基于 Web 服务的 SOA 设计与传统软件设计方法的关系①

Comparing Traditional Soft Design Methods with SOA Design Methods Based on Web Services

何珍祥 （甘肃政法学院 计算机科学学院 甘肃 兰州 730070）
董逸生 （东南大学 计算机科学与技术工程学院 江苏 南京 210096）

摘 要: 本文通过对传统软件设计方法和基于 Web 服务的 SOA 设计方法的分析，阐述了基于 Web 服务的 SOA 设计方法是对结构化程序设计、面向对象的程序设计、组件化程序设计的继承和发展。同时指出目前软件设计的抽象层次、代码重用进一步提高，组成软件的粒度不断变粗，程序设计中的概念更接近实际业务；快速敏捷和灵活变化是软件设计的新要求，软件从业人员的分工细化是行业的新特点。

关键词: Web 服务 SOA 设计 软件设计 组件

SOA（Service – Oriented Architecture）构建的系统出现的比较早，多采用组件方式，而概念在 1996 才由 Gartner 提出[1]。Web 服务的基本标准 SOAP1.1（Simple Object Access Protocol）2000 年由 W3C 发布，UDDI（Universal Description，Discovery Integration）2000 年底由 OASIS 发布，WSDL（Web Service Description Language）2001 年由 W3C 发布[2]，其余的各种补充标准或规范还在不断完善中；基于 Web 服务的 SOA 是 SOA 的最新形式。Web 服务丰富了 SOA 的各种理论，并促成了 SOA 的成熟。基于 Web 服务的 SOA 已成为目前软件架构的主体。Gartner 预言，至少有 60% 的企业在 2008 年将采用 SOA 做为其 IT 架构[3]。

要采用 SOA 进行异构系统的集成、新系统的开发，深刻理解基于 Web 服务的 SOA 开发方法与原先的结构化程序设计（面向过程的程序设计）、面向对象的程序设计、组件化程序设计之间的关系，与软件工程中开发方法之间的关系是极为必要的。

1 基于 Web 服务的 SOA 方法是传统软件设计方法的继承和发展

1.1 软件开发方法、技术的发展过程

软件技术的发展是不断变化、继承、发展的过程，其变化如图 1 所示。

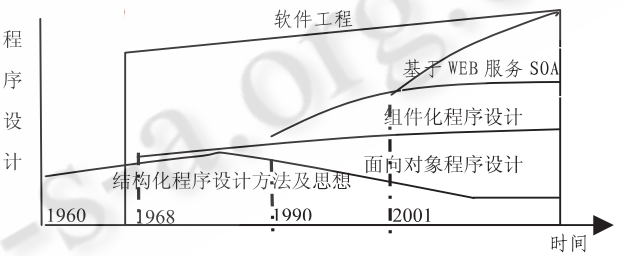

图 1 软件开发方法、技术的发展过程

1.1.1 结构化程序设计阶段

结构化程序设计出现在 60 年代初期，根据用户需求分析，将用户需求划分为若干个子需求，每个需求用一个具有独立功能的模块实现，这一模块被称为过程或子程序，包括函数。这些模块被主程序通过子程序、过程、函数调用构成一个软件系统，模块之间只能传递简单的参数信息。此阶段设计人员对系统的分析是针对功能的，用户需求的变化往往造成系统结构的较大变化。

① 基金项目：江苏省高技术项目“基于语义网格的系统集成方法和技术的研究与开发”（编号：BG2004034）

1.1.2 面向对象的程序设计阶段

面向对象的程序设计起源于 1960 年的 simula 语言,在 1990 年左右逐渐成熟。面向对象的方法将世界看成是相互联系着的各种对象的组合,对象将数据和操作封装在一起,对外界提供一些接口,其内部的实现细节和数据结构对于外部的对象是不可见的,对象之间通过消息相互通信,层次结构的对象通过相互间的调用构成一个软件。类、对象、实例、接口、属性、方法、事件构成了概念基础,封装、继承、多态、关联、实例化构成了操作基础。封装实现了不同实体间的隔离。继承让类的代码为子类重用、实例化让类的代码为对象重用,提高了代码的重用能力。不足的是类的粒度较细,无法描述业务流程,基类的代码变化对子类及其实例化的对象都将产生影响,造成了对象与类的紧耦合,使软件缺乏灵活性。

1.1.3 组件化程序设计阶段

组件化程序设计以 Microsoft 公司的 COM/DCOM/COM +、OMG(Object Management Group)1991 年提出的 CORBA(Common Object Request Broker Architecture)、Sun Microsystems 1996 年提出的 Java Beans 为标志。Microsoft 的 COM 及基于 COM 的对象连接和嵌入应用程序接口 OLE(Object Linking and Embedding)是组件思想的发源地[4]。

