第 29 卷 第 7 期
2006 年 7 月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 29 No. 7
July 2006

一个面向服务的支持动态演化的软件模型

李长云[1),2)] 李 莹[1)] 吴 健[1)] 吴朝晖[1)]

[1)](浙江大学计算机科学与技术学院 杭州 310027)

[2)](湖南工业大学计算机科学与技术系 株洲 412008)

摘 要 为了使面向服务的架构更适应动态演化的需求，提出了基于体系结构空间、支持动态演化的软件模型 SASM. SASM 是使用反射技术，通过具有因果相联的基层和元层来构造. 基层由可运行的服务形成，元层是一个具备树形层次、层间存在求精关系的体系结构空间. 该空间反映了需求的层次性，提供给用户不同抽象级别的视图和管理手段. 通过对体系结构空间的观察，可获知系统的结构和行为信息. 通过对体系结构空间的在线调整可实现对基层的修改进而实现系统的非预设动态演化，而空间中不同层次之间的求精关系保证了应用的完整性和演化的一致性.

关键词 动态演化；软件体系结构；求精；反射；服务

中图法分类号 TP311

A Service-Oriented Software Model Supporting Dynamic Evolution

LI Chang-Yun[1),2)] LI Ying[1)] WU Jian[1)] WU Zhao-Hui[1)]

[1)](*College of Computer Science, Zhejiang University, Hangzhou* 310027)

[2)](*Department of Computer Science and Technology, Hunan University of Technology, Zhuzhou* 412008)

Abstract In order to adapt to the opening environment of Internet and the variable requirements of users, software systems should be able to adjust themselves dynamically. Dynamic evolution is required to facilitate service-oriented application system. So this paper proposes an architectural space based, dynamic evolution oriented software application model, SASM (Software Architecture Space based Model), which is constructed by reflection technology. Its meta-level RSAS is an architecture space with tree-like layers, between which there exist refinement relations, and its base-level is composed of physical components. The meta-level and base-level are associated in the form of cause and effect. Within RSAS, different layers of architecture provide views and management means at different abstraction levels to the users, which meet the variable requirement of various roles from management layer to operation layer. Through the observation of RSAS the information of structure and behavior of application system is acquired. The on-line adjustment for RSAS can modify base-level so as to evolve application system.

Keywords dynamic evolution; software architecture; refinement; reflection; service

---

收稿日期：2006-02-17；**修改稿收到日期**：2006-05-07. **本课题得到国家“九七三”重点基础研究发展规划项目基金**(2003CB316906)**和浙江省重大基础软件产品开发专项基金**(2004C11053)**资助**. **李长云**，男，1972 年生，博士，副教授，研究方向为软件体系结构和语义 Web 服务. E-mail: lichangy@sina.com. **李 莹**，男，1973 年生，博士，副教授，研究方向为软件体系架构、中间件. **吴 健**，男，1975 年生，博士，讲师，研究方向为 Web 服务、网格计算、数据挖掘. **吴朝晖**，男，1966 年生，教授，博士生导师，CCF 高级会员，研究领域为分布式人工智能、网格计算、中间件.

1 引　言

随着软件技术的发展和软件系统规模的扩大，特别是 Web 服务为代表的技术的发展与成熟，人们在开发软件系统中关注的重点已经从各个功能的实现逐渐转移到如何将实现具体功能的各个服务组装起来形成完整的系统. 此外，软件的可演化性和软件的构造性密切相关，演化性高的软件其设计具有良好的构造性. 软件的构造性除了和软件的开发过程有关外，还取决于软件采取的应用程序模型. 在构建分布式的应用时，其首要问题就是确定应用的体系结构. 软件系统的体系结构原本是一个较为抽象的规约，最终分散而隐含地体现在系统各个服务及其间的交互行为之中. 关键的问题是：如何从总体上刻划并实现系统的结构，即所谓“体系结构”或“系统配置”的问题. 除此之外，分布式应用系统常常表现出行为动态性和结构灵活性的特点. 这些特点使得现有的框架形态和应用程序模型并不能完全适应分布式应用系统的需求. 需考虑的主要问题有：

(1) 采取什么样的应用程序模型来实现上面描述的体系结构. 这样的应用程序模型中，定义怎样的操作才能实现系统体系结构的动态演化，又如何约束这些动态演化.

(2) 作为一个处在运行时态的软件体系结构(RSA)，如何准确地描述目标系统的真实状态与行为，RSA 的改变又如何直接触发、驱动系统自身的动态调整，以支持分布应用系统的行为与结构演化.

(3) 需要良好的运行环境和平台支持. 这样的平台提供控制变更过程的手段，动态演化的全过程都可以得到监视和控制，保障演化完整进行，提供对变更前后状态进行切换的机制，以维持运行期间的上下文一致性.

本文提出一个面向服务的、基于软件体系结构、支持动态演化的应用程序模型. 该模型以 D-ADL 语言[1]为基础，旨在解决上述提出的问题. 本文将在第 2 节介绍相关工作；第 3 节描述该软件模型的原理、框架和机制；在第 4 节提出系统演化方法；最后进行全文总结.

2 相关工作

近年来，许多研究者关注在软件体系结构的指导下维护和演化系统，注意到了在系统运行时刻显式地维护体系结构信息的重要性. 其中的关键原因是：网络的开放性和动态性导致软件的演化性和复杂性进一步增强，越来越多的系统需要能够对自身进行动态调整；而面向服务的体系结构包括构成系统计算单元的服务、服务间交互行为和约束关系的连接件、服务和连接件形成的连通图结构，因此软件体系结构也可以揭示系统的变化性，它提供了一种在较高抽象层次观察系统并推理系统行为和性质的方式. 可见，运行时刻软件体系结构相关信息的改变可用来触发、驱动系统自身的动态调整.

