面向服务的可信软件体系结构代数模型

赵会群 孙 晶
(北方工业大学计算机系 北京 100144)

摘 要 针对面向服务体系结构(Service Oriented Architecture,SOA)在形式化和可信属性建模方面的不足,用代数学方法对服务、服务组合以及服务体系结构的属性和行为特征进行抽象,把服务组合解释成组件"运算"实现,并结合进程代数中算子概念,定义了多种服务组合运算,从而建立SOA的代数模型.在代数模型基础上,进一步对SOA可信属性建模,提出多种SOA可信范式,为可信SOA软件设计提供理论支持.最后介绍了一个应用案例.

关键词 服务;服务组合;服务体系结构;抽象代数;进程代数
中图法分类号 TP311 **DOI号**: 10.3724/SP.J.1016.2010.00890

An Algebraic Model of Service Oriented Trustworthy Software Architecture

ZHAO Hui-Qun SUN Jing
(*Department of Computer Science, North China University of Technology, Beijing* 100144)

Abstract This paper focuses on the lacks at methodology of describing formal model and trustworthy attributes of the Service Oriented Architecture (SOA in short), abstracts and describes the attributes and the behaviors of Service, Service combination and SOA by algebraic method. By understanding the Service Combination as operation, that is a new ideal, and extending the calculus in Process Algebraic, some Service combination operators are defined and then a formal algebraic model of SOA is proposed. Based this model some trustworthy attributes are analyzed and a few Trustworthy Normal Formats is proposed. All above viewpoints construct theoretical footstone for designing trustworthy SOA. At last a case study is introduced to explain how the above algebraic model can be used.

Keywords service; service combination; service oriented architecture; abstract algebra; process algebra

1 引 言

可信软件是指正确、安全和可靠的软件[1].目前软件的可靠性和安全性不能令人满意.软件设计缺陷、软件系统被恶意攻击都给计算机系统的正常运行带来了不良的影响.如何在软件的开发和运行中保证软件具有高可信性,已成为软件理论和技术的重要研究方向[1].

所谓面向服务的软件体系结构(Service Oriented Architecture,SOA)是一种充分利用Internet技术来满足企业对不断增长的业务运营模式需求的应

收稿日期:2008-07-08;最终修改稿收到日期:2010-01-25.本课题得到国家"八六三"高技术研究发展计划项目基金(2007AA010302)、北京市属高等学校人才强教深化计划学术创新团队建设计划、北京市教委科研计划项目基金(KM200710009009)资助.**赵会群**,男,1960年生,博士,教授,现从事可信网络和可信软件方面的教学与研究工作.E-mail: zhaohq6625@sina.com. **孙 晶**,女,1968年生,硕士,副教授,现从事程序设计与软件测试方面的教学与研究工作.

用框架，该模式需要具有灵活、安全和无逢地处理异构、异质的内、外资源的能力①. IBM、Microsoft、Gartner 等 IT 领军企业，先后提出了自己的 SOA 解决方案②③④，OASIS（Organization for the Advancement of Structured Information Standards，OASIS）提出了 SOA 参考模型 1.0[2]，OSOA（Open Service Oriented Architecture）给出了 SOA 编程模型 SCA（Service Component Architecture，SCA）等⑤. SOA 研究已经成为热点问题.

最能够体现软件服务质量的指标之一就是可信. 而有关可信 SOA 的基础理论和相关技术研究还有较大的研究空间，表现在以下几个方面：

（1）可信 SOA 建模目前仍处在经验阶段. OASIS 的参考模型给出与服务相关的概念以及服务过程中与用户的交互描述[2]. 主要的贡献是明确了 SOA 中的基本概念以及概念间的相互关系. OSOA 给出的 SCA 模型，对软件体系结构中组件和连接器概念进行了扩展，提出组件不仅是一个计算逻辑，也包括计算逻辑的开发与支撑环境；连接器不仅是对组件连接的参数传递，同时也包括通过网络交互的协议等. SCA 的贡献是给出了 SOA 软件的实现范例. 文献①②③也提出了自己的 SOA 参考模型，对 SOA 建模理论研究有所贡献. 但上述研究并没有深入讨论可信问题，没有对服务、交互和行为属性进一步分类，没有给出形式化描述，没有建立严格的理论体系，这对可信 SOA 研究的指导意义不明显. 文献[5]用进程代数对 Web 服务的合成与替换建模，并提出替换一致性检测算法；文献[6]采用进程代数对 Web 服务建模，并导出性能模型. 上述研究虽然都采用进程代数作为建模工具，但都没有结合 SOA 特性对进程代数进行扩展. 另外，上述研究是针对 Web 服务的建模，而 Web 服务仅是 SOA 的一种表现形式，所以可信 SOA 的建模型问题还没有很好地解决.

（2）可信软件体系结构建模研究有待深入. SOA 作为软件体系结构（Software Architecture，SA）的特例，其可信属性建模应属于 SA 建模范畴. 文献[7]综述了 10 年来 SA 的研究进展，从软件生命周期各个阶段诠释了 SA 的作用. 不仅如此，还指出："随着网构软件的复杂程度增加，进一步加大了对网构软件结构的理解、分析和开发的难度；如何界定 SA 在网构软件整个生命周期中的角色，将是一个值得关注和研究的课题；其中主要研究点包括：对网构 SA 描述与分析的方法、基于体系结构的网构软件的质量属性保障机制等".

目前，对 SA 建模方法的研究还是集中在软件体系结构描述语言（Architecture Description Language，ADL）方面. 较著名的 ADL 有 ACME、C2、Darwin、Rapide 和 Wright 等[8]. UML（Union Model Language，UML）是广泛使用的软件系统建模语言，虽然对 SA 的描述能力不足，经过改进后可以作为一种 ADL[9]. Taylor 提出了一种可信 SA 描述语言 xADL，通过明确主题、资源、特权、安保和策略等相关概念，描述访问控制等安全事务[10]. ADLs 采用形式化的方法描述 SA，但并未对组件、连接器和 SA 的属性和行为给出一致的定义和描述. 文献[8]提出一个具有普遍意义的 SA 框架，但未对框架中的组件、连接器以及 SA 的属性和动态行为给出抽象化的描述，未对不同的连接方式进行分类，未讨论各类 SA 间的相互关系. 从系统分析角度，SA 建模方法还应该提供对质量属性的描述能力，而上述 ADLs 不具备这些功能. Taylor 提出的 xADL 虽然对可信软件建模有指导意义，但并没有涉及 SOA 的概念，也存在着形式化描述不够、对组件连接方式分类不明确等问题. 为此，用数学理论建立完备的可信 SOA 模型，并在此基础上研究可信 SOA 的描述、检测和评价等问题，还有较大的研究空间.

本文从软件体系结构建模入手，用进程代数对可信 SOA 的建模方法进行研究，通过对传统的进程代数扩展，提出多种服务组合运算，建立 SOA 的代数模型，并在此基础上研究 SOA 的可信设计问题，为可信 SOA 设计提供理论支持.