组件化程序设计将复杂的应用程序设计成一些小的、功能单一的模块,这些模块可以单独开发、编译、调试和测试,最终形成二进制软件单元,把它们经过组合就可以形成完整的软件系统,这样的模块被称为组件。当系统的软硬件环境发生变化或者用户的需求有所更改时,只需对受影响的组件进行修改,然后重新组合得到新的升级软件,实现软件的动态更新。对语言依赖性低,与具体技术的耦合性进一步降低。

CORBA 采用 OMG 所定义的公共对象模型,支持类、封装、继承和多态,对象或类之间按客户/服务器方式相互调用,传递消息采用信文。类不仅提供调用方法接口,还提供实现方法的代码,方法只能通过接口访问,客户方通过接口调用其它对象中的方法,服务方通过接口接收方法的调用。CORBA 采用 OMG 的 IDL 作为接口定义语言,用存根型调用或动态调用机制形成请求。ORB(Object Request Broker)是连接各方的核心,客户应用程序的请求交给 ORB,通过其为客户选择合适的服务器和方法代码,通过对象适配器 OA(Object Adapter)引导至相应方法的二进制形式的实现程序,在服务方执行完毕后,结果信文再由 ORB 传回。

Java Beans 是 1996 提出的基于 Java 的分布式对象模型,其构件叫 Beans,是以 Java 语言中的类和对象为基础定义的[2]。其类是生成对象的模板,而对象(即 Bean)是由类生成的一个实例。Java Beans 通过 Introspection 类的 Core Reflection API 到类中直接读取类及其方法的定义生成 Bean 和对 Bean 进行操作。Java Beans 通过类似 PRC 的远程方法调用接口 RMI(Remote Method Interface)通过网络远程访问 Beans 中的方法,通过 JVM 减少与硬件和操作系统的依赖,实现松耦合。

1.1.4 服务化程序设计阶段

服务化程序设计是在 1996 年 XML 语言诞生的基础上,以 2001 年后出现的 SOAP、UDDI、WSDL 为标志。服务化程序设计过程主要分为服务的确定、服务的描述、服务的实现、服务的发布、服务的编排、服务的请求。

服务是基于独立业务功能的模块,服务的定义分为服务的描述和服务的实现两部分。服务的实现可以用各种语言、在不同的操作系统和硬件上实现。服务的描述主要是完成服务的 WSDL 文档的生成,用来说明服务采用的数据类型、提供的操作、服务与 SOAP 的绑定、服务的地址和名称等。服务的发布就是将完成特定业务的程序打包成 Web 服务,在服务平台中注册。服务的编排将单个或合成服务整合成一个新的服务,完成业务流程。服务请求者和服务提供者之间通过 SOAP 消息发送请求和传递处理结果。服务请求者必须输入实际参数、根据服务的 WSDL 编写和解读 SOAP,才能调用相关服务。这些可以由用户编写程序来实现,也可以由 Web 服务平台自动生成,即根据 WSDL 生成服务代理(Service Proxy)与服务框架(Service Skeleton)。服务调用者根据服务代理生成的调用代码调用服务,服务开发者根据服务框架生成实现服务的代码框架。基于 Web 服务的 SOA 的程序架构如图 2 所示。

