服务组件架构的框架元模型及实现扩展点研究①

俞庆平，卢荣胜，夏瑞雪

(合肥工业大学 仪器科学与光电工程学院，合肥 230009)

摘 要：软件框架为特定领域内的软件重用带来了极大的便利,然而随着业务系统的发展，传统的基于组件的软件框架已经越来越难以满足业务系统发展的需要.面向服务架构作为新型的软件架构，具有极大的灵活性和扩展能力来支持业务系统的发展需要,在没有具体的指导规范下，SOA 开发是相当困难的。服务组件架构作为目前 SOA 最佳的编程模型，对其深入研究和理解,将有助于快速开发基于 SOA 的应用。从 SCA 的基本概念出发，分析了 SCA 的组成结构，在此基础上给出了 SCA 框架元模型，分析研究了实现扩展类型，并给出了具体应用示例。

关键词：面向服务；服务组件；SCA；SOA

Framework Meta-model for Service Component Architecture and Study of Implementation Extension

YU Qing-Ping, LU Rong-Sheng, XIA Rui-Xue

(School of Instrument Science and Optoelectronic Engineering, Hefei University of Technology, Hefei 230009, China)

Abstract：Software framework brings lots of conveniences for software reuse in specific domains. But with the development of business systems, traditional component-based software framework has become increasingly incompetent to meet the developmental requirements of business systems development. Service-oriented architecture (SOA) is a new bright software architecture with great flexibility and scalability to support the developmental requirements of business systems. But in the absence of specific guidances or specification, developing SOA application is very difficult task. Service Component Architecture (SCA) is the best one of all SOA development approachs. So study and research of SCA approach will be very useful for rapidly developing SOA-based applications. In this paper, the basic concepts of SCA will be introducted, and the SCA basic structure will be analyzed. Following these results, the SCA framework meta-model and a study result of implementation extension will be given. Then a specific example is given.

Key words：service oriented; service component; SCA; SOA

众所周知，软件框架（Software Framework）是特定领域软件体系架构(DSSA, Domain specific software architecture)[1,2]的实例，它具有这样的几个特点：1)框架是面向特定领域的，它构成了软件产品线的核心资产；2) 框架是 DSSA 的实例，具有部分实现的特性，它反映了产品线中应用的体系结构；3) 框架由一组协作的成分构成；4)利用框架开发应用系统是通过扩展点的实例化过程实现的[1]。因此，软件框架是特定领域部分实现的软件体系架构，是支持软件设计和实现复用的系统骨架，是可复用的软件制品，通过对它的扩展，可以生成特定领域内的应用。

目前框架开发实践方法主要有三种：面向对象的框架(OOF，Object Oriented Framework)，基于组件的框架(CBF，Component Based Framework)和服务组件的框架(SCF，Service Oriented Framework)[1-3]。OOF 是基于继承的白盒框架，其主要扩展方法是通过继承基类，重载函数等方式达到扩展的目的，因为 OOF 的种种缺陷[1,2]，纯粹的 OOF 已经很少见。随着设计模

① 基金项目:国家高技术发展计划(863)(2009AA04Z114);国家自然科学基金(50875074)
收稿时间:2011-10-23;收到修改稿时间:2011-11-17

式和组件技术的兴起。CBF 是主流的系统应用构建方法，通过对组件接口的扩展，可以有效地实现技术同构的应用系统。SCF 是面向服务体系架构(SOA, Serivce Oriented Architecture)的体系架构实例，因为 SOA 重心在于思想，而不是指导规范，不能作为具体实践的指导。服务组件架构(SCA，Service Component Architecture)是语言独立无关的 SOA 编程模型，它提供统一的调用方式，从而可将不同的组件类型通过统一的标准接口来封装和调用[4-6]。SCF 是针对特定领域的 SCA 框架，它负责通过组装已开发的组件服务，以快速应对领域的变化性为目的，这种组装过程往往是通过简单的配置来完成。由简单的分析可知，OOF，CBF 和 SCF 的主要区别在于实现技术和提供的扩展点的机制不同。