为开发能随着环境和需求变化而动态自适应的系统，Dowling 等人设计了 K-Component 框架元模型[2]. 在 K-Component 元模型中，一个有类型的有向配置图被用来表示应用系统的软件体系结构，其中图节点表示构件接口；类型标签表示构件；有向边表示连接件. 一个反射机制被设计用来在这个有类型的有向配置图和目标系统间建立因果连接，使得配置图可以反映系统的真实软件体系结构.

根据关注分离(separation of concerns)这一软件工程基本原则，尽量使适配逻辑与计算逻辑解耦，K-Component 元模型提供一个专门的机制“adaptation contracts”来显式地表达适配逻辑，而不将其固化于程序语言或支撑平台之中. “adaptation contracts”使用一个叫做“adaptation contract description language”的描述语言书写，从而使得适配逻辑可编程和动态地修改. 配置管理器提供“adaptation contracts”的部署和运行环境，允许“adaptation contracts”动态地装载、修改和卸载. 一旦需要动态调整系统，适配事件产生，配置管理器依据“adaptation contracts”调用重配置操作，而重配置操作的结果将导致系统配置图的修改和迁移，进而更改系统本身. K-Component 元模型的缺陷是配置图虽然直观，但并不能严格和全面地表达软件体系结构的行为语义和结构语义.

ArchStudio 是基于软件体系结构的开发和运行环境，支持 C2 体系结构风格软件的动态修改[3]. ArchStudio 所定义的系统动态演化方法是如何使在体系结构层面表达的动态调整在具体系统中得以实施的一个典型代表，图 1 说明了这个方法.

目前，ArchStudio 包含三种体系结构变更源(sources of architectural modification)工具：Argo，ArchShell 和扩展向导(extension wizard). Argo 提供一个体系结构的图形描述和操作手段；ArchShell 提供一个文本的、命令式的体系结构变更语言；扩展

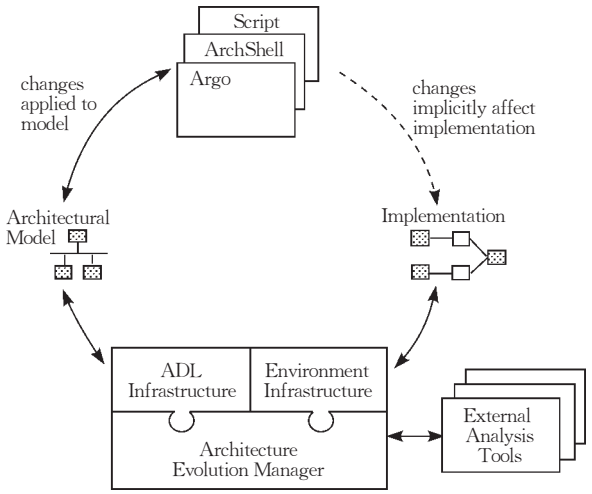

图 1 ArchStudio 的动态演化模型

向导提供一个可执行的脚本更改语言用来对体系结构进行连续演化. 但是 ArchStudio 采用的软件体系结构是设计阶段的制品,仅仅作为一种辅助文档,无法刻画系统真实的运行状态和行为,而且仅限于使用 Java 编写的遵循 C2 体系结构风格的软件动态修改,演化的合理性与正确性难以保证.

北京大学的 PKUAS 系统是基于体系结构反射的中间件系统,其目的是解决现有中间件系统过于注重系统局部或单个实体的反射,而缺乏全局视图的问题,以及过于注重中间件平台内部功能的反射而对上层应用反射不够的问题[4,5]. 对此 PKUAS 采用基于构件化的平台内部体系结构,并引入运行时软件体系结构(简称 RSA)作为全局视图以实现反射体系对系统整体的表示和控制.

PKUAS 系统中对反射的实现主要通过元模型、元协议和元数据三者进行. PKUAS 的元模型为各种容器和各种服务 Mbean,用于反射基层实体,管理部署信息和运行上下文导出元数据;元数据是由设计阶段的软件体系结构、部署阶段的部署信息以及运行阶段的上下文导出并累积而成的 RSA;元协议包括服务反射元协议和构件反射元协议,前者负责服务元数据的访问和修改,并执行装载、启动、停止、释放、替换和动态配置等动作. 后者主要通过构件容器来负责对构件元数据的访问和修改,并维护元数据与构件状态和行为的因果关联. PKUAS 通过扩展传统的体系结构描述语言 ABC/ADL 描述 RSA,并使之具备继承设计阶段体系结构所富含语义的天然能力. 但由于 ABC/ADL 缺乏构件行为和交互的形式化描述,没有解决构件替换时的可观察行为的一致性问题.

此外,还存在其它的一些基于体系结构的动态演化模型,如 Cazzola 等人提出的体系反射,着重考察软件体系结构与运行系统之间的关系[6~9];Garlan 等人利用度量计评估运行系统的状态和行为然后调整系统[10];南京大学提出了一种内置运行时体系结构对象动态协同架构[11]. 但所有这些工作都存在一些问题,主要表现为对 RSA 的行为语义描述不够严密,不能保证系统的完整演化和演化过程的一致性.

