的表3.1中的行为特征元模型的变迁规则。

由于依赖规约描述了环境对构件的观察，因此若演化方案中描述的构件依然满足依赖规约规定的行为，则从构件外部观察看来，演化方案中描述的构件与演化前的构件并无差别，哪怕它们的内部实现有巨大的不同。可见，行为的外部一致性保持的关键在于演化方案中描述的构件依然能"模拟"其依赖规约的行为。但是，这种"模拟"与内部一致性的构件强模拟关系不同：这种"模拟"是从外部观察的角度要求的一种"观察模拟"，它只关注环境能观察到的"外部行为"，而忽略了外部观察不到的"内部行为"。

接下来，参照进程代数中的弱模拟关系[11]，定义演化方案对依赖规约的弱模拟关系。

定义 23(演化方案对依赖规约的弱模拟关系) 假设对于演化方案描述的构件 Com，依赖规约为 $S=p_0$，G_1 是 S 的行为规约图，G_2 是 Com 在初始输入状态 s_0 下的构件行为图，称演化方案的构件 Com 弱模拟依赖规约 S，记为 $Com(s_0)\rightarrow S(p_0)$，满足：

(1) 对于 G_1，若存在事件行为序列 $e=d_1\cdots d_n$，即 $G_1\xrightarrow{e}G_1'$，其中 d_k 对应事件行为(p_{k-1},t,p_k)，$1\leqslant k\leqslant n$，则对于 G_2，存在对应的变迁行为序列 $e=b_1\cdots b_n$ 满足 $G_2\xrightarrow{e}G_2'$，其中 b_k 对应变迁行为(s_{k-1},t,s_k)，$1\leqslant k\leqslant n$；

(2) 对于 G_1'和 G_2'，满足 $Com(s_n)\rightarrow S(p_n)$。

定义中的$\xrightarrow{e}$表示一个试验 $e=\lambda_1\cdots\lambda_n$ 的执行，期间可能夹杂了任意数量的外部不可观察的内部事件 τ。

演化方案对依赖规约的弱模拟意味着：从构件外部的环境观察，构件的实现满足其依赖规约的行为要求。在对构件实施动态演化之前，构件的内部实现通常是满足其依赖规约的。为了保证行为的外部一致性，要求构件在实施动态演化之后，不管其内部实现如何改变，它必须仍然满足构件的依赖规约，即演化方案必须能弱模拟依赖规约的行为。

定理 3 假设对于演化方案描述的构件 Com 及初始输入状态 s_0，其依赖规约为 $S=p_0$，若满足条件：$Com(s_0)\rightarrow S(p_0)$，则称该演化方案保持了行为的外部一致性。

证明：只需证明对于 S 的迹集合 $T(S)$ 中的任意一个元素 λ，若 $Com(s_0)\rightarrow S(p_0)$，则必存在一个 $\mu\in\Sigma$，使得 $\mu\backslash E(S)=\lambda$，$\Sigma$ 是 $s_0(Com)$ 状态下的变迁序列集合。归纳证明如下：

第 1 步 对于 $T(S)$中的任意一个元素 λ，假设 $\lambda_1,\cdots,\lambda_l$ 分别是λ 序列的第 1 个到第 l 个事件，依此类推。对于 G_1，若存在事件行为序列 $e=\lambda_1\cdots\lambda_k$，试验 e 的执行使得 G_1 状态变化，即 $G_1\xrightarrow{e}G_1'$，由于 $Com(s_0)\rightarrow S(p_0)$，因此对于 G_2，必然也存在对应的变迁行为序列 σ 满足 $G_2\xrightarrow{e}G_2'$，且 $\sigma\backslash E(S)=e$，同时，对于 G_1'和 G_2'，满足 $Com(s_n)\rightarrow S(p_n)$。