本文从框架的组成和出发，简要分析了研究 SCA 框架的必要性，结合中间件 Tuscany 和自己的项目实践，应用 UML 视图给出了 SCA 框架元模型，论述了组成框架的元素、框架的边界元素以及框架与元素、元素与元素之间的关系，并且结合具体实践，给出了一个 SCA 框架的具体实例。

1　问题的提出

近二十年来，基于组件的软件架构(CBA，Component Based Architecture)和 CBF 的研究和实践了许多，已留下大量宝贵的领域内的 CBF 和组件。然而这些智力成果往往是多种实现语言或技术完成的，如 JAVA，C++，BPEL，PHP，Spring，C#，VB.NET 等，组件访问方式也是多种多样，如 Web Service，JMS，EJB，JSON RPC，COM+，JMI，Tuxedo，COBRA，API 等，主机环境也是各有不同，如 Tomcat，Jetty，OSGI，Websphere，IIS 或应用程序等，而且很多组件都是跨多个进程或多个主机环境部署的。

面向如此繁杂的技术环境，让开发人员更加集中精力关注于业务逻辑的开发，而不落入技术陷阱中尤为重要。给这些不同的接口提供一个统一的调用方式是很有必要的，因此，业界提出了面向服务的体系架构（Service Oriented Architecture，SOA)的概念[3-5]，在该架构模型中下，每个业务功能都将定义为精心设计的服务，再通过这些服务的各种组合，最终形成各种应用系统。

因为，SOA 仅限于概念的讨论和研究，不方便指导 SOA 框架的构建，所以，业界又提出了方便构建 SOA 应用的服务组件架构(SCA，Service Component Architecture)编程模型。SCA 是实现 SOA 的模型规范，该规范描述了使用 SOA 构建应用程序和系统的架构模型，SCA 可以通过已有系统或服务来实现新的服务，简化了 SOA 应用的开发过程。SCF 的主要优势在于提供多种扩展机制来组装跨进程，跨机器边界的系统或组件服务，SCA 应用是多个应用间的协作[3-5]，CBF 更多集中应用程序内部组件的协作与扩展[1,2]，两者的侧重点各不相同，SCF 无法取代 CBF，SOA 也不是比 CBA 更加优越。所以，研究 SCA 框架的结构和扩展机制，无论是在使用已有 SCA 框架还是构建新的 SCA 框架都是很有价值的工作。

2　SCA框架元模型

由文献[6]~[9]可知，SCA 应用主要有几个特点：1)SCA 应用包含多个 SCA 组合(SCA Composite)；2)SCA 组合是构建 SCA 应用的基本管理单位，是个逻辑概念，在应用的配置文件中定义。3)组合间的通讯是提升组件的服务或引用完成的，其自身并不提供服务或引用。4)引用是用于绑定目标服务的，通过引用不同功能的服务,以达到扩展应用的目的。其中，SCA 组合主要特点有：1)每个组合内包含若干个组件；2)组件可分布在多个进程中，进程可运行在不同机器上；3)每个组件都是通过服务向外提供功能；4)组件间通讯使用统一的调问方式；5)组件具有高的可替换性，只要组件服务功能相同，组件的替换不会影响到应用；6)组合内组件服务的技术平台等可能各不相同，如图 1 所示。

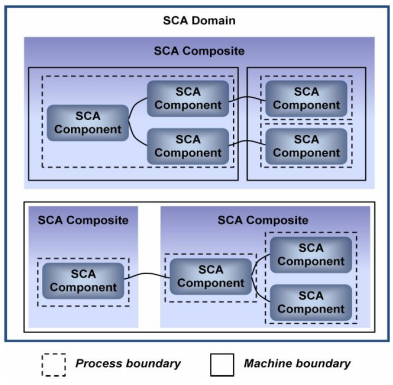

图 1　SCA 应用结构

SCA 作为服务组件体系结构，将所有的集成组件

都描述为具有定义明确的接口的服务组件。SCF 的主要任务就是有效地衔接服务组件，使其有效的协作构成领域应用，应对领域变化性，如图 2 所示。因为 SCA 主要是通过 WSDL (Web Service Description Language) 接口来访问服务组件的，服务组件在保持 WSDL 接口不变的情况下，可以通过不同实例或实现技术满足领域变化性的。