3 软件模型 SASM 的原理、框架和机制

3.1 基本原理

实际上软件体系结构不仅仅只是作为设计概念,任何可运行的软件系统都存在客观上的软件体系结构,这种客观存在的运行时软件体系结构是作为设计概念的软件体系结构的具体实现. 但是目前存在的一个主要问题就是在软件体系结构概念和具体实现之间存在语义鸿沟,主要表现为许多结构问题被分散实现在应用代码中,甚至于被嵌入到中间件平台和操作系统内部以及语言的运行时支持库中. 连接件嵌入在构件的代码片段内,而它依赖于中间件或操作系统提供的 API,那样的接口被实现在平台内部,细节对用户不可见. 因此,结构的问题被传播到操作系统和中间件内,而不再是平台独立.

究其原因,这是因为在实现和运行阶段,软件体系结构本身从整体上来说在系统中没有对应的一类显性实体,而以隐性的方式存在. 即使在设计阶段,软件体系结构的构造是良好的,但是在实现阶段和运行时也不能清晰可见. 因此,我们工作的基本出发点就是:使软件体系结构在可运行应用系统中以一类实体显性地表示,并且被整个运行环境共享,作为整个系统运行的依据. 也就是说,运行时刻体系结构相关信息的改变可用来触发、驱动系统自身的动态调整.

软件演化产生的动力主要来源于两个方面:(1)需求的变化;(2)设计的变化. 这两方面又是密切相关的. 一个较复杂的系统需要满足从管理层到操作层不同层次的用户角色的需求,而这些需求抽象层次和视图各不相同,具有不同的表现方式:目标、任务、功能和过程等. 考察一般的软件开发特点,软件开发过程应当是在不同抽象级别考虑和处理问题的过程,开发过程的每一步都是对较高一级抽象的解作一次较具体化的描述. 由于用户的需求是主观的和多层次的,因此不可能一开始就将它们进行完整描

述，最初总是从较抽象的、宏观的需求出发. 随着开发进程的不断发展，需求逐渐被细化、具体化. 在每个阶段，依据某种设计策略，这个阶段的需求驱动本阶段软件制品的产生. 当然，不同的设计策略，即使需求相同也可能产生不同的软件制品. 但总的说来，某个阶段的软件制品总是为了满足相应的需求而开发的. 因此，软件开发过程中各个阶段的软件制品也具有不同的抽象层次，也是不同抽象层次的用户需求的映射.

对于软件的静态演化过程，即通常所说的软件维护过程，一方面，为满足和适应管理层或操作层的不同层次角色的需求变化，总是从这个层次的需求所对应的软件制品入手，对其进行变更，并使这种变更逐次迁移繁殖，影响下阶段的软件制品，使之做出响应上阶段变更要求的设计变更. 另一方面，由设计策略的变化而导致的静态演化，最终也落实到某阶段的软件制品的变更，进行和需求变化引起的静态演化相同的过程. 由此可见，软件静态演化必须依赖丰富的语义信息，尤其是那些软件设计过程中产生的信息，否则，静态演化的合理性与正确性难以保证.

本文提出的支持动态演化、基于体系结构的软件模型（SASM）正是基于以上认识而提出来的. SASM 模型的出发点是在可运行系统中尽量多地保留设计时期的建模信息. 由于软件体系结构模型集中反映了开发者的设计决策，提供了一种观察系统并推理系统行为和性质的方式，SASM 使用软件体系结构模型作为系统设计期的建模信息载体. 又由于设计的层次性，我们在构造这种体系结构模型时也使之具有多层表示. 不同层次的体系结构反映不同用户角色的要求，提供给用户不同抽象级别的体系结构视图和管理手段. 这样的体系结构模型是一个体系结构空间，同时又是运行时可操作的，我们称之为运行时体系结构空间（Runtime Software Architecture Space，RSAS）. RSAS 作为一个在运行时刻有状态、有行为、可访问的对象，在系统实现中直接地、集中地反映软件体系结构规约，从而在最终实现中保留了体系结构决策，极大地提高了系统的构造性，为系统投入运行以后的演化行为提供了必要的依据和支持.

现在的关键问题是如何使 RSAS 准确地描述目标系统的真实状态与行为，如何使对 RSAS 的改变能够直接导致运行系统的相应变化. 反射技术是一个自然的选择. 在 SASM 中，RSAS 以元层形式存在，是关于系统本身的表示，这种表示可以被用户检查和调整；可运行的二进制物理构件（Web services）是基对象，一系列基对象形成基层，执行应用领域中的功能. 元层和基层因果相联，形成一个反射体系，从而可解决上述问题.

3.2　软件框架

基于以上思想的 SASM 模型的总体框架见图 2. 整个框架由因果相联的元层和基层组成，形成一个反射系统.