2　SOA 代数模型

下面用进程代数解释服务组件、服务组合以及服务体系结构等概念，最后给出 SOA 代数模型.

---

① Patrick F. Carey Bernard W. Gleason. Solving the Integration Issue- Service-Oriented Architecture. http://www.zdnet.co.uk/tsearch/Service-Oriented+Architecture.htm. Feb. 2006

② David Sprott and Lawrence Wilkes. Understanding Service-Oriented Architecture，CBDI Forum. http://www.microsoft.com/china/MSDN/library/architecture/, January 2004

③ Microsoft Whitepaper. Enabling Real World SOA through the Microsoft Platform. http://msdn2.microsoft.com/en-us/architecture/aa948857.aspx. 2007. Sep. 2007

④ Yefim V Natis. Service-Oriented Architecture Scenario. Gartner Group. http://www.gartner.com/resources/114300/114358/114358.pdf, 2003

⑤ SCA Service Component Architecture-SCA Assembly Model V1.00. http://www.osoa.org/display/Main/. May, 2007

2.1 服务组件

把服务解释成:“一个具有服务能力的、可按照一定的策略和特性(如可信性)组合的组件集合,这些组件可以是一个正在执行的程序代码段、也可以是程序运行的系统环境.”下面分别给出服务组件、服务组合和服务体系结构的形式化描述以及服务同态、同构概念.

定义 1. 服务组件(简称组件)可以是一段程序代码,可以是实现程序代码的语言,可以是语言的运行环境等计算单元,也可以是数据、数据的处理结果的数据单元.它由组件接口和组件实现模块组成.组件接口是组件与外部接触点的集合,即$\langle Port_1, Port_2, \cdots, Port_n\rangle$,而每一个接触点 $Port_i$ 是一个八元组$\langle ID, Publ_i, Extn_i, Priv_i, Beha_i, Msgs_i, Cons_i, Non\text{-}Func_i\rangle$.其中:

ID 是组件的标识;

$Publ_i$ 是第 i 个接触点提供给环境或其它组件的功能(活动)集合,活动执行记为 x;下同.

$Extn_i$ 是第 i 个接触点运行所需环境或其它组件的功能(活动)集合;

$Priv_i$ 是第 i 个接触点私有功能(活动)集合;

$Beha_i$ 是第 i 个接触点行为语义描述,由谓词表达式构成;

$Msgs_i$ 是第 i 个接触点所有消息的集合,由事件标示构成;

$Cons_i$ 是对第 i 个接触点行为约束,它通常包括组件运行的初始条件、前置条件和后置条件,有时为了明确表示这 3 个条件可把它写成 $Cons(init, pre\text{-}cond, post\text{-}cond)$,$init$、$pre\text{-}cond$ 和 $post\text{-}cond$ 分别表示初始条件、前置条件和后置条件的集合;

$Non\text{-}Func_i$ 是组件第 i 个接触点非功能说明,包括组件的服务策略、合同、安全性、可靠性说明等.

下面给出组件相等和组件进化概念.

定义 2. 设 A、B 是论域 $Dom(U)$ 中的两个组件,U 为组件集合,下同.如果 A、B 满足下列条件

$$\begin{cases}(1)\ Dom(A)=Dom(B)\\(2)\ Publ(A)=Publ(B)\\(3)\ Extn(A)=Extn(B)\\(4)\ Priv(A)=Priv(B)\\(5)\ Beha(A)\Leftrightarrow Beha(B)\\(6)\ Msgs(A)=Msgs(B)\\(7)\ Cons(A)\Leftrightarrow Cons(B)\\(8)\ Non\text{-}Func(A)\Leftrightarrow Non\text{-}Func(B)\end{cases}\quad(1^*)$$

则称 A、B 相等,记作 $A=B$.‘$\Leftrightarrow$’为逻辑等价,‘$\Rightarrow$’为永真蕴涵,下同.

定义 3. 设 A 和 B 是论域 $Dom(U)$ 中的两个服务组件,如果 A 和 B 满足下列条件,则称 B 是 A 的一个进化,记为 $Evolve(B,A)$.

$$\begin{cases}(1)\ Dom(A)=Dom(B)\\(2)\ Publ(B)\supseteq Publ(A)\\(3)\ Extn(B)\subseteq Extn(A)\\(4)\ Priv(B)\subseteq Priv(A)\\(5)\ Beha(B)\Rightarrow Beha(A)\\(6)\ Msgs(B)=Msgs(A)\\(7)\ (Cons(B)=Cons(A))\ \text{or}\\\quad(Cons(B)\Rightarrow Cons(A))\\(8)\ (Non\text{-}Func(B)=Non\text{-}Func(A))\ \text{or}\\\quad(Non\text{-}Func(B)\Rightarrow Non\text{-}Func(A))\end{cases}\quad(2^*)$$

2.2 服务组合

下面结合服务软件特点,对进程代数中的算子进行扩展,把服务组合解释成服务组件连接运算的实现.

定义 4. 设 A、B 是论域 $Dom(U)$ 中的两个组件,若 $\exists x\in Extn(A)\wedge\exists y\in Publ(B)$使得

$$[pre\text{-}cons(A)\wedge pre\text{-}cons(B)\wedge(x\Rightarrow y)]\wedge[Msgs(x)\rightarrow Msgs(y)],$$

即组件 A 通过发送一个消息“激发”组件 B 中的 $Publ(B)$来实现功能需求,就称组件 A,B 进行了一次“激发”运算,记作 $A\oplus B$.特别地把 $A\oplus B$ 记为 $x\oplus y$.

$A\oplus B$ 仍然是一个组件,它满足下列性质:

$$\begin{cases}(1)\ Dom(A\oplus B)=Dom(A)\cup Dom(B)\\(2)\ Publ(A\oplus B)=Publ(A)\cup Publ(B)\\(3)\ Extn(A\oplus B)=Extn(A)\cup Extn(B)\\(4)\ Priv(A\oplus B)=Priv(A)\cup Priv(B)\\(5)\ Beha(A\oplus B)\Leftrightarrow Beha(A)\wedge Beha(B)\\(6)\ Msgs(A\oplus B)=(Msgs(A)\cup Msgs(B))\\(7)\ Cons(A\oplus B)\Leftrightarrow Cons(A)\wedge Cons(B)\\(8)\ Non\text{-}Func(A\oplus B)\Leftrightarrow Non\text{-}Func(A)\wedge\\\quad Non\text{-}Func(B)\end{cases}\quad(3^*)$$

定义 5. 设 A、B 是论域 $Dom(U)$ 中的两个组件,若 $\exists x\in Extn(A)\wedge\exists y\in Publ(B)$使得

$$[pre\text{-}cond(A)\wedge pre\text{-}cond(B)\wedge(x\Rightarrow y)]\wedge[y\in Priv(A)],$$

即组件 A 通过“使用”组件 B 中的 $Publ(B)$来实现功能需求,就称组件 A,B 进行了一次“使用”运算,记作 $A\otimes B$.特别地把 $A\otimes B$ 记为 $x\otimes y$.