为了更好的理解服务组件，这里给出了 SCA 服务组件与传统组件的主要区别：1) 服务组件往往是粗粒度的，而传统组件以细粒度居多；2)服务组件的接口是标准的，主要是 WSDL 接口，而传统组件常以具体 API 形式出现；3)服务组件的实现与语言是无关的，而传统组件常绑定某种特定的语言；4)服务组件可以通过组件容器提供服务，而传统组件完全由程序代码直接控制。如视觉测量系统中，三维重建算法初期被设计成传统组件，在主机进程中执行，随着设备进入了试用阶段，需要在进程外部监控重建过程和结果数据，可以将传统组件改造为服务组件，在独立的服务进程或主机中执行，以便于应用系统和检测系统同时使用该服务，分享三维重建算法服务。

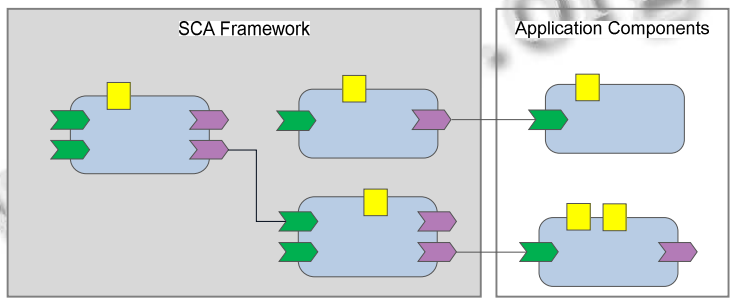

图 2 SCA 框架结构

通过分析，可以定义服务组件架构的软件框架为下面的五元组：

SCF={Service Components, Extension Points, Patterns, Constraints, Wires}

从图 2 可知，SC 及连接(Wire)构成了框架的主体结构。约束(Constraints)用于限定组件的属性，引用(Reference)和服务，还用于限定组件的类型，控制流程等。连接是服务和引用之间的关系，它确保组件以正确的方式访问目标的服务。模式(Patterns)是针对特定问题的通用解决方案，在组件的设计上常用 GoF 模式[10]，应用系统和框架的总体设计可采用架构模式和分析模式[11-16]。扩展点是框架可扩展部分，视应用具体情况进行扩展，在 SCA 中，主要有实现(implementation)，绑定(binding)，接口(interface)和数据绑定(databinding)等方面。综上所述，参照文献[1,2]的 CBSF 元模型，给出了 SCF 元模型，如图 3 所示。

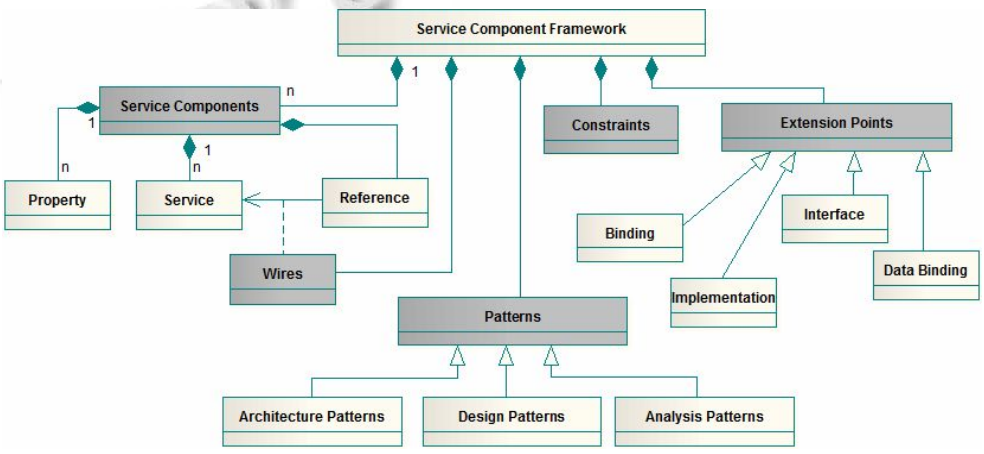

图 3 服务组件软件框架元模型

3 实现扩展点

框架中支持灵活扩展和定制的机制称为框架的扩展点，支持符合应用系统特定需求的实现。扩展点作为一类扩展机制，是框架支持大粒度复用，为系统扩展提供灵活性的关键所在[2]。根据扩展点在 SCA 框架中的作用和特点，可将其分为四类：实现(implementation)，绑定(binding)，数据绑定(data binding)和接口(interface)，限于文章篇幅，本文结合对中间件 Tuscany 的研究，给出实现扩展点的实现机制并举例说明。