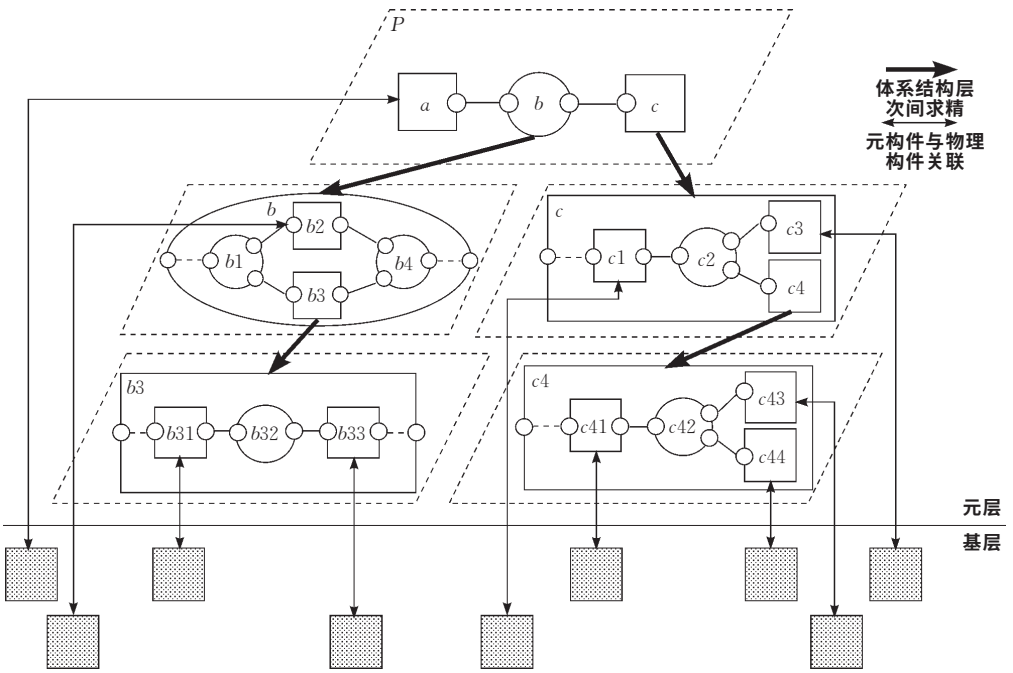

图 2　SASM 模型框架

元层由 RSAS 模型构成，描述系统软件体系结构语义，是应用领域分析设计的结果，使用 D-ADL 语言描述. 作为元层的 RSAS 模型是一个由不同层次的体系结构形成的信息空间. 其中，不同抽象层次的体系结构之间存在求精关系，高层次体系结构是低层次体系结构的抽象、约束，低层次体系结构是高层次体系结构的具体化和实现. 鉴于隔离求精方式的特点，RSAS 模型采取隔离求精方式构造. 隔离求精使得高层次体系结构中某单一构件(单一连接件)可能被求精为低层次中具有内部结构的某一复合构件(连接件)，但后者的外部观察行为(路由行为)和前者保持一致，后者的属性信息满足前者的属性约束.

依据隔离求精的特点，RSAS 模型的层次性表现为树或森林的形状，树或森林的所有叶子节点可连结在一起形成应用系统的最具体体系结构. 在 RSAS 模型中，构件叶子节点被称之为元构件，是软件功能的领域概念单元，描述了应用领域的计算语义和约束；连接件叶子节点被称之为元连接件，显性表示构件之间的交互行为和协作语义. 例如图 2 中 a，$b1$，$b2$，$b31$，$b33$，$c1$，$c3$，$c41$，$c43$，$c44$ 为元构件，$b1$，$b4$，$b32$，$c2$，$c42$ 为元连接件. 基层由可运行的二进制物理服务形成，真正地执行应用领域中的单元计算. 在 RSAS 中的每一个元构件都绑定一个物理服务，元构件是对应物理服务的信息映像，物理服务是其对应元构件的实现承载体.

在开放的环境下，交互模式应该根据应用和环境的特点而设，并可能需要动态调整. 对于普通的二进制物理服务，服务之间交互是通过对象引用进行，导致了服务互相依赖和通信协议被固化在服务内部的缺陷，使得整个应用难于联机演化，尤其是结构动态重配置. SASM 模型中的物理服务遵循服务工业标准实现，因此在一定条件下也可以直接部署并向外界提供服务. 但这些物理服务实现时其内部不包含其它服务的接口信息，不相互直接引用，是相对独立的. 这使得服务之间耦合度降低，对物理服务的替换可以不影响到其它服务.

所有要求其它物理服务提供的服务都依据元连接件规约进行. 交互事件发生时，运行环境从物理服务中接过控制权，分析传输过来的数据，解析执行元连接件规约所定义的服务间通信协议、信息交流和数据转换等功能. 同时运行环境依据 RSAS 描述语义，定位、调用其它物理服务. 图 3 说明了这个过程.

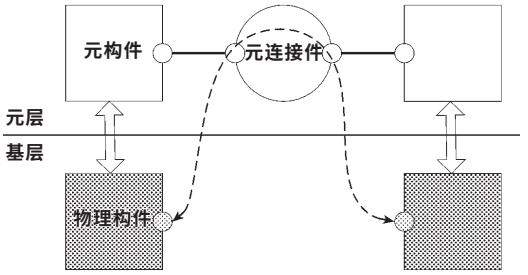

图 3 物理构件之间的交互

元连接件规约实际上起到了物理服务之间的粘结剂和交互线索的作用. 变更元连接件规约将导致物理服务间的交互和协调过程随之改变，交互可根据当前的体系结构配置情况动态地决定含义. 同时元连接件规约也可以附加信任控制、事务处理、QoS 管理等非功能属性.

3.3 SASM 模型中的反射机制

实现反射的关键是设计如何在运行期检查和调整系统的机制[12]. 通过检查机制可以观察到系统的当前状态；而调整机制可以在运行期改动系统的行为以更好地适应系统当前的执行环境.

要考察 SASM 模型中反射机制的具体情形，我们先从符合 SASM 模型框架的应用系统的构建过程谈起. 图 4 说明了这个构建过程.