$A\otimes B$ 仍然是一个组件,它满足下列性质:

$$\begin{cases}(1)\ Dom(A\otimes B)=Dom(A)\cup Dom(B)\\(2)\ Publ(A\otimes B)\subseteq Publ(A)\cup Publ(B)\\(3)\ Extn(A\otimes B)\subseteq Extn(A)\cup Extn(B)\\(4)\ Priv(A\otimes B)=Priv(A)\cup Priv(B)\\(5)\ Beha(A\otimes B)\Rightarrow Beha(A)\wedge Beha(B)\\(6)\ Msgs(A\otimes B)=(Msgs(A)\cup Msgs(B))\\(7)\ Cons(A\otimes B)\Rightarrow Cons(A)\wedge Cons(B)\\(8)\ Non\text{-}Func(A\otimes B)\Rightarrow Non\text{-}Func(A)\wedge\\ \quad Non\text{-}Func(B).\end{cases}\quad(4^*)$$

“使用”和“激发”运算是最基本的服务组合运算. 在不需要区分二者的环境下，可以把二者统称为“调用”运算，记为 $A \copyright B$. 可以证明“调用”运算满足结合率[13].

定义 6. 设 A、B 是论域 U 中的两个组件，若 $\exists x\in Extn(A)$，对 $\forall y\in Publ(B)$，若 $[(x\Rightarrow y)\wedge((pre\text{-}cond(A)\wedge pre\text{-}cond(B))]\wedge[Msgs(y)\in Msgs(A)]$，则称 A 与 B 进行一次带有“反馈”的调用运算.

显然，反馈运算依赖“使用”或“激发”运算，所以反馈本身一定与这两种运算一起构成表达式. 如 A 激发 B 后把结果反馈给 A. 把反馈运算记为 $A \copyright B\uparrow$，特别地把 $A \copyright B\uparrow$ 记为 $x \copyright y\uparrow$.

定义 7. 设 A、B 是论域 U 中的两个组件，若 $\exists x\in Publ(A)$，$\exists y\in Extn(B)$ 使得 $[(pre\text{-}cond(B))\wedge(y)\Rightarrow((pre\text{-}cond(A))\wedge(x)]$，反之 $\exists x\in Extn(A)$，$\exists y\in Publ(B)$ 使得 $[(pre\text{-}cond(A))\wedge(x)\Rightarrow((pre\text{-}cond(B))\wedge(y)]$. 则称组件 B 与组件 A 协同运算，记作 $A\Theta B$. $H=pre\text{-}cond(A)\cap pre\text{-}cond(B)$ 称为协同条件集. 特别地把 $A\Theta B$ 记为 $x\Theta y$.

显然，“协同”运算满足交换率.

这里 $x\Leftrightarrow y$ 并没有限定协同运算的交互方式，所以协同运算依赖于“使用”或“激发”运算.

定义 8. 设 A、B 是论域 U 中的两个组件，若 $\exists x\in Publ(A)$，$\exists y\in Extn(B)$ 可以有 $[(pre\text{-}cond(B))\wedge(y)\Rightarrow((pre\text{-}cond(A))\wedge(x)]$ 成立，反之 $\exists x\in Extn(A)$，$\exists y\in Publ(B)$ 可以有 $[(pre\text{-}cond(A))\wedge(x)\Rightarrow((pre\text{-}cond(B))\wedge(y)]$ 成立，但服务过程中 $pre\text{-}cond(A)\cap pre\text{-}cond(B)=\varnothing$，则称组件 A 与组件 B 并行，记作 $A\parallel B$. 特别地把 $A\parallel B$ 记为 $x\parallel y$.

显然，并行运算是协同运算的特例，并都满足交换率.

定义 9. 设 A、B 是论域 U 中的两个组件，$\exists x_1\in Publ(A)$，$\exists y_1\in Extn(A)$，$\exists x_2\in Publ(B)$，$\exists y_2\in Extn(B)$，若 $[(post\text{-}cond(A)\wedge pre\text{-}cond(B))\wedge(y_1\Rightarrow x_2)]\Leftrightarrow[(post\text{-}cond(B)\wedge pre\text{-}cond(A))\wedge(y_2\Rightarrow x_1)]$，则称 A 与 B 重复，记为 $A\cdot B$. 特别地，当 $A=B$ 时，称 A 重复执行，记为 $x_1\cdot A$，简记：$\cdot A$.

“反馈”与“重复”的区别在于，“反馈”只是结果的返回，而“重复”需要重复执行原组件方法 x. 显然，“反馈”与“重复”都依赖“使用”或“激发”两种运算. 可以证明“重复”满足交换率.

定义 10. 设 A、B 是论域 U 中的两个组件，若 $\exists x\in Publ(A)$，$\exists y\in Publ(B)$ 使得 $[pre\text{-}cond(A)\wedge(x)]\vee[pre\text{-}cond(B)\wedge(y)]$，则称组件 A 与组件 B 选择执行，记为 $A+B$. 特别地把 $A+B$ 记为 $x+y$.

定义 6～10 中的运算也有与式(3*)类似的性质. 这里略.

可以证明“选择”运算满足交换率. “反馈”、“重复”、“激发”与“使用”、“协同”与“并行”运算对“选择”运算满足分配率. 下面仅就“重复”对“选择”的分配律加以证明，其它证明略.

定理 1. “重复”运算对“选择”运算满足分配律，即

$$(A+B)\cdot C=A\cdot C+B\cdot C \qquad (1)$$

证明. 要证明式(1)成立，只要证明等式的左边和等式右边满足定义 2 中的 8 个条件即可. 为此，我们有代表性的证明第 2 个和第 5 个条件成立，其它证明同理.

首先，对 $\cdot$ 和 $+$ 运算，都有 $Publ(AOP_iB)=Publ(A)\cup Publ(B)$ 性质，所以对 $\forall x\in Publ((A+B)\cdot C)$，有 $x\in Publ(A)\vee x\in Publ(B)\vee x\in Publ(C)$，可得 $x\in Publ(A\cdot C+B\cdot C)$. 同理，对 $\forall y\in Publ(A\cdot C+B\cdot C)$ 有 $y\in Publ((A+B)\cdot C)$. 因此，有 $Publ((A+B)\cdot C)=Publ(A\cdot C+B\cdot C)$，定义 2 的性质(2)成立.

首先，对 $\cdot$ 运算，有 $Beha(A\cdot B)\Leftrightarrow Beha(A)\wedge Beha(B)$ 性质；而对 $+$ 运算，有 $Beha(A+B)\Leftrightarrow Beha(A)\vee Beha(B)$ 性质. 由 $Beha((A+B)\cdot C)\leftrightarrow(Beha(A+B))\wedge Beha(C)\leftrightarrow(Beha(A)\vee Beha(B))\wedge Beha(C)\leftrightarrow(Beha(A)\wedge(Beha(C))\vee(Beha(B)\wedge(Beha(C))\leftrightarrow Beha(A\cdot C+B\cdot C)$，则 $Beha((A+B)\cdot C)\Leftrightarrow Beha(A\cdot C+B\cdot C))$. 定义 2 的性质(5)成立. 证毕.