第 2 步 对于 $T(S)$中的任意一个变迁序列 λ，假设在 Σ 中存在一个变迁序列 σ，使得 $\sigma\backslash E(S)$的前 n 个变迁序列为$\lambda_1,\cdots,\lambda_n$。由于 $Com(s_n)\rightarrow S(p_n)$，因此在 n 个相同变迁序列的情况下，必定存在 n 组对应的试验行为，故有：$Com(s_n^n)\rightarrow S(p_n^n)$。进一步，对于 G_1，若存在事件行为序列 $b_1^{n+1}=(s_1^n,e_{n+1},s_1^{n+1})$，其中 e_{n+1}对应 λ_n 后的一个序列串，则对于 G_2，必然也存在对应的 $b_2^{n+1}=(s_2^n,\sigma_{n+1},s_2^{n+1})$，由于 $E(S)$之外的其它变迁行为都被抽象为哑动作 τ，因此：$\sigma_{n+1}\backslash E(S)=e_{n+1}$；同时，使得 $Com(s_n^{n+1})\rightarrow S(p_n^{n+1})$。所以，对于 $T(S)$中的任意一个变迁序列 λ，在 Σ 中必定存在一个对应的变迁序列 μ，使得 $\mu\backslash E(S)=\lambda$。

第 3 步 依此类推，必定存在 $\sigma\backslash E(S)=\lambda$。因此，对于任意的变迁序列 $\lambda\in E(S)$，必然存在一个 $\mu\in\Sigma$，使得 $\mu\backslash E(S)=\lambda$，即 $T(S)\subseteq(\Sigma\backslash E(S))$。

即，若 $Com(s_0)\rightarrow S(p_0)$，则方案 B_{Com} 满足行为的外部一致性。□

6 案例研究

6.1 案例说明

为了说明本文所提方法的可行性，以文献[28]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[28]已建立了该案例的 SA 模型，接下来将进一步对其进行面向动态演化的一致性分析。

6.2 可动态演化性分析

接下来对可动态演化性分析进行案例研究。考虑业务处理子系统中的代缴电费复合构件，该构件包含以下 4 个子构件：缴费预处理子构件、短信验证子构件、缴费实施子构件、转账子构件。

由于结构一致性分析较为直观和简单，因此这里重点对行为一致性分析进行示例。以缴费实施子构件为例，对其分别进行行为的内部一致性和外部一致性的可动态演化性分析。在分析之前，先给出缴费实施子构件的构件模型，如图 1 所示。

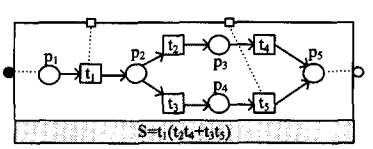

图 1 缴费实施子构件

6.3 演化方案的内部一致性分析

内部一致性分析与构件所处的状态相关，因此首先给出动态演化之前的构件所处的状态，如图 2 所示，其状态 M_0 为：$\{p_2=1\}$。

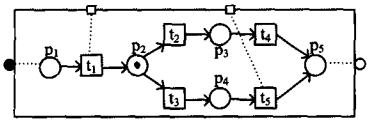

图 2　缴费实施子构件动态演化前的状态

考虑以下两种对该构件实施动态演化的方案(分别称动态演化方案 1 和动态演化方案 2),其对应的构件结构和状态分别如图 3 和图 4 所示。

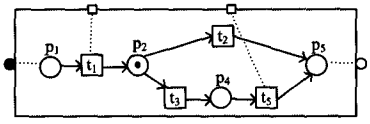

图 3　动态演化方案 1

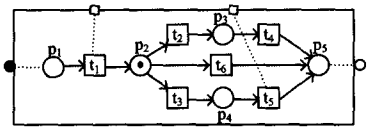

图 4　动态演化方案 2

在分析是否保持内部一致性之前,给出演化之前以及两种演化方案的构件行为图,如图 5 所示。

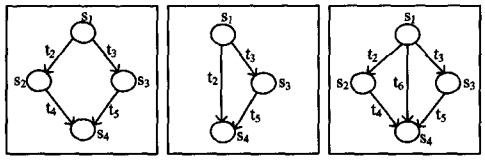

图 5　内部一致性对应的 3 个构件行为图

由图 5 分析易知,动态演化方案 2 的构件行为图与演化前的行为图相比,多出一个分支 t_6,并且演化前构件的状态和选择分支都得到保留,即在对应状态下演化方案 2 可以强模拟演化之前的构件所处的状态;而对于演化方案 1,由于在其构件行为图中找不到一个状态对应演化前构件的 s_2 状态,而无法强模拟演化之前的构件所处的状态,因此方案 1 不能保证保持行为的内部一致性,而方案 2 保持了行为的内部一致性。