1.2 应用发展驱动 IT 技术的不断革新

1.2.1 软件工程方法和技术的发展

随着软件系统规模不断扩大,结构化程序设计无

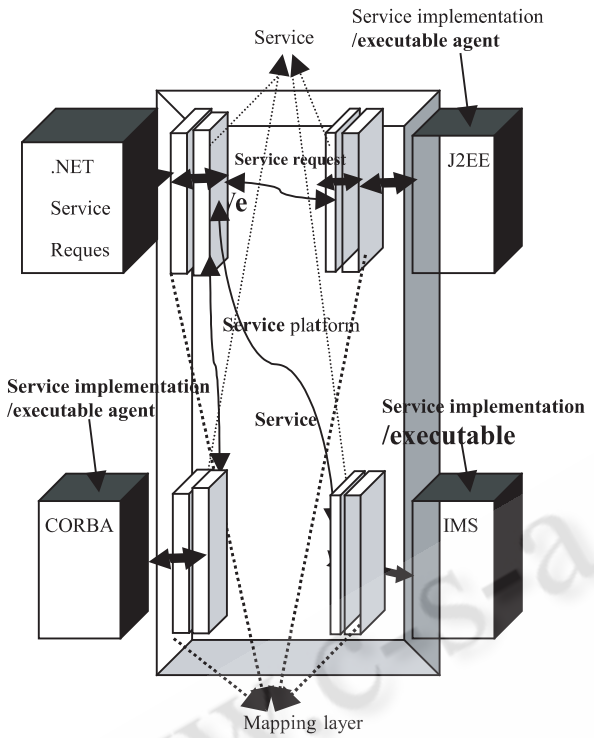

图 2　Web Service Architecture

法建立通用性较强的程序库,程序和数据与具体应用紧密的结合在一起,导致软件无法适应应用的变化,软件的升级更新、错误修改成本大,可维护性差,爆发了软件危机。1968 年产生了软件工程,即用工程的方法解决软件开发问题,软件周期、瀑布模型的理论成为软件开发的指导方针,工厂化生产软件成为软件开发追求的目标[5]。

减少系统中模块的耦合性,隔离关注等方法的深化,在实践中逐渐创造出了软件工程的新途径——面向对象方法学。面向对象技术以对象为中心构造软件系统,软件结构以问题域建立模型,不是功能的分解,功能的变化不会影响到整个系统,对象的封装、继承等增强了代码的重用。

1.2.2　组件化设计模式的出现

Microsoft 用动态连接库 DLL(Dynamic Linking library)提高程序的共享程度,但 DLL 中任何程序的微小修改都会影响其他所有的共享者,结果出现了许多 DLL 版本,以致开发者难以辨识和选用,即出现了版本地狱(Version Hell),严重地影响软件的可靠性[4]。基于此,Microsoft 提出 COM,做为共享程序和数据的二进制基本单位;COM 构件通过多个接口供用户调用,如果构件修改,则生成一个新的接口并改用一个新的全局标识符,从而避免了版本地狱。接口与接口的实现进行分离,降低了软件系统中构件的耦合性,同时在二进制代码级的重用率比基于对象的代码级重用率要高的多。

1.2.3　基于 Web 服务的软件设计模式的出现

Internet 的发展,出现了信息孤岛,现有的组件设计系统 COM/DCOM/COM + + 只能在 Windows 平台中运行;CORBA 的 IDL 没有严格的标准;Java Beans 是基于 Java 语言,这些组件设计系统之间也不能实现互操作。为解决这一问题,迫切需要一种技术完成各种情况下的互操作,而基于 HTTP 协议、XML 语言,采用 SOAP、UDDI、WSDL 标准的 Web 服务技术,可说是目前比较理想的一种解决方案。

2　SOA 是对传统软件设计方法的新发展

2.1　软件设计的基本概念更接近实际业务

基于 Web 服务的 SOA,以实际业务层面的基本概念为理论框架。流程、业务流程、服务是核心概念,它们根据企业实际的业务活动、业务数据、业务指标、业务规则来确定。服务的确定、描述、实现、发布、请求是基本操作,业务建模、流程建模是系统分析的基本模式。这些概念比结构化程序的过程、主程序、子程序、函数,面向对象的类、子类、对象、接口,面向组件的构件更能表达企业的实际业务,更容易实现企业业务人员与 IT 实现人员之间的沟通。

2.2　系统的耦合度进一步降低

对象、组件和服务都是对现实世界的抽象描述。对象是对客观世界中最基本实体的描述,数据与对数据操作代码封装在一起,构成软件的对象与采用的语言亦密切相关,多个对象及相互作用才能共同构成一个服务,任何业务的变化而引发的数据变化都将导致软件整体的变化。组件虽然实现了数据描述与数据处理代码的分离,但由于各组件实现系统采用各自的标准,不能实现互操作,其只能是细粒度的服务。服务用 WSDL 文件描述服务的数据、操作、访问等,用不同的语言、操作系统、硬件实现服务的代码编程,使描述与实现真正分离,因而使服务的粒度更粗,相互间的耦合性进一步降低。

2.3 代码的重用率进一步提高

对象是通过继承实现复用,组件则是通过合成实现复用,而服务是通过流程的编排来实现粗粒度复用[6]。通过面向对象的分析与设计,可以封装对象(或对象组)的某些方面,以简化复杂业务场景的分析。基于组件的分析与设计是从对象范例中自然发展而来的[7],在面向对象的分析与设计的早期,细粒度的对象提供"重用"的机制,但是这样的对象粒度级别太低。在应用程序开发和系统集成中,通过内聚一些细粒度的对象或组件来提供粗粒度的 Web 服务,进一步提高了代码的重用率。