以上借鉴进程代数中算子概念，对组合进行分类，下面进一步讨论服务组合相关的概念.

定义 11. 服务组合是组件运算的实现. 它是

一个六元组〈*ID*, *Role*, *Beha*, *Msgs*, *Cons*, *Non-Func*〉. 其中：

ID 是组合的标识.

Role 为服务组合与组件的交互点的集合，每个 *Role*=〈*Id*, *Action*, *Event*, *LConstrains*〉组成. 其中：*Id* 是 *Role* 的标识；*Action* 是 *Role* 活动的集合，每个活动由事件的连接(谓词)组成；*Event* 是 *Role* 产生的事件集合；*LConstrains* 是 *Role* 的约束集合. 我们把 *Role* 从组合的其它属性分开来描述的目的是突出组合的多态性，即一个组合可同时与多个组件组合.

Msgs 是组合中各 *Role* 中事件产生的消息的集合.

Beha 是组合行为的语义描述.

Cons 是组合约束的集合，它包括组合的初始条件、前置条件和后置条件，有时为了明确表示这 3 个条件可把它写成 *Cons*(*init*, *pre-cond*, *post-cond*)，*init*,、*pre-cond* 和 *post-cond* 分别表示初始条件、前置条件和后置条件的集合.

Non-Func 是组合的非功能说明，包括按何种策略、合同、安全性和可靠性组合等.

2.3 服务体系结构代数模型

下面给出服务体系结构的定义.

定义 12. 设论域为 U，

(1) 组件是一个服务体系结构；

(2) 组合是一个服务体系结构；

(3) 由组件经有限次组合(组件运算)后是服务体系结构.

服务体系结构，记为 $S=\langle C,O\rangle$，简称服务. 其中 C 表示组成服务的组件集合，O 表示组件运算的集合.

由定义 12 可得服务体系结构的性质如下：

(1) 封闭性. 即组件与组件、组件与服务、服务与服务组合后仍是一个服务.

(2) 层次性. 即服务可由组件组合而成，而服务又可以再经过组合组成新的更大的服务.

(3) 可扩充性. 即一个满足条件的新组件可以通过组合加入到服务中.

从组合是组件运算实现角度，可以进一步证明 SOA 对任意一个运算构成代数系统.

定理 2. 设 $S=\langle C,O\rangle$ 是服务，则 S 对 O 中的每一个组合运算都构成代数系统.

证明. 由服务组合运算的封闭性可得定理 2 的正确性.

为此，把 $S=\langle C,O\rangle$ 称为 SOA 的代数模型，也称 SOA 的代数表达式.

在 SOA 代数模型定义基础上，可以进一步利用代数学方法挖掘 SOA 的代数性质，为 SOA 基础理论研究奠定基础.

定义 13. 设 $S_1=\langle C_1,O_1\rangle$，$S_2=\langle C_2,O_2\rangle$ 是两个不同服务，若 $\forall x\in C_1$ 总有 $\exists y\in C_2$，使得 y 与 x 对应，则称 S_1 与 S_2 之间存在着一个映射，记为 $f:S_1\rightarrow S_2$，或 $y=f(x)$. 若映射是满射，则称服务间映射为满射；若映射是一对一的，则称服务间为一一映射.

定义 14. 设 $S_1=\langle C_1,O_1\rangle$，$S_2=\langle C_2,O_2\rangle$ 是两个服务，f 是 S_1 到 S_2 的一个映射，若对 $\forall x\in C_1$ 和 $\forall y\in C_1$ 有 $f(xOP_iy)=f(x)OP_jf(y)$，其中 $OP_i\in O_1$、$OP_j\in O_2$. 则称 f 为从 S_1 到 S_2 的同态，也称 S_1 与 S_2 同态；若 f 是单射，则称 f 是从 S_1 到 S_2 的单一同态；若 f 是一一映射，则称 f 是从 S_1 到 S_2 的同构，也称 S_1 与 S_2 同构.

用同态与同构概念可以进一步描述基于 SOA 软件的进化性、服务之间的关系等概念，这里仅给出后者.

定义 15. 给定两个服务 $S_1=\langle C_1,O_1\rangle$、$S_2=\langle C_2,O_2\rangle$，构造一个新的服务 $S_1\times S_2=\langle C_1\times C_2, OP_k\rangle$. 其中 $C_1\times C_2$ 是组件集合的笛卡尔乘积，而 OP_k 定义成对 $\forall x_1,x_2\in C_1$ 和 $\forall y_1,y_2\in C_2$ 有 $\langle x_1,y_1\rangle OP_k\langle x_2,y_2\rangle=\langle x_1\,OP_i\,x_2,\ y_1\,OP_j\,y_2\rangle$，$OP_i\in O_1$，$OP_j\in O_2$. 称 $S_1\times S_2$ 是 S_1 到 S_2 的积结构，而 S_1 和 S_2 是 $S_1\times S_2$ 的因子，这里的 OP_i、OP_j 和 OP_k 运算是任意组件连接运算.

在定义 15 的基础上可以定义服务间的关系.

定义 16. 对 $\forall x_1,x_2\in C_1$ 和 $\forall y_1,y_2\in C_2$，在定义 15 中，所有满足 $\langle x_1OP_ix_2\rangle$ 的 (x_1,x_2) 的任意一个子集合称为运算 OP_i 的一个关系；所有满足 $\langle y_1OP_jy_2\rangle$ 的 (y_1,y_2) 的任意一个子集合称为运算 OP_j 的一个关系.

3 SOA 可信属性建模

可信是评价软件服务质量重要的指标之一. 基于 SOA 代数模型，讨论如何通过建立 SOA 的可信范式对 SOA 可信属性建模.

3.1 可信性模型

定义 17. 设 $S=\langle C,O\rangle$ 是一个服务，若 $\exists Y\in C$；$\forall X\in C$ 都有 XOp_iY，则称 Y 为服务 S 的候选核(Candidate Core)，若候选核多于一个时，可选定其

中一个作为主核(Primary Core). 其中，Op_i 为组合运算.

定义 18. 设 $S=\langle C,O\rangle$ 是一个服务，若任意一个组件在执行时，都至少与另一个组件发生组合，则称 A 满足第一范式，记为 1NF.

显然，1NF 是 SOA 软件设计的最基本的要求.

定义 19. 设 $S=\langle C,O\rangle$ 是一个服务，若 A 中存在核，则称 A 满足第二范式. 记为 2NF.

结合组合运算定义可以证明，“使用”运算组合是可信性最低的一种组合.

定理 3. 设 $S=\langle C,O\rangle$ 是一个服务，若对 $\forall X,Y\in C$ 有 XOp_iY，Op_i 是“使用”组合，则 XOp_iY 的可信性依赖组件 Y 的可信性，进一步有 X 的可信性依赖 Y.