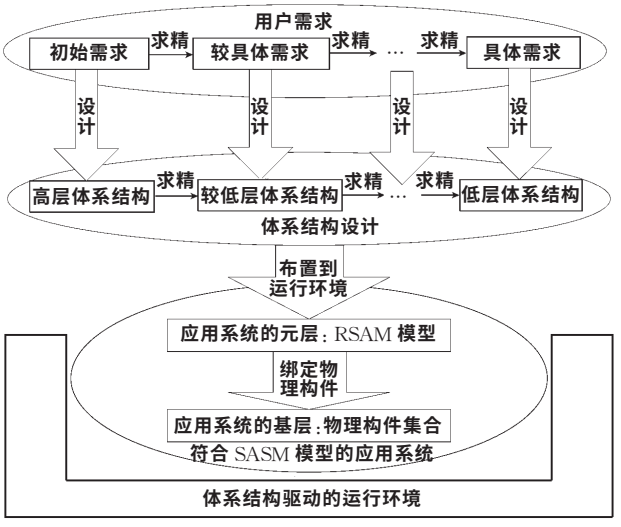

图 4 SASM 应用系统的构建过程

从初始宏观需求出发，随着对系统认识和分析的不断深化和细化，需求不断被求精而具体化. 需求驱动体系结构的设计，宏观需求转化为高层体系结构设计，微观需求转化为低层体系结构设计. 同时，低层体系结构也是高层体系结构的求精(采取隔离求精方式)结果. 所有层次的体系结构规约形成一个体系结构空间. 把这个体系结构空间部署到面向动态演化、体系结构驱动的运行环境中，就初步得到了

应用系统的元层：RSAS 模型. 对每一个 RSAS 中元构件依据其规约设计实现一个物理构件，或者从物理构件库中检索符合元构件规约的物理构件，所有这些物理构件形成应用系统的基层. 至此，符合 SASM 模型的应用系统得到建立.

可见，一方面，作为元层的 RSAS 模型描述了系统的体系结构规约，在系统中保留了体系结构设计决策. 通过对 RSAS 的观察，可获知系统的结构、行为知识. 另一方面，物理服务是其对应元构件的实现承载体，元构件是对应物理构件的信息映像. 因此，在元层和基层中建立起了初步的反射关系，这种反射既有结构的反射又有行为的反射，为系统投入运行以后的演化行为提供了必要的依据和支持.

下面进一步地考察系统的运行时情况. 当某一个物理服务正处于运行态时，运行环境跟踪、监控这个服务的执行，收集它的运行信息，如运行线程数、缓存大小、构件状态等，并把这些信息实时地存入其对应元构件规约中的动态属性值中，以反映系统的当前运行情况. 运行环境也对这些信息进行统计和分析，以判断服务的运行是否违反了元层 RSAS 规约中的行为约束，并向用户提出警报和建议. 同时由上节可知，服务间的通信通过对元连接件规约的解析执行完成，可理解为通信发生在元层，从而建立了一个元通信反射机制.

现在考虑元层 RSAS 发生变动的情形. 当元构件规约改变时，要重新进行服务的绑定过程，新的服务将引起系统单元计算过程的变更；当元连接件规约改变时，发生在服务间的交互和协调过程随之改变. 总之，对元层 RSAS 的调整，毫无疑问将导致系统的演化. 可见，在运行期间，通过元层 RSAS 模型实现了对基层的观察和修改，元层和基层之间因果相联. 一方面，通过检查元层 RSAS 模型，除获得系统的结构知识和行为规约外，还可获知系统的运行状态和信息. 另一方面，对元层 RSAS 的调整实现了对基层进而系统的修改.

4　系统演化方法

在 SASM 系统中，更常见的演化情形是由元层中 RSAS 的变更而引起的动态演化. RSAS 作为一个在运行时刻有状态、可访问的对象，在系统中直接地、集中地反映软件体系结构规约，从而促使系统开发人员从系统整体的层面理解和支持演化行为，在 RSAS 上所施行的更改操作通过 SASM 的反射机制最终影响到基层的变更，导致系统的演化.

RSAS 是一个由不同层次体系结构规约形成的体系结构空间，不同层次的体系结构规约反映了不同抽象级别的体系结构视图. 为提供给用户不同抽象级别的系统演化管理手段，RSAS 上的变更可以发生在每个层次上，每一层次都可进行构件、连接件的增删、替换. 例如图 5 可视化了图 2 中 RSAS 模型的树形层次结构的变更，图 6 可视化了对应体系结构空间的实际变更.

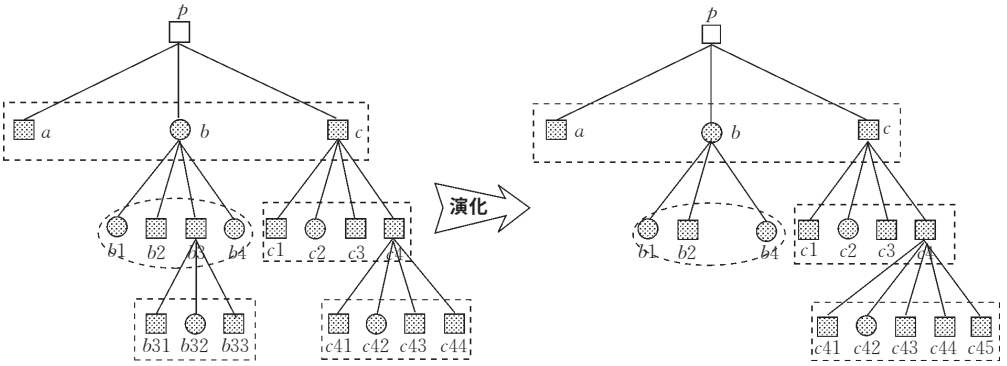

图 5　RSAS 模型的树形层次结构的变更

这种变更是以一种受控的方式进行：一方面，由于在构造过程中，我们知道 RSAS 模型的层次结构是一个树形结构，其中每个结点本身都是单一构件或单一连接件，由某个结点求精所得的所有下层结点可共同组成一个复合构件或复合连接件. 在 RSAS 上发生的变更正是以这些复合构件（复合连接件）为变更对象，对其包含的成员构件、成员连接件做增删、替换操作，以及在成员构件、成员连接件之间作关联或解关联操作，这些操作的结果还是复合构件（复合连接件）. 另一方面，由于层次之间存在着求精关系，高层次体系结构是低层次体系结构的抽象、约束，低层次体系结构是高层次体系结构的具体化和实现. 因此发生在某个层次上的变更不仅受到上一层次的约束，也将影响繁殖到下一层次. 上一