6.4　演化方案的外部一致性分析

由于行为的外部一致性分析以构件对依赖规约的弱模拟关系为判断标准,因此需要首先讨论构件的依赖规约的行为图。由缴费实施子构件的依赖规约 $S=t_1\cdot(t_2\cdot t_4+t_3\cdot t_5)$ 易知:演化前的构件实现满足对规约的弱模拟关系,如图 6 所示。

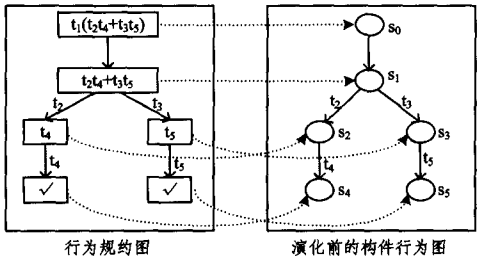

图 6　演化前的构件对依赖规约的弱模拟关系

类似地,考虑以下两种对构件实施动态演化的方案(分别称动态演化方案 3 和方案 4),其对应的构件结构和初始输入状态如图 7 和图 8 所示。

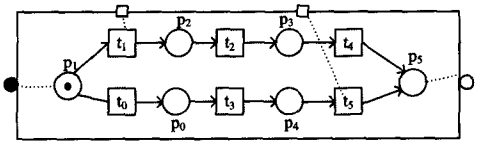

图 7　动态演化方案 3

动态演化方案 3 将原变迁 t_1 演化为两个对应变迁 t_0 和 t_1,在库所 p_1 处进行选择,若选择变迁 t_1,接着将执行变迁 t_2 和 t_4;若选择变迁 t_0,接着将执行变迁 t_3 和 t_5。

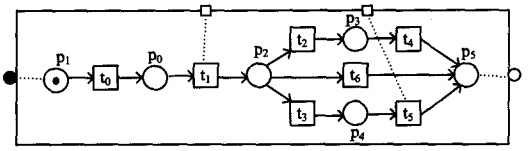

图 8　动态演化方案 4

动态演化方案 4 在原变迁 t_1 之前添加对应变迁 t_0,同时,在库所 p_2 处添加一个选择分支 t_6。

首先,分析方案 3 是否满足行为的外部一致性,行为规约图和方案 3 的构件行为图如图 9 所示。

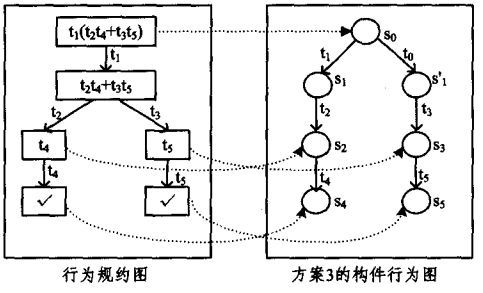

图 9　行为规约图与方案 3 的构件行为图

由图 9 中行为规约图与方案 3 的构件行为图分析可得:对于行为规约图中的状态 $t_2\cdot t_4+t_3\cdot t_5$,在方案 3 的构件行为图中无法找到一个与之对应的状态,因此方案 3 的构件行为图无法弱模拟行为规约图,即:方案 3 无法保证行为的外部一致性得到保持。

实际上,反过来,行为规约图可以弱模拟方案 3 的构件行为图,因为无论方案 3 的构件行为图中是 s_1 或是 s_1',都可以用规约图中的状态 $t_2\cdot t_4+t_3\cdot t_5$ 来与之对应,可见,行为规约图的行为模式较方案 3 的行为模式更为丰富。原因在于:行为规约图中的状态 $t_2\cdot t_4+t_3\cdot t_5$ 还具有选择的行为,而方案 3 中,无论是 s_1 或是 s_1',都已失去了选择的行为。

接着,分析方案 4 是否满足行为的外部一致性,行为规约图与方案 4 的构件行为图如图 10 所示。

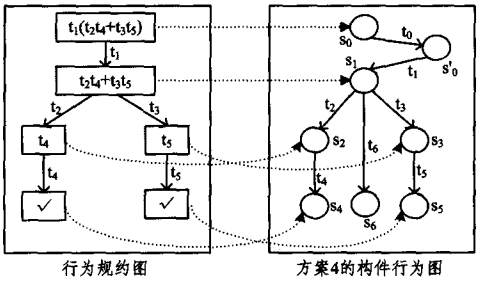

图 10　行为规约图与方案 4 的构件行为图

由图 10 中行为规约图与方案 4 的构件行为图分析可知:在方案 4 的行为图中,一方面,状态 s_0 必须先执行 t_0,但由于变迁 t_0 将被抽象为特殊事件 τ,即作为外部不可观察的行为