证明. 根据“使用”组合运算定义，有 $\exists x\in Extn(X)\wedge\exists y\in Publ(Y)$ 使得

$$(x\Rightarrow y)\wedge(pre\text{-}cond(X)\wedge pre\text{-}cond(Y))\wedge(Beha(X)\Leftrightarrow Beha(Y))$$

成立；若 y 失效，即 $Beha(Y)$ 为假，则 $Beha(X)\Leftrightarrow Beha(Y)$ 为假，$Beha(X)$ 为假，XOp_iY 组合失效. 因此，XOp_iY 和 X 的可信性依赖组件 Y 的可信性. 这里的 y 失效是指被恶意攻击或软件故障. 证毕.

定义 20. 设 $S=\langle C,O\rangle$ 是一个服务，若 A 满足第一范式或第二范式，并且 A 中的组合运算均为“激发”组合，则称 A 满足可信基础范式，记为 RNF.

第一范式是典型的分布式软件结构，而第二范式是典型的集中式结构. 这两种范式的可信性有所不同，可以证明在非“使用”组合下，第一范式比第二范式的可信性要高.

定理 4. 设 $S=\langle C,O\rangle$ 是一个服务，若 S 中有核 Y，则 S 的可信性依赖 Y 的可信性.

证明. 设 $S=\langle C,O\rangle$ 是一个服务. 根据核的定义，$\exists Y\in C$，$\forall X\in C$ 都有 XOp_iY. 若 y 失效，则 XOp_iY 失败. 因此，XOp_iY 的可信性依赖组件 Y 的可信性. 证毕.

定义 21. 设 $S=\langle C,O\rangle$ 是一个服务，若 S 满足第一范式和 RNF 范式，则称 S 满足可信一型范式，记为 R1NF；若 S 满足第二范式和 RNF 范式，则称 S 满足可信二型范式，记为 R2NF.

下面讨论基于“协同”和“并行”运算的可信范式模型.

引理 1. 任何一个程序可以只用顺序、分支和循环结构来编码[11].

定理 5. 任意一个 $S=\langle C,O\rangle$ 的代数表达式都可以写成下面的标准型.

$$S=a_1©C_1\uparrow+a_2©C_2\uparrow+\cdots+a_k©C_m\uparrow \quad (2)$$

或简写成

$$S=a_1C_1\uparrow+a_2C_2\uparrow+\cdots+a_kC_m\uparrow \quad (3)$$

其中 $a_i\in C_j$；$C_j\in C$，$i=1,2,\cdots,K$，$j=1,2,\cdots,M$，i,j 可能不等，©是“使用”或“激发”运算.

证明.

用归纳法证明. 假设 $k=0$ 时成立，此时 S 只有一个组件构成，$S=a_1C_1\uparrow$ 表示该组件的启动、运行与反馈的状态转换.

现假设 $K=n-1$ 时成立，去证明 $K=n$ 时成立. 即 $S=a_1C_1\uparrow+a_2C_2\uparrow+\cdots+a_{n-1}C_{n-1}\uparrow$（$a_i\in C_j$；$C_j\in C$，即 $i,j=1,2,\cdots,K$，i、j 可能不等）成立，去证明 $S=a_1C_1\uparrow+a_2C_2\uparrow+\cdots+a_{n-1}C_{n-1}+a_nC_n\uparrow$.

(1)“选择”运算时显然成立.

$$S=a_1C_1\uparrow+a_2C_2\uparrow+\cdots+a_{n-1}C_{n-1}\uparrow+a_nC_n\uparrow$$

（设 $a_n\in C_j$，$C_n\in C$）.

(2)证明“重复”运算成立. 设 $a_n\in C$，有 $S=(a_1C_1\uparrow+a_2C_2\uparrow+\cdots+a_{n-1}C_{n-1}\uparrow)\cdot a_nC_n\uparrow$；由↑满足对＋和·的分配律得，$S=((a_1C_1+a_2C_2+\cdots+a_{n-1}C_{n-1})\cdot a_nC_n)\uparrow$；又由于由·满足对＋的分配率，则 $S=(a_1C_1\cdot a_nC_n+a_2C_2\cdot a_nC_n+\cdots+a_{n-1}C_{n-1}\cdot a_nC_n)\uparrow$；设 $x_i\in Publ(C_i)$，则 $S=(a_1x_1\cdot a_nC_n+a_2x_2\cdot a_nC_n+\cdots+a_{n-1}x_{n-1}\cdot a_nC_n+C)\uparrow$；构造 $b_i=a_ix_i\cdot a_n$，$b_nC_n=C$（一个空调用），则 $S=b_1C_1\uparrow+b_2C_2\uparrow+\cdots+b_{n-1}C_{n-1}\uparrow+b_nC_n\uparrow$，即定理在“重复”运算下成立.

(3)证明“顺序”运算成立. 设 $a_n\in C,C_n\in C$，©为⊕或⊗运算，有 $S=(a_1C_1\uparrow+a_2C_2\uparrow+\cdots+a_{n-1}C_{n-1}\uparrow)©a_nC_n\uparrow$；由↑满足对©的分配律得，$S=((a_1C_1+a_2C_2+\cdots+a_{n-1}C_{n-1})©a_nC_n)\uparrow$；由©对＋运算满足分配率，有 $S=(a_1C_1©a_nC_n+a_2C_2©a_nC_n+\cdots+a_{n-1}C_{n-1}©a_nC_n)\uparrow$；设 $x_i\in Publ(C_i)$，则 $S=(a_1x_1©a_nC_n+a_2x_2©a_nC_n+\cdots+a_{n-1}x_{n-1}©a_nC_n+C)\uparrow$；构造 $b_i=a_ix_i©a_n$、$b_nC_n=C$（一个空调用），则 $S=b_1C_1\uparrow+b_2C_2\uparrow+\cdots+b_{n-1}C_{n-1}\uparrow+b_nC_n\uparrow$，即定理在“顺序”运算下成立.

根据引理 1 得定理 5 成立. 证毕.

推论 1. 在只考虑服务组件内部行为过程时，任意一个 $S=\langle C,O\rangle$ 的代数表达式都可以写成下面的标准型.

$$S=a_1\cdot C_1+a_2\cdot C_2+\cdots+a_k\cdot C_k \quad (4)$$

或简写成

$$S=a_1C_1+a_2C_2+\cdots+a_kC_k \quad (5)$$

其中，$a_i \in C_i (i=1,2,\cdots,K)$.

证明. 与定理 5 的证明过程相似，略.

在只考虑服务组件内部活动时，有 $a_i \in C_i, i=1,2,\cdots,K$，“©”运算条件均满足“·”运算条件，所以 $a_i ©C_i \uparrow$ 可以通过 $a_i \cdot C_i \uparrow$ 实现，即 $a_i ©C_i \uparrow = a_i \cdot C_i \uparrow$；又由于只在组件内部活动，所以 $a_i \cdot C_i \uparrow = a_i \cdot C_i$. 因此，推论 1 成立.