层次的约束意指变更发生前后的复合构件(复合连接件)都应该是其上层对应单一构件(单一连接件)的求精结果;影响繁殖到下一层次意指当复合构件(复合连接件)包含的某个元素删除时,由其衍生的下层节点都应删除,当复合构件(复合连接件)中增加或替换某个元素时,如果这个元素有进一步的求精,求精结果应该反映在这个元素衍生的下层节点上.

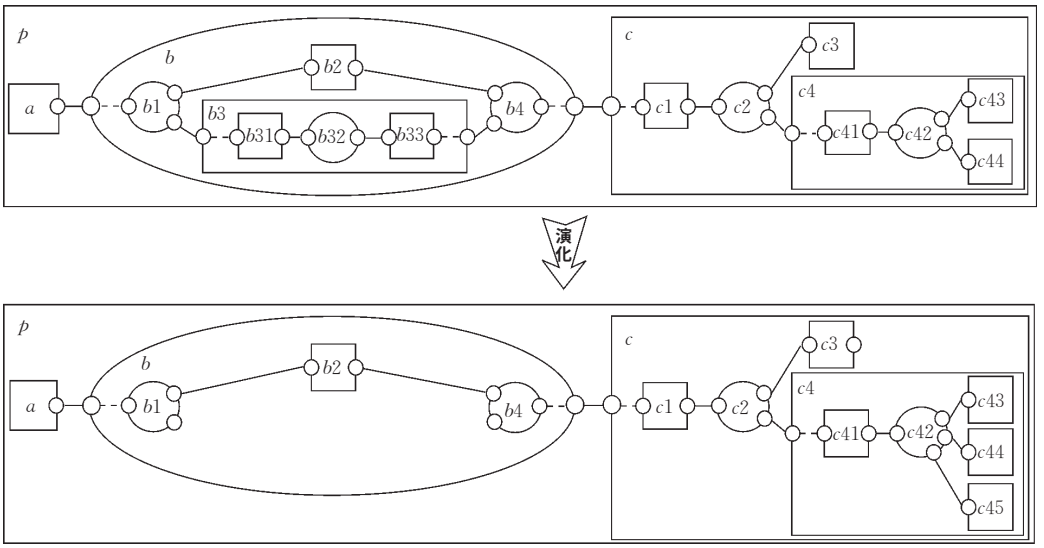

图 6 体系结构空间的实际变更

例如图 5 中 $b3$ 的删除将引起 $b31$,$b32$,$b33$ 的删除,同时删除 $b3$ 后由 $b1$,$b2$,$b4$ 组成的复合连接件还是 b 连接件的求精结果,即 $b1$,$b2$,$b4$ 组成的复合连接件行为分支弱模拟 b 连接件行为.同样的道理,增加 $c45$ 构件后,由 $c41$,$c42$,$c43$,$c44$ 和 $c45$ 组成的复合构件还是 $c4$ 构件的求精结果,即这个复合构件计算行为可以弱模拟 $c4$ 构件计算行为.

针对产生机制的不同,RSAS 上的变更可分为两种形式:预设变更和非预设变更.

预设的变更意指事先可知的变动,变动可实现为系统的固有功能,成为应用的一部分.在 SASM 模型应用中,通过在 RSAS 上定义预设的变更策略,以应对预设的动态演化.RSAS 规格使用 D-ADL 语言描述,而在 D-ADL 中,这种变更的表示是通过复合构件(复合连接件)的 choreographer 段处理的.Choreographer 段首先定义触发变更产生的事件(通常是某个输入事件),然后定义变更的处理策略(动态创建和删除构件、连接件、端口以及通道,变更体系结构元素之间的连接).需注意的是,choreographer 段规约本身是一个高阶 π 演算进程,而借助高阶 π 演算进程的迁移公式可归约得到进程执行结果.因此,choreographer 段规约对复合构件(复合连接件)进行变更或重配置,可事先通过计算得到这种变更或重配置的结果(新的复合构件或复合连接件).这种结果可用来评估是否符合其上层的约束,即是否还是上层单一构件(单一连接件)的求精结果,以决定这种演化计划是否有效.

在 SASM 模型中,虽然元构件行为规约是通过一个实现载体-基层服务执行完成的,但复合构件(复合连接件)的 choreographer 段规约是通过运行环境对其直接解析执行的,这和元连接件规约的执行方式相同.运行过程中,当变更事件发生时,运行环境首先执行变更处理策略,修改这个复合构件(复合连接件)的配置,然后把这种变更向 RSAS 的下层繁殖.