变迁，因此 t_0 的存在不会使得弱模拟关系受到破坏；另一方面，状态 s_1 多了一个选择分支 t_6，同样，这是对行为模式的丰富，也不会使得弱模拟关系受到破坏。可见，方案 4 的构件行为图对依赖规约图满足弱模拟关系。因此，方案 4 保持了行为的外部一致性。

7 相关工作

目前，围绕软件动态演化及其 SA 建模和一致性分析的研究工作主要集中在 3 个方面：1）基于 SA 的动态演化描述和规则制定。研究者以 SA 为基础，通过引入各种操作和规则，以描述、指导和支持动态演化。例如：Miladi 等应用统一建模语言（Unified Modeling Language，UML）profile 建立构件和连接件的添加、删除规则[17]；kacem 等扩展了 UML，通过动态元类建立 SA 的添加、删除规则和操作[18]；Bruni 等采用类型图文法建立相关的重配置规则[19]。2）支持动态演化的结构模型和体系结构描述语言（Architecture Description Language，ADL）设计。例如：Abi-Antoun[20]、梅宏[21]等使用不同的 ADL 描述 SA 动态演化；丁博[22]、赵会群[23]、王映辉[24]等提出了不同的 SA 动态演化的模型。3）支持动态演化的软件框架或平台。例如：OSGi 联盟制定的服务平台规范及其实现的 OSGi 框架为服务构件（被称为 Bundle）的动态演化提供了一个通用的基础设施[25]；黄罡等提出了基于体系结构反射的 PKUAS 系统[26]；马晓星等提出了一种基于图文法的支持动态 SA 对象演化的支撑环境[27]。在以上相关工作的基础上，本文从结构和行为两个视角进行可动态演化性分析，分析了动态演化方案在结构和行为两个层次是否满足一致性，进而判断演化方案是否可以用于动态演化实施。目前，关于一致性保持的分析方面的工作大都是从结构角度进行静态分析，从行为视角进行一致性分析的工作尚属鲜见。

结束语 针对软件动态演化面临的挑战，本文提出了从结构和行为两个视角对动态演化方案进行一致性分析的方法。本文工作为建立具有高动态演化性的系统模型奠定了基础，为动态演化分析和实施提供了 SA 视图，为保证动态演化实施的可靠性提供了支持。

下一步工作：以本文的工作为基础，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、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，对动态演化的一致性分析为行为相关性分析提供了一致性保证，行为图等的构造为行为相关性问题的研究做了部分准备工作。

参 考 文 献

[1] Yang Fu-qing. Thinking on the Development of Software Engineering Technology[J]. Journal of Software, 2005, 16(1): 1-7(in Chinese)
杨芙清. 软件工程技术发展思索[J]. 软件学报, 2005, 16(1): 1-7

[2] Li Tong. An Approach to Modelling Software Evolution Processes[M]. Springer-Verlag, Berlin, 2008

[3] Lehman M M. Laws of software evolution revisited[C]// Proceeding of the European Workshop on Software Process Technology. Namcy, 1996: 108-124

[4] Li Chang-yun. Research on Architecture-Based Software Dynamic Evolution[D]. Hangzhou: Zhejiang University, 2005(in Chinese)
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

[5] Xu Hong-zhen, Zeng Guo-sun, Chen Bo. Conditional Hypergraph Grammars and Its Analysis of Dynamic Evolution of Software Architectures[J]. Journal of Software, 2011, 22(6): 1210-1223(in Chinese)
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

[6] Mens T, Buckley J, Zenger M, et al. Towards a Taxonomy of Software Evolution[C]// Proceeding of International Workshop on Unanticipated Software Evolution. Warsaw, Poland, 2003: 1-18

[7] Moazami-Goudarzi K. Consistency preserving dynamic reconfiguration of distributed systems[D]. London: Imperial College, 1999

[8] Dou Lei. Research on Dynamic Reconfiguration Technology in Component-Oriented Complex Software System [D]. National University of Defense Technology. Changsha, 2005(in Chinese)
窦蕾. 面向构件的复杂软件系统中动态配置技术的研究[D]. 长沙：国防科学技术大学, 2005