标准型(4)的物理意义为：a_i 是行为明确的活动，如安全性确认等可信活动等，所以可以把 a_i 看成是常量；而 C_i 为服务功能组件，可以看成是变量. 从可信角度，如果 a_i 是为了实现可信性保证的活动，则标准型(4)可以看成是一种可信范式. 把标准型(4)称为第三可信范式，记为 3NF.

定义 22. 设 $S=\langle C,O\rangle$，$S=a_1 \cdot C_1 + a_2 \cdot C_2 + \cdots + a_k \cdot C_k$. 其中 $a_i \in C_i$；$C_i \in C$. 如果 a_i 是明确的，为了实现可信性保证的活动，则称 $S=a_1 \cdot C_1 + a_2 \cdot C_2 + \cdots + a_k \cdot C_k$ 为第三可信范式，记为 3NF；a_i 称为可信常量.

下面讨论“协同”组合的可信范式.

根据 CCS 中组合算子的扩展规则和隐藏内部活动规则[12]以及提出的协同运算定义有以下规则.

并行组合运算扩展规则. 设 $F=a_1 \cdot f_1 + a_2 \cdot f_2 + \cdots + a_k \cdot f_k$；$G=b_1 \cdot g_1 + b_2 \cdot g_2 + \cdots + b_k \cdot g_k$ 是两个服务组件，那么

$$F\Theta G = (a_1 f_1 \Theta G) + \cdots + (a_m f_m \Theta G) + (b_1 F \Theta g_1) + \cdots + (b_n F \Theta g_k) + \Sigma (a_i \Theta b_j) f_k \Theta g_k \quad (6)$$

其中 a_i、b_j 为协同常量对，$a_i \Theta b_j$ 记为 ε；$a_i \cdot f_i$ 简写为 $a_i f_i$；$\Sigma (a_i \Theta b_j) \cdot f_k \Theta g_k$ 表示多项选择.

上述规则表明，两个协同的服务可以通过代数表达式描述. 下面进一步给出协同表达式的化简规则.

协同常量 $a_i \Theta b_j$ 为明确的服务可信保障行为，所以在分析服务软件的可信性时可以忽略协同可信常量，把分析重点放在没有实现可信保障的行为中，为此有下面隐藏规则.

隐藏内部协同常量规则 $\varepsilon \Theta F = F$；$a \Theta \varepsilon \Theta F = a \Theta F$.

在服务代数模型表达式中，有些项可能与所讨论的问题无关或对讨论的问题不产生干扰，所以可以把这些项消去，为此有限制规则.

限制规则. 在式(6)中，如果把常量 $a_i (i=1,2,\cdots,m)$ 或 $b_j (j=1,2,\cdots,n)$ 列入限制范围，那么式(6)的扩展式中 $a_i f_i \Theta G$ 和 $F \Theta b_j g_j$ 可以消去.

那么，哪些项可以被列入限制范围呢？要根据问题域来确定. 本文把内部协同的可信常量对列入限制范围. 下面给出定理 6.

定理 6. 设 $a_i (i=1,2,\cdots,m)$ 和 $b_j (j=1,2,\cdots,m)$ 是可信协同常量对，则在式(6)中 $a_i f_i \Theta G$ 和 $F \Theta b_j g_j$ 可以消去.

证明. 先对 $a_i f_i \Theta G$ 证明. 设 $G = b_1 g_1 + b_2 g_2 + \cdots + b_k g_k$，由“Θ”运算定义，则一定存在一个常量 b_j 是 a_i 的可信协同常量伴侣，使得 $a_i f_i \Theta G = a_i f_i \Theta b_j g_j = (a_i \Theta b_j) \cdot f_i \Theta g_j$，该项已经在式(6)中的多项选择中出现，所以该项在式(6)中可以消去. 同理可证 $F \Theta b_j g_j$ 也可以消去. 证毕.

为了分析服务系统的可信性，下面给出两个相关概念——死锁与活锁.

定义 23. 如果一个服务无限循环执行某几个组件功能，则称系统出现“活琐”.

“活锁”表现为服务代数表达式中包括递归.

定理 7. “活锁”表现为服务代数表达式是一个递归表达式.

证明. 定理显然成立.

定义 24. 对一个服务软件系统，如果服务总能到达，那么称该服务系统有活动性；如果一个服务调用在执行某个组合运算后非法终止，则称该服务系统出现“死锁”，记为 ♯.

“终止”动作 ♯ 是一种正常的组件行为，但 ♯ 出现在需要继续执行的调用行为中时，就会引起“死锁”. 根据定义 24，显然有定理 8 成立.

定理 8. 在有“终止”动作 ♯ 的服务软件系统中，其服务表达式中某项的常量只有 ♯.

证明. 反证法. 假设服务表达式 $S=a_1 C_1 + a_2 C_2 + \cdots + a_k C_k$，其中 a_i 为 ♯，记有 $S=a_1 C_1 + a_2 C_2 + \cdots + \sharp C_i + \cdots + a_k C_k$. 根据 ♯ 的定义，一定不存在 $x \in C_i$ 与 ♯ 发生调用运算和反馈运算，所以 $\sharp C_i = \sharp$，定理成立. 证毕.

例 1. $F=aG+bF$；$G=cG+dG$. G 无限自循环，系统出现活琐.

$X=aY+bX$；$Y=cY+dX+\sharp$. 如果网络 Y 执行常量 ♯，那么系统进入终止状态.

定义 25. 如果一个服务代数表达式中没有出现“活锁”和“死锁”，则称该服务是可信服务，所形成的代数表达式称为协同可信范式.

4 应用研究

提出的 SOA 代数模型和可信范式理论可以直

接应用到以下领域：(1) SOA 的形式化描述；(2) SOA 软件体系结构设计；(3) SOA 可信属性分析等. 由于篇幅有限，这里仅介绍在 SOA 可信属性分析中的应用. “?”表示需要对接收的报文进行可信验证.

案例分析 1. 一个可信服务协同组合过程如图 1 所示.

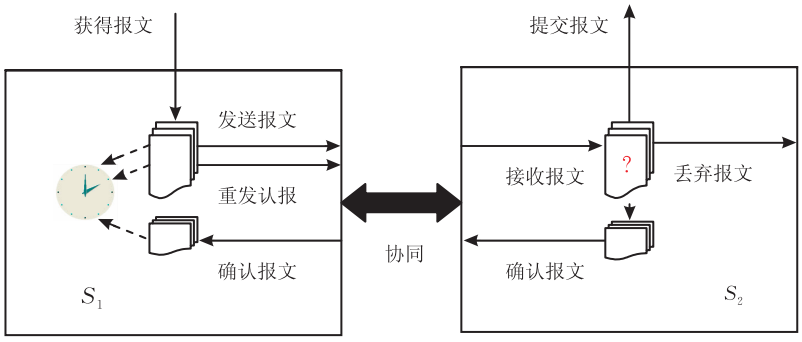

图 1　可信协同组合

S_1 行为描述：服务组件 S_1 向服务组件 S_2 发送可信协商报文(记为 A_1)，并启动时钟(记为 T)；如果在指定时间内没有得到可信确认报文(记为 B_1)，则重发可信协商报文(仍记为 A_1)，此过程重复多次，直到得到可信确认报文. S_1 的 SOA 代数模型如下：

$$S_1 = A_1 \cdot S_{11} \tag{7}$$

$$S_{11} = T \cdot A_1 \cdot S_{11} + B_1 \cdot S_1 \tag{8}$$

其中 S_{11} 是一个临时服务组件.