非预设的 RSAS 变更是通过对单一构件(单一连接件)的结构和行为直接进行不同的求精来完成的,为非预设的软件动态演化提供支持.对于 RSAS 树形层次结构中任意的一个节点(代表单一构件或单一连接件),不同的求精方案,可得到不同的求精结果(复合构件或复合连接件).因此,对某个非叶子节点采取和原来不同的求精方案得到一个新的求精结果,使这个新的求精结果取代原来的求精结果,从而可得到新的 RSAS 模型.表现在 RSAS 树形层次结构上就是从这个非叶子节点上衍生的代表原来的求精结果的子节点被代表新的求精结果的子节点所取代.求精方案的改变,既可源于用户某个需求的变动,也可源于设计策略的变动,而这些变动可能是事先无法预估的或不能完全预估的.

无论是预设的还是非预设的变更,在 RSAS 上的变更发生后,新增加的元构件需要绑定、载入符合其规约的物理构件,已删除的元构件需要从基层中

销毁其对应的物理构件.

对照 Cuesta 等人[13]三个级别的软件体系结构动态性的划分来看，本文的方法支持所有级别的动态性. 不仅允许结构动态性，即构件和连接件实例的创建、增加和删除，也支持体系动态性，即结构可以被重定义，如新的构件类型的定义. 这超越了一般的动态软件体系结构研究仅支持结构动态性的局限.

5　原型实现及相关测试

5.1　原型实现

模型原型实现在 BEA 的 Weblogic 8.0 中间件平台上，其中，原型系统由部署在 Weblogic 里的一系列协作的 EJB(元连接件引擎、交互事件截取器、运行信息跟踪器、物理构件管理器和动态演化管理器等)组成，元连接件引擎支持基于 J2EE 平台的 RMI 通信和基于 Web Services 的 SOAP 通信. 构造工具集的操作界面如图 7 所示.

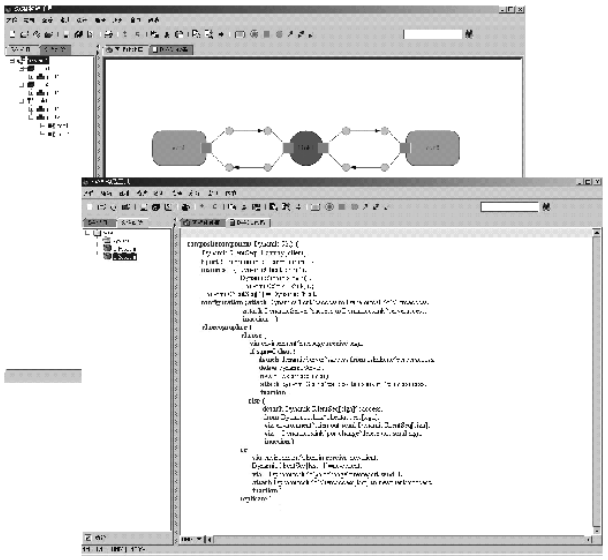

图 7　构造工具的可视化建模及编辑界面

5.2　对原型环境中服务构件间的通信测试

与直接通过 Weblogic 8.0 进行的服务构件引用交互方式相比，通过元连接件引擎进行的服务构件交互的额外开销主要耗费在元连接件执行管理器的派发操作和连接件执行器对元连接件规约语义的解析执行上. 对原型系统作了一些基本的性能测试，结果表明其带来的额外开销被控制在一个合理的可以接受的范围之内.

在实验中我们使用如下环境：6 台配置 800MHz Intel 奔腾处理器、256MB 内存的 Windows2000 PC 工作站，一台配置 2.1GHz Intel 奔腾处理器、1GB 内存的 Windows2000 Server 服务器，整个环境为 100Mbps 以太网. 工作站安装 J2EE 客户端用户软件和原型系统交互点运行时 Java 支持类，服务器安装 Weblogic 和原型系统的执行工具集 EJB 以及用户 EJB. 在实验中，J2EE 客户软件向服务器里的用户 EJB 发送消息，后者收到消息后立即将这个消息送回前者. 我们对 SASM 原型系统与直接使用 RMI 通过 Weblogic 在处理消息传递时引起的额外开销进行对比，并测量不同大小的消息往返所需的时间. 结果数据如图 8 所示. 从图 8 中可以看到，原型系统相对 RMI 的额外开销平均大至为 55ms 左右. 而且，尽管原型系统消息传递的总开销随着消息长度的增长而线性增长，原型系统本身导致的额外开销却增长十分缓慢，消息长度从 1 字节增长到 100000 字节，而这个额外开销仅从 40ms 增长到 75ms.

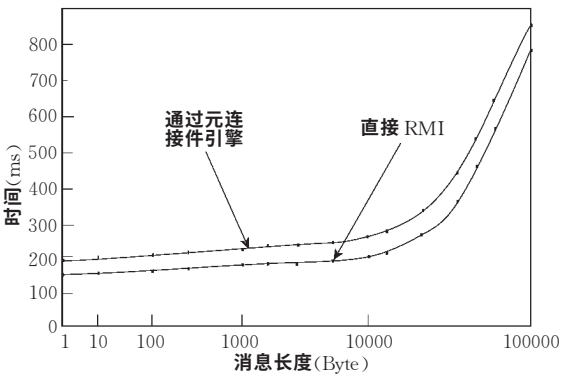

图 8　物理构件间的通信测试

基本性能测试表明，SASM 这种分布式应用模型在性能方面具有较好的可行性，与带来的优点比较，SASM 的额外开销在大多数的应用上是可以接受的.

6　总结与展望

演化性高的软件其设计具有良好的构造性，而构造性和软件采取的应用模型密切相关. 本文提出的软件模型 SASM 旨在提高软件的构造性，以更好地支持软件动态演化.