3.1 定义

实现(implementation)：是指服务组件的具体实现，在不引起歧义的情况下，我们称为实现或组件实现，是组件服务的提供者，组件服务被调用时，调用的就是实现提供的功能。

实现类型(IMPT，implementation type)：是指实现服务组件的具体的组件技术实现类型，如中间件 Tuscany 已支持的实现类型有 Java，JavaScript，Spring，BPEL，SCA 规范中还支持 C++和 C 等。

实现扩展点：对于 SCA 框架来说，实现扩展点有两层含义：一、业务层面，因领域的变化性，开发具体的业务服务组件，达到扩展的目的。二、技术层面，是对更多的实现类型提供支持，可以理解为对实现类型的扩展。通常所指的实现扩展点，均指的是业务层面的实现扩展，但从 SCA 框架的构建角度，更多是侧重于技术层面，在复杂的应用领域，规范的实现类型在不能满足应用时，将会需要对新的实现类型的提供支持。

3.2 实现机制:

(1) 业务层面：

从 SCA 规范给定的 C 和 C++实现模型来看[17,18]，利用实现的进行业务扩展的基本步骤为：1)用具体的 IMPT 定义服务接口(Service Interface)，对接口添加相应的 SCA 标记；2)在同种 IMPT 下，实现该服务接口，对字段和方法添加相应的 SCA 标记；3)在配置文件中按 SCA 规范定义组件类型(COMT，component type)；4)根据需求部署服务组件到应用中，部署过程中需要指定 COMT 和 IMPT。

IMPT 和 COMT 的主要区别在于：IMPT 是框架部分，IMPT 使用 SCA 框架的同种技术实现；COMT 不同是业务扩展逻辑，可以使用不同的技术实现，该技术由 IMPT 规定，COMT 的加载，调用等均通过 IMPT 完成。

如下面的示例，先用 JAVA 定义了服务接口 LoginService，并用类型 LoginServiceImpl 实现了该接口，其后在配置中定义了登录服务接口类型，并将服务接口实现部署到了 sample.clientModule 的配置中。

```
<?xml version="1.0" encoding="utf-8"?>
<componentType xmlns="http://www.osoa.org/xmlns/sca/1.0">
  <service name="LoginService">
    <interface.java interface="services.security.LoginService"/>
  </service>
</componentType>
```

```
<?xml version="1.0" encoding="utf-8"?>
<module xmlns="http://www.osoa.org/xmlns/sca/1.0"
  xmlns:v="http://www.osoa.org/xmlns/sca/values/1.0" name="sample.clientModule" >
  <component name="LoginServiceComponent">
    <implementation.java class="services.security.LoginServiceImpl"/>
  </component>
</module>
```

(2) 技术层面：

SCA 规范中提出了对实现类型的扩展支持，经过对业务层面和中间件 Tuscany 的研究，结合项目实践，对 IMPT 的扩展的可概括为：框架提供 IMPT 扩展点的系列接口，构建 IMPT 扩展模块时实现这些接口。基本构建步骤为：1)定义扩展模块的配置文件；2)实现扩展点接口；3)实现扩展点注册和激活接口；4)实现配置文件解析器；5)部署模块配置文件到框架元数据区(可以是数据库或文件系统)；其中，扩展点接口不同的框架的构建模式不同，其接口也有所差异，但总体都可形式的定义为：

EXTENSIONPOINT:IMPLEMENTATION_TYPE

{

DEFINITIONS:

NAME：<自定义 IMPT 的名称>；

REFERENCE：<自定义 IMPT 模块的引用方式等>

CONFIG：<配置文件，约束类型，扩展类型，自定义属性等 >；

MODULES：

EXTENDED INTERFACES：<框架接口的自定义扩展接口，可选>

REALIZED CLASSES：<框架扩展点接口或自定义扩展接口的具体实现>

PARSER CLASS：<模块中定义，用于解析模块的定义>

LOADER CLASS：<模块中定义，创建/加载/注册实现类型等>

}

SCA 规范和中间件 Tuscany 都没有提供对 C#类型的支持，下面针对中间件 Tuscany 扩展 C#实现类型：

1) 在配置文件中定义 C#类型：

```
<element name="implementation.cs" type="SCA:CSharpImplementation" substitutionGroup="SCA:implementation" >
  <complexType name=" CSharpImplementation">
    <complexContent>
      <extension base="SCA:Implementation">
        <attribute name="directory" type="string" use="optional" />
      </extension>
    </complexContent>
  </complexType>
  ....
</element>
```

```
<component name="CSharpComponent">
  <cs:implementation.cs directory=".../csharp/" />
</component>
```

在对应的 schema 文件中定义：

2) 构建支持 C#的 JAVA 模块

a)实现 CSharpModuleActivator 类，该类实现了 ModuleActivator 接口，用于开始或停止模块提供的 IMPT。b)定义 ModuleActivator 的配置文件，它负责向框架提供 IMPT 的注册信息；c)实现 CSharp Artifact Processor 类，该类实现了 StAXArtifactProcessor 接口，它用于解析 IMPT 的配置文件；d)实现 CSharp Implementation 类，该类实现了 Implementation 接口，它负责向框架提供 C#类型的运行时实例。下面的 UML 图是描述了这些类型之间的关系。

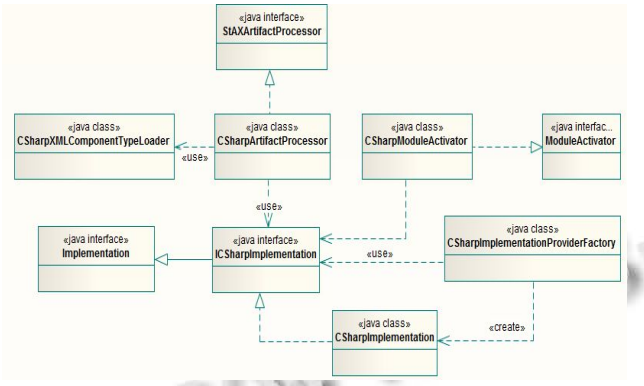

图 4 Tuscany 的 C# 实现类型扩展模型

4 实例研究

精密仪器设备框架的任务主要是使设备的各个部件协作工作，保证部件间数据共享，信号同步等，在实际应用环境中，设备的各个部件很多时候是空间位置上是分散的，同时，在企业应用中，还需要将设备的运行状况，数据等与企业的管理系统接口，如果使用基于组件的框架，无论是复杂的构建环境还是复杂的应用接口需求，框架的任务都将相当困难，而采用面向服务的软件框架，将使得新业务的扩充和演化更加容易，并实现业务工作流程的自动化定制和管理。

如图 5 所示，双目视觉测量系统的 SCA 软件框架，省略元素间的连接。所有服务组件都是 C#类型的，是实现扩展点的业务类型扩展，SCA 中间件为 Tuscany，因为不支持 C#类型，所以对 Tuscany 扩展了 C#的实现类型，服务组件采用 WCF[20]开发。

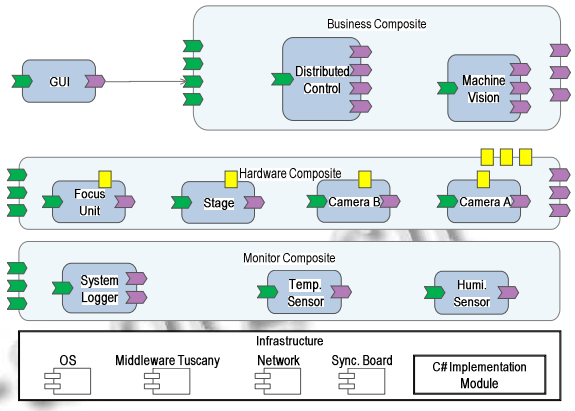

图 5 双目视觉测量系统软件框架

通过本实例：1)解决了 C# 开发接入 Tuscany 中间件的问题，使用 WSDL 作为调用接口，对实现扩展点进行必要的尝试和研究；2)服务组件间有效分离，极大方便了业务领域变化性需要；3)框架内各个组件都可以通过重新开发进行替换或扩展，增加了系统的灵活性；4) 对系统各个部件的诊断，日志，跟踪等更加灵活；5)机器视觉组件在硬件计算时，有效地节约硬件资源；