S_2 行为描述：服务组件 S_2 接收可信协商报文(该活动记为 A_2)；如果该报文是正确的报文，则向服务组件 S_1 发送可信确认报文(记为 B_2)；如果报文有错，则丢弃协商报文(记为 D). S_2 的 SOA 代数模型如下：

$$S_2 = A_2 \cdot (D \cdot S_2 + B_2 \cdot S_2) \tag{9}$$

在上述描述中，A_1 与 A_2、B_1 与 B_2 为两对可信确认活动对，而 D 和 T 是为了完成可信确认的辅助活动，所以均可认为是明确的可信活动. 基于第三可信范式的可信设计验证过程如下：

$$\begin{aligned}
S &= S_1 \Theta S_2 = (A_1 \cdot S_{11}) \Theta (A_2 \cdot (D \cdot S_2 + B_2 \cdot S_2)) \\
&= A_1 \cdot (S_{11} \Theta (A_2 \cdot (D \cdot S_2 + B_2 \cdot S_2)) + \\
&\quad A_2 \cdot (A_1 \cdot S_{11} \Theta (D \cdot S_2 + B_2 \cdot S_2)) + \\
&\quad \varepsilon \cdot (S_{11} \Theta (D \cdot S_2 + B_2 \cdot S_2)) \\
&\quad //\varepsilon = (A_1 \cdot A_2), (A_1 \cdot A_2) \text{是内部协同可信活动，列入有限. 根据限制规则，第一、二项可以削去. 下同.} \\
&= S_{11} \Theta (D \cdot S_2 + B_2 \cdot S_2) \\
&\quad //\text{根据式(8)，把 } S_{11} \text{ 代入上式.} \\
&= (T \cdot A_1 \cdot S_{11} + B_1 \cdot S_1) \Theta (D \cdot S_2 + B_2 \cdot S_2) \\
&= T \cdot A_1 \cdot S_{11} \Theta (D \cdot S_2 + B_2 \cdot S_2) + \\
&\quad B_1 \cdot (S_1 \Theta (D \cdot S_2 + B_2 \cdot S_2)) + \\
&\quad D \cdot ((T \cdot A_1 \cdot S_{11} + B_1 \cdot S_1) \Theta S_2) + \\
&\quad B_2 \cdot ((T \cdot A_1 \cdot S_{11} + B_1 \cdot S_1) \Theta S_2) + \\
&\quad (B_1 \Theta B_2) \cdot (S_1 \Theta S_2) \\
&\quad //\varepsilon = (B_1 \cdot B_2), (B_1 \cdot B_2) \text{是内部协同可信活动，列入有限.} \\
&= [T \cdot A_1 \cdot (S_{11} \Theta (D \cdot S_2 + B_2 \cdot S_2))] + \\
&\quad [D \cdot ((T \cdot A_1 \cdot S_{11} + B_1 \cdot S_1) \Theta S_2)] + \\
&\quad [\varepsilon \cdot (S_1 \Theta S_2)]
\end{aligned} \tag{10}$$

根据 S_1 和 S_2 描述中对活动的定义，上述 SOA 表达式中的 $T \cdot A_1$、D、ε 均为可信活动，所以式(10)满足第三可信范式的定义，是可信设计.

案例分析 2. 一个非可信服务协同组合过程如图 2 所示.

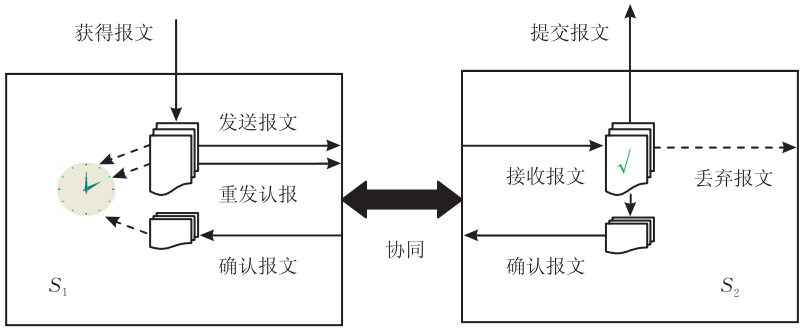

图 2　非可信协同进程

在案例 1 基础上，如果对 S_2 错误地设计成如下行为过程："不进行确认检测(在图 2 中用√表示)，直接返回可信确认报文"，其它描述不变，则行为描述如下：

$$S_2=A_2\cdot B_2\cdot(D\cdot S_2+S_2) \tag{11}$$

下面给出基于第三可信范式的非可信设计验证过程：

$$\begin{aligned}
S&=S_1\Theta S_2=(A_1\cdot S_{11})\Theta(A_2\cdot B_2\cdot(D\cdot S_2+S_2))\\
&=A_1\cdot(S_{11}\Theta(A_2\cdot B_2\cdot(D\cdot S_2+S_2))+\\
&\quad A_2\cdot(A_1\cdot S_{11}\Theta B_2\cdot(D\cdot S_2+S_2))+\\
&\quad \varepsilon\cdot(S_{11}\Theta B_2\cdot(D\cdot S_2+S_2))\\
&\quad //\varepsilon=(A_1\cdot A_2),(A_1\cdot A_2)\text{是内部协同可信活动，列入有限}\\
&=S_{11}\Theta B_2\cdot(D\cdot S_2+S_2)//\text{根据式(8)，把 } S_{11}\text{ 代入.}\\
&=(T\cdot A_1\cdot S_{11}+B_1\cdot S_1)\Theta B_2\cdot(D\cdot S_2+S_2)\\
&=T\cdot A_1\cdot S_{11}\Theta(B_2\cdot(D\cdot S_2+S_2))+\\
&\quad B_1\cdot(B_2\cdot(S_1\Theta(D\cdot S_2+S_2))+\\
&\quad D\cdot((T\cdot A_1\cdot S_{11}+B_1\cdot S_1)\Theta S_2)+\\
&\quad((T\cdot A_1\cdot S_{11}+B_1\cdot S_1)\Theta S_2)+\\
&\quad(B_1\Theta B_2)\cdot(S_1\Theta S_2)\\
&\quad //\varepsilon=(B_1\cdot B_2),(B_1\cdot B_2)\text{是内部协同可信活动对，列入有限.}\\
&=[T\cdot A_1\cdot(S_{11}\Theta(B_2\cdot(D\cdot S_2+S_2))]+\\
&\quad[D\cdot((T\cdot A_1\cdot S_{11}+B_1\cdot S_1)\Theta S_2)]+\\
&\quad[(T\cdot A_1\cdot S_{11}+B_1\cdot S_1)\Theta S_2]+\\
&\quad[\varepsilon\cdot(S_1\Theta S_2)]\\
&=[T\cdot A_1\cdot(S_{11}\Theta(B_2\cdot(D\cdot S_2+S_2))]+\\
&\quad[D\cdot(S_{11}\Theta S_2)]+[S_{11}\Theta S_2]+\\
&\quad[\varepsilon\cdot(S_1\Theta S_2)]
\end{aligned}$$

//利用式(8)把 $T\cdot A_1\cdot S_{11}+B_1\cdot S_1$ 还原成 S_{11}.