[9] Mei Hong, Shen Jun-rong. Progress of Research on Software Architecture[J]. Journal of Software, 2006, 17(6): 1257-1275(in Chinese)
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

[10] Xie Zhong-wen, Li Tong, Dai Fei, et al. An Approach to Modeling and Normalizing Dynamic-evolution-oriented Software Requirements[J]. Journal of Frontiers of Computer Science and Technology, 2012, 6(6): 557-576(in Chinese)
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

[11] Milner R. Communicating and Mobile Systems: the pi-calculus [M]. Cambridge University Press, 1999

[12] IEEE. IEEE recommended practice for architectural description of software-intensive systems[J]. IEEE, IEEE Std1471-2000, 2000

[13] Qin Zheng, Xing Jian-kuan, Dong Jin-chun, et al. Software Architecture(Second Edition)[M]. Beijing: Tsinghua University Press. 2008(in Chinese)
覃征，刑剑宽，董金春，等. 软件体系结构(第 2 版)[M]. 北京：清华大学出版社, 2008

[14] Girault C, Valk R. Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications[M]. Berlin: Springer-Verlag. 2003

[15] Fokkink W. Introduction to Process Algebra [M]. NewYork, Springer-Verlag, 2007

[16] Xie Zhong-wen. Model and Methodology of Requirements-Driven and SA-Based Software Dynamic Evolution[D]. Kunming: Yunnan University, 2012(in Chinese)
谢仲文. 一种需求驱动、以体系结构为视图的面向软件动态演化

的模型与方法[D]. 昆明:云南大学,2012

[17] Miladi M N, Jmaiel M, Kacem M H. A UML profile and a FUJABA plugin for modelling dynamic software architectures[C]// Proc. of the Workshop on Model-Driven Software Evolution. Washington: IEEE Press, 2007: 20-26

[18] Kacem M H, Kacem A H, Jmaiel M, et al. Describing dynamic software architectures using an extended UML model[C]// Proc. of the Symp. on Applied Computing. New York: ACM Press, 2006: 1245-1249

[19] Bruni R, Bucchiarone A, Gnesi S, et al. Modelling dynamic software architectures using typed graph grammars[J]. Electronic Notes in Theoretical Computer Science, 2008, 213(1): 39-53

[20] Abi-Antoun M, Aldrich J, Garlan D, et al. Modeling and implementing software architecture with acme and archJava[C]// Proc. of the 27th Int'l Conf. on Software Engineering. New York: ACM Press, 2005: 676-677

[21] Mei H, Chen F, Wang Q X, et al. ABC/ADL: An ADL supporting component composition[C]// ICFEM 2002. Shanghai, 2002: 38-47

[22] Ding Bo, Wang Huai-min, Shi Dian-xi, et al. Component Model Supporting Trustworthiness-Oriented Software Evolution[J]. Journal of Software, 2011, 22(1): 17-27(in Chinese)
丁博,王怀民,史殿习,等. 一种支持软件可信演化的构件模型[J]. 软件学报,2011,22(1):17-27

[23] Zhao Hui-qun, Sun Jing. An Algebraic Model of Service Oriented Trustworthy Software Architecture[J]. Chinese Journal of Computer, 2010, 33(5): 890-899(in Chinese)
赵会群,孙晶. 面向服务的可信软件体系结构代数模型[J]. 计算机学报,2010,33(5):890-899

[24] Wang Ying-hui, Liu Yu, Wang Li-fu. SA Dynamic Evolution Model Based on Static-Point Transition[J]. Chinese Journal of Computer, 2004, 27(11): 1451-1456(in Chinese)
王映辉,刘瑜,王立福. 基于不动点转移的 SA 动态演化模型[J]. 计算机学报,2004,27(11):1451-1456

[25] OSGi Service platform [OL]. http://core.ac.uk/display/38522655

[26] Huang G, Mei H, Yang F Q. Runtime software architecture based on reflective middleware[J]. Science in China(Series E), 2004, 34(2): 121-138(in Chinese)
黄罡,梅宏,杨芙清. 基于反射式软件中间件的运行时软件体系结构[J]. 中国科学(E 辑),2004,34(2):121-138