SASM 使用反射技术构造，其元层由一个具有树形层次的体系结构空间 RSAS 组成，以反映不同用户角色的需求，保留体系结构设计和决策信息，从而为系统投入运行以后的演化行为提供充分的依据和支持. 基层由可运行的物理构件形成，真正地执行应用领域中的单元计算. 物理构件彼此独立，不相互直接引用，其交互通过元连接件完成. 元层和基层因果相联，通过对元层 RSAS 的在线调整可实现对基层

的修改进而实现系统的动态演化. RSAS 提供给用户不同抽象级别的体系结构视图和管理手段,不同层次之间存在的求精关系保证了应用的完整性和演化的一致性. SASM 也支持所有级别的体系结构动态性.

近年来,把软件体系结构应用到动态演化领域的研究工作还只是一个开端. 文中提出的 SASM 模型及相关工作是这方面的一个较系统化的尝试和探讨,目前的原型系统只是 SASM 支撑平台设计的一个初步实现,进一步完善原型系统,为用户提供更好的系统开发、运行和演化环境,是一件非常有意义的事情.

参 考 文 献

1 Li Chang-Yun, Li Gan-Sheng, He Pin-Ji. A formal dynamic architecture description language. Journal of Software, 2006, 17(6): 1349～1359(in Chinese)
(李长云,李赣生,何频捷. 一种形式化的动态体系结构描述语言. 软件学报, 2006, 17(6): 1349～1359)

2 Dowling J., Cahill V., Clarke S.. Dynamic software evolution and the *k*-component model. In: Northrop L., Vlissides J. eds. Proceedings of the Workshop on Software Evolution, New York: ACM Press, 2001

3 Oreizy P., Medvidovic N., Taylor R. N.. Architecture-based runtime software evolution. In: Proceedings of the ICSE'20, Kyoto, Japan, 1998, 177～186

4 Huang Gang, Wang Qian-Xiang, Mei Hong, Yang Fu-Qing. Research on architecture-based reflective middleware. Journal of Software, 2003, 14(11): 1819～1826(in Chinese)
(黄 罡,王千祥,梅 宏,杨芙清. 基于软件体系结构的反射式中间件研究. 软件学报, 2003, 14(11): 1819～1826)

5 Huang Gang, Wang Qian-Xiang, Cao Dong-Gang, Mei Hong. PKUAS: A domain-oriented component operating platform. Acta Electronica Sinica, 2002, 30(12A): 39～43(in Chinese)
(黄 罡,王千祥,曹东刚,梅 宏. PKUAS:一种面向领域的构件运行支撑平台. 电子学报, 2002, 30(12A): 39～43)

6 Cazzola W., Savigni A., Sosio A. *et al*. Architectural reflection: Bridging the gap between a running system and its architectural specification. In: Proceedings of the 6th IEEE Reengineering Forum, Florence, Italy, 1998, 12-1～12-6

7 Cazzola W., Savigni A., Sosio A. *et al*. Architectural reflection: Concepts, design, and evaluation. University degli Studi di Milano, Milan: RI-DSI 234-99, 1999

8 Tisato F., Cazzola W., Savigni A., Sosio A.. Architectural reflection: Realizing software architectures via reflective activities. In: Proceedings of the 2nd International Workshop on Engineering Distributed Objects(EDO 2000), Davis, USA, 2000, 102～115

9 Ghoneim A., Cazzola W.. RAMSES: A Reflective Middleware for Software Evolution. RAM-SE, 2004, 21～26

10 Garlan D., Schmerl B., Chang J.. Using gauges for architecture-based monitoring and adaptation. In: Proceedings of the Conference on Complex and Dynamic Systems Architecture, Brisbane, Australia, December. 2001. http://www-2. cs. cmu. edu/-able/publications/cdsa01

11 Ma Xiao-Xing, Yu Ping, Tao Xian-Ping, Lu Jian. A service-oriented dynamic coordination architecture and its supporting system. Chinese Journal of Computers, 2005, 28(4): 467～477(in Chinese)
(马晓星,余 萍,陶先平,吕 建. 一种面向服务的动态协同架构及其支撑平台. 计算机学报, 2005, 28(4): 467～477)

12 Edmond D., Papazoglou M. P.. Reflection is the essence of cooperation. In: Papazoglou M. P., Schlageter G. eds.. Cooperative Information Systems. New York: Academic Press, 1998, 1～34

13 Cuesta C. Q., Fuente P., Solórzano M. B.. Dynamic coordination architecture through the use of reflection. In: Proceedings of the 2001 ACM Symposium on Applied Computing, Las Vegas, Nevada, United States, 2001, 34～140

LI Chang-Yun, born in 1972, Ph.D. candidate, associate professor. His main research interests include software architecture, and semantic Web services.

LI Ying, born in 1973, Ph. D., associate professor. His main research interests include software architecture, compiler technology and middleware technology.

WU Jian, born in 1975, Ph. D., lecturer. His main research interests include Web service, grid computing, data mining.

WU Zhao-Hui, born in 1966, Ph. D., professor, supervisor. His main research interests include artificial intelligence, grid computing, embedded system and so on.

Background

This work is supported by the National Basic Research Program of China (973 Program) under grant No. 2003CB316906: "The Research on Application of Semantic Grid on the Knowledge Sharing and Service of Traditional Chinese Medicine" and also supported by the Key Basic Software Special Program of Zhejiang Province under grant No. 2004C11053: "Project of Development and Industrialization of the Platform for Middleware".

In order to adapt to the opening environment of Internet and the variable requirements of users, software systems should be able to adjust themselves dynamically. Dynamic evolution is required to facilitate service-oriented application system. The paper contributes to the theory and applications of software model which supports dynamic evolution.