2.4 软件设计的层次增加,操作进一步抽象,互操作性显著增强

基于 Web 服务的 SOA,将软件系统的实现分为多个层次,即底层的代码实现层,服务的描述层,流程编排和服务的请求层。代码实现仍然用面向对象的各种语言如 JAVA、C++、C#等实现,类、对象、方法、事件、接口是其基本编程单位,在事件、方法中也可用函数等技术。服务的描述层负责服务使用的数据类型、传递消息、操作、采用的协议、服务名称、地址的说明,与实现无关。流程编排完成由单个服务或组合服务组合成实际复杂业务过程的描述,在此基础上引入状态、服务质量的控制,在服务请求层完成服务或流程的使用。这样服务描述、流程编排、服务请求不再依赖具体的技术和提供者,从业务角度考虑软件设计,极大的提高了软件的互操作性。

3 基于 Web 服务的 SOA 对软件开发的影响

3.1 IT 行业从业人员分工进一步细化

基于 Web 服务的 SOA,需要将 IT 人员的角色重新定位,不同的角色侧重不同的任务,其技能要求也不尽相同。出现了业务分析师、集成开发人员、软件架构师、J2EE/Java/.NET 应用开发人员、企业原有系统开发人员等。

3.2 以资产的观点对待 IT 资源

各部门、单位、企业已用的传统 IT 系统为企业积累了大量的数据财富,同时是拓展新业务的信息基础。基于 Web 服务的 SOA 将传统系统通过包装成 Web 服务,重新开发新的业务,使原有投入得到最大限度的利用,资产得以保值和升值。

3.3 更加关注业务本身

业务驱动是基于 Web 服务 SOA 的根本特征。将企业应用程序构建为一系列服务,IT 资产通过服务的形式得到重用。业务模式也因此而在动态业务环境中重新组合,变得更加灵活,企业也可以更好地调整他们的投资。基于代码设计的服务实现不再是 IT 人员的中心任务。

3.4 更加关注服务质量和多渠道同质服务

服务渠道以及最终用户设备种类的增加,给企业机构通过员工、、AMT 机、计算机网络等渠道随时随地为客户提供同质服务,给 IT 部门增加了巨大压力。软件系统应更加关注各种设备的同时使用,并提供高质量的服务。

3.5 更加关注行业组织及相关标准

由于企业兼并、机构重组现象随时发生,软件系统的变化比较频繁,已有软件与后续开发的互操作性关系企业 IT 系统可持续发展。而互操作是建立在各种标准之上的,只有密切关注制定标准的各种组织,及各自制定的各种标准被接受的程度、相互竞争的标准,才能在自己设计系统时做出正确的选择。

3.6 实现渐进式的系统开发

基于 Web 服务的 SOA 可以采用战术和战略两种手段实现,所谓战术手段就是做几个 Web 服务就提供给用户使用,边开发边交付,使用户的投入以最快的速度获得回报。战略手段侧重于整体规划,一次完成,多用在无传统系统的机构且系统规模不太大的场景下。

3.7 以工厂化模式进行软件开发,使软件工程提高到新阶段

软件工程追求工厂模式生产软件,在 Web 服务阶段,由于服务的实现与描述的分离,使服务的实现可以类似插件一样完成服务描述的功能。服务的组合,可构成更大的插件。在软件设计中,要将服务的可插性做为一个重要性质去考虑,最终实现工厂化生产的目标。

4 结束语

基于 Web 服务的 SOA,是目前软件设计的主流趋势。对其设计方法做对比性分析,是实现此类 SOA 的基本前提,在继承与发展中找出脉络,是做好它的基础。更深入的分析,如代理程序的(下转第 14 页)

（上接第 59 页）

生成等未展开叙述，可参阅相关资料。

参考文献

1 王仰富. FSOA 与企业 IT 架构. Fhttp://www.ciotimes.com/news/2007－7－27/2007727152033.html.

2 Newcomer E, Lomow G 著，徐涵译 FUnderstanding SOA with Web Services 中文版 F 北京：电子工业出版社，2006.30－40.

3 毛新生. FSOA 原理方法实践. F 北京：电子工业出版社，2007.10－50.

4 王能斌，王洌，王泓. F Web 数据的管理和交换. 北京：科学技术出版社，2006.215－226.

5 徐敏，周定康. 组件技术在软件开发中的应用. 计算机与现代化. 2002，(2)：89－91.

6 曹莹. F 组件化程序设计方法. F 福建电脑，F2003(3)：44－45.

7 杨晓红，朱庆生. 组件化程序设计方法及组件标准. 重庆大学学报，2001，(11)：120－123.