其中第 3 项$[S_{11}\Theta S_2]$没有进行可信活动就进入协同过程，所以不满第三可信范式. 因此，是一种不可信的 SOA 设计.

5 相关研究比较及结论

以上用代数方法对面向服务的可信软件体系结构进行了讨论，给出服务组件、服务组合和服务体系结构的形式化定义，提出了 SOA 的代数模型. 在该模型基础上，给出了可信范式概念，并提出多种可信 SOA 范式，为基于 SOA 的可信软件开发提供理论支持. 上述研究中一个重要观点是把服务组合解释成服务运算的实现，由此可以用进程代数进一步讨论可信 SOA 的建模问题. 上述研究与文献[5-6]的不同在于，本文对进程代数中的算子进行了扩展，并结合面向服务的软件特性对算子的物理意义加以解释. 利用进程代数讨论软件体系结构中的行为描述问题，作者在文献[13]中曾经采用过，并取得较好的研究效果. 本文的研究思路与文献[18]相似，但有较大改进，主要表现在：增加了"选择"、"协同"、"并行"、"重复"和"反馈"运算，并进一步明确了上述运算与"使用"和"激发"运算的关系. 不仅如此，在代数模型基础上，还讨论了 SOA 的可信属性建模问题；而文献[13]仅讨论了"使用"和"激发"两种运算的情况，而且没有讨论 SOA 相关内容.

可信 SOA 代数模型为进一步研究 SOA 的可信问题奠定了基础. 然而，可信软件系统是一个复杂的系统，仅凭一个模型无法描述所有的特征，我们将继续研究其它形式的建模方法. 另外，可信性测试也是可信性保证的重要研究内容之一. 由于篇幅有限，本文仅介绍了可信 SOA 的建模方法，并没有给出该模型的应用，我们将在以后的研究中，进一步表现该模型的应用价值.

参 考 文 献

[1] Chen Huo-Wang, Wang Ji, Dong Wei. High confidence software engineering technologies. Acta Electronica Sinica, 2003, 31(12): 1933-1938(in Chinese)
(陈火旺，王戟，董威. 高可信软件工程技术. 电子学报，2003，31(12)：1933-1938)

[2] OASIS SOA Reference Model Technical Committee. Reference Model for Service Oriented Architecture 1.0. Beijing China, OASIS China, soa-rm-cs, 2006

[3] Ma Zhi-Yi, Chen Hong-Jie. Service oriented architecture reference model. Chinese Journal of Computers, 2006, 29(7): 1011-1019(in Chinese)
(麻志毅，陈泓婕. 一种面向服务的体系结构参考模型. 计算机学报，2006，29(7)：1011-1019)

[4] Zhu Lei, Zhou Ming-Hui, Liu Tian-Cheng, Mei Hong. A permission management model in service oriented architecture model. Chinese Journal of Computers, 2005, 28(4): 677-685(in Chinese)
(朱磊，周明辉，刘天成，梅宏. 一种面向服务的权限管理模型. 计算机学报，2005，28(4)：677-685)

[5] Liu Fang-Fang, Shi Yu-Liang, Zhang Liang, Shi Bo-Le. Substitution analysis of Web service composition via process algebra. Chinese Journal of Computers, 2007, 30(11): 2033-2039(in Chinese)
(刘方方，史玉良，张亮，施伯乐. 基于进程代数的 Web 服务合成与替换分析. 计算机学报，2007，30(11)：2033-2039)

[6] Martin Wirsing. Sensoria process calculi for service-oriented computing//LNCS 4661. Berlin, Heidelberg: Springer, 2007: 30-50

[7] Mei Hong, Shen Jun-Rong. Progress of research on software architecture. Journal of Software, 2006, 7(6): 1257-1275(in Chinese)
(梅宏，申峻嵘. 软件体系结构研究进展. 软件学报，2006，7(6)：1257-1275

[8] Medvidovic N, Taylor R N. A classification and comparison framework for software architecture description languages. IEEE Transactions on Software Engineering, 2000, 26(1): 156-168

[9] James Ivers, Paul Clements, David Garlan, Richard Nord, Bradley Schmerl, Jaime Oviedo. Documenting component and connector views with UML2.0. Software Engineering Institute, Pittsburgh, USA: Technical Report CMU-SEI-2004-TR-008, 2004

[10] Jie Ren, Taylor R N. A secure software architecture description language//Proceedings of the Workshop on Software Security Assurance Tools, Techniques, and Metrics. California, USA, 2005. Gaithersburg MD, U.S. National Institute of Standards and Technology, 2006: 82-90

[11] Bohm C, Jacopini G. Flow diagram, Turing machine and languages with two formation rules. Communications of the ACM, 1996, 9(5): 645-653

[12] Milner R. Communication and Concurrency. New Jersey, USA: Prentice Hall, 1989

[13] Zhao Hui-Qun, Sun Jing. Software architecture abstract model. Chinese Journal of Computers, 2002, 25(7): 730-736(in Chinese)
(赵会群，孙晶. 软件体系结构抽象模型. 计算机学报，2002，25(7)：730-736)

ZHAO Hui-Qun, born in 1960, Ph. D., professor. His research interests include dependable computer network and dependable software architecture.

SUN Jing, born in 1968, M. S., associate professor. Her research interests include software testing and decision support system.

Background

Service-oriented architecture (SOA) has been widely used as a framework for structuring software systems around the concept of service to provide integrated services to business processes. One of the important measures of the quality of the service software is dependability that ensures the safety and reliability of the services. Although a lot of efforts have been made towards dependable SOA, few methodologies have been reported for formal model design and describing the dependability attributes of the SOA. Although some progress has been made, research on SOA theory and theory-related technologies are still in the early stage. This paper proposes an algebraic model for abstracting and describing the attributes and behaviors of service, service composite, and SOA. The dependability of service component and the dependability of SOA software system are defined in this model. Based on this model some dependability attributes are analyzed and several dependability normal forms are defined, including a 3nd normal form (3NF). The authors conclude that a SOA design is dependable if it is in 3NF. Several case studies are discussed to illustrate how the above algebraic model can be used.

The trustworthy SOA algebra model has provided a functional base for studying dependability aspects of SOA software system. Based on SOA algebra model the authors are developing an evaluation model and some algorithm for analyzing the trustworthy and reachability of SOA service.

The work is partially supported by the National High Technology Research and Development Program (863 Program) of China (grant No. 2007AA010302), Beijing Natural Science Foundation (grand No. 4062012), and Beijing Government and Education Committee (grant No. KM200710009009).