5 结语

本文结合基于组件框架的研究和若干特定领域框

架的开发为基础，提出了面向服务组件软件框架的元模型，结合实例描述了服务组件框架的扩展点，特别是给出了实现扩展点的两种形式和实现机制，对服务组件框架的设计和实现具有一定借鉴和指导意义。对于框架的形式化描述和其他扩展点的实现机制还有待于进一步加强和完善。

组成单元：

1）GUI 负责与操作人员交互

2）Distributed Control 负责协调各个服务之间的协同关系

3）Machine Vision 向外提供机器视觉及算法服务

4）Focus Unit 自动聚焦单元；

5）Stage 系统定位平台

6）Camera A,B 负责驱动双相机；

7）System Logger 负责向其他服务组件提供日志服务。

8）Sensor 系统环境传感器

9）C# Implementation 负责提供中间件 Tuscany 的扩展类型，以支持 WCF 开发的服务组件；

参考文献

1 Liu Y, Zhang SK, Wang LF, Yang FQ. Component-Based software frameworks and role extension form Journal of Software, 2003,14(8):1364−1370.

2 Hu WH, Zhao W, Zhang SK, Wang LF. Study of application framework meta-model based on component technology. Journal of Software, 2004,15(1):1−8.

3 Hewitt E. Java SOA Cookbook. Sebastopol:O'Reilly Media. 2009.

4 Josuttis NM. SOA in Practice. Sebastopol:O'Reilly Media. 2007.

5 Erl T. SOA: Principles of Service Design. Boston: PRENTICE HALL. 2008.

6 Chappell D. Introducing SCA. http://www.davidchappell.com/articles/Introducing_SCA.pdf. 2007

7 Laws S, Combellack M, Feng R, Mahbod H, Nash S. Tuscany SCA.in.Action. Stamford: Manning Publications. 2011.

8 BEA Inc, IBM Corp etc. SCA Assembly Model V1.00 . http://www.osoa.org/download/attachments/35/SCA_Assembly Model_V100.pdf. 2007.

9 BEA Inc, IBM Corp etc. SCA Policy Framework V1.00. http://www.osoa.org/download/attachments/35/SCA_Policy_Framework_V100 .pdf. 2007.

10 Gamma E, Helm R, Johnson R, Vlissides J, Design Patterns: Elements of Reusable Object-Oriented software. Addison Wesley,1995.

11 Schmidt D, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture, Volume 1. England: John Wiley & Sons. 2000.

12 Schmidt D, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture-Patterns for Concurrent and Networked Objects Volume 2. England: John Wiley & Sons, 2000.

13 Kircher M, Jain P. Pattern-Oriented Software Architecture-Patterns for Resource Management Volume 3. England: John Wiley & Sons. 2004.

14 Buschmann F, Henney K, Schmidt DC. Pattern-Oriented Software Architecture-A Pattern Language for Distributed Computing Volume 4. England: John Wiley & Sons, 2007.

15 Buschmann F, Henney K, Schmidt DC. Pattern-Oriented. Software. Architecture-On Patterns and Pattern Languages. England: John Wiley & Sons. 2007.

16 Fowler M. Analysis Patterns-Reusable Object Models. Addison-Wesley, Inc. 1997.

17 BEA Inc, IBM Corp etc. SCA C Client and Implementation V1.00.http://www.osoa.org/download/attachments/35/SCA_ClientAndImplementationModelforC_V100.pdf. 2007.

18 BEA Inc, IBM Corp etc. SCA CPP Client and Implemen- tation V1.00. http://www.osoa.org/download/attachments/35/. SCA_ClientAndImplementationModelforCPP_V100.pdf. 2007.

19 Li QX, Dong SW. Modern precision instrument design. Beijing:Tsinghua University Press. 2004.

20 Lowy J. Programming WCF Services, Third Edition. Sebastopol: O'Reilly Media. 2010.