[27] Ma X X, Cao C, Yu P, et al. A supporting environment based on graph grammar for dynamic software architectures[J]. Journal of Software, 2008, 19(8): 1881−1892(in Chinese)
马晓星,曹春,余萍,等. 基于图文法的动态软件体系结构支撑环境[J]. 软件学报,2008,19(8):1881-1892

[28] Xie Zhong-wen, Li Tong, Dai Fei, et al. Modelling Dynamic Evolution-oriented Software Architecture Based on Petri Net[J]. Computer Applications and Software, 2012, 29(10): 36-39, 127(in Chinese)
谢仲文,李彤,代飞,等. 基于 Petri 网的面向动态演化的软件体系结构建模[J]. 计算机应用与软件,2012,29(10):36-39,127

(上接第 233 页)

[5] Rizomiliotis P. On the resistance of Boolean functions against algebraic attacks using univariate polynomial representation[J]. IEEE Transactions on Information Theory, 2010, 56(8): 4014-4024

[6] Tu Z R, Deng Y P. A class of 1-resilient function with high nonlinearity and algebraic immunity[R]. Ryptography ePrint Archive, Report 2010, 2010/179

[7] Wang Q, Peng J, Kan H, et al. Constructions of cryptographically significant Boolean functions using primitive polynomials[J]. IEEE Transactions on Information Theory, 2010, 56(6): 3048-3053

[8] Li C L, Zhang H G, Zeng X Y, et al. The lower bound on the second-order nonlinearity for a class of Bent functions[J]. Chinese Journal of Computers, 2012, 35(8): 1588-1593(in Chinese)
李春雷,张焕国,曾祥勇,等. 一类 Bent 函数的二阶非线性度下界[J]. 计算机学报,2012,35(8):1588-1593

[9] Sarkar S, Gangopadhyay S. On the second order nonlinearity of a cubic Maiorana-McFarland Bent Functions[J]. International Journal of Foundations of Computer Science, 2010, 21(3): 243-254

[10] Su S H, Tang X H. Construction of rotation symmetric Boolean functions with optimal algebraic immunity and high nonlinearity[J]. Designs, Codes and Cryptography, 2014, 71(2): 183-199

[11] Chen Y D, Zhang Y N, Tian W. Construction of Even-variable Rotation Symmetric Boolean Functions with Optimal Algebraic Immunity[J]. Journal of Cryptologic Research, 2014, 1(5): 437-448(in Chinese)
陈银冬,张亚楠,田威. 具有最优代数免疫度的偶数元旋转对称布尔函数的构造[J]. 密码学报,2014,1(5):437-448

[12] Sarkar S, Maitra S. Construction of rotation symmetric Boolean functions with optimal algebraic immunity[J]. Computation Systems, 2009, 12(3): 267-284

[13] Fu S, Qu L, Li C, et al. Balanced rotation symmetric Boolean functions with maximum algebraic immunity[J]. IET Information Security, 2011, 5(2): 93-99

[14] Dong D S, Li C, Qu L J, et al. Rotation symmetric Boolean functions in even-variable maximum algebraic immunity[J]. Journal of National University of Defense Technology, 2012, 34(4): 85-89(in Chinese)
董德帅,李超,屈龙江,等. 偶变元 MAI 旋转对称布尔函数[J]. 国防科技大学学报,2012,34(4):85-89

[15] Rothaus O S. On bent functions[J]. Journal of Combinatorial Theory, Series A, 1976, 20: 300-305

[16] 温巧燕,钮心忻,杨义先. 现代密码学中的布尔函数[M]. 北京:科学出版社,2000

[17] Li W, Wang Z, Huang J. The e-derivative of boolean functions and its application in the fault detection and cryptographic system[J]. Kybernetes, 2011, 40(5/6): 905-911

[18] Huang J L, Wang Z. The relationship between correlation immune and weight of H Boolean function[J]. Journal on Communications, 2012, 33(2): 110-118(in Chinese)
黄景廉,王卓. H 布尔函数的相关免疫性与重量的关系[J]. 通信学报,2012,33(2):110-118

[19] Zhao M L. Method of detecting special logic function based on Boolean e-derivative[J]. Journal of Zhejiang University (Science Edition), 2014, 41(4): 424-426(in Chinese)
赵美玲. 基于布尔 e 导数的特殊逻辑函数检测方法[J]. 浙江大学学报(理学版),2014,41(4):424-426