基于 SOA 的电信 CRM 与 SFS 系统接口的设计与实现

Design and Implementation of the Telecom Interface between CRM and SFS System Based on SOA

李 波　刘卫国　（中南大学信息科学与工程学院 长沙 410083）

摘要：首先介绍了 SOA 的基本概念和关键性特征，然后针对电信 CRM 和 SFS 系统接口的具体业务要求，提出了一种基于 SOA 的设计与实现，并且给出了相应的算法和流程，讨论了消息 HUB 和消息通讯的算法实现。通过项目实践证明，这种基于 SOA 的电信 CRM 与 SFS 系统接口的设计策略缩短了开发应用的周期，使得代码的可移植性和重用性得到提高，并大大减少了系统维护的工作量，提高了系统的性能。

关键词：SOA　CRM　SFS　松散耦合　消息

1 引言

在软件开发中，软件应用程序的需求总是动态的，现在的解决方案必须能够灵活地适应未来的需求，而用户的需求是一个不断变化的灵活体。特别是在相关系统之间的数据交互中，由于系统设计人员对其他子系统的数据未知性，导致程序设计陷于无限制的设计演变[7]、代码开发和重新测试中，并导致系统的可读性和可维护性的大幅降低，最终发生不可预见的错误。预料到需求会不断的变化，所以需要软件架构能够被设计得使未来的需求改变对系统的冲击最小化。CRM 系统和 SFS 系统是电信运营支撑系统的核心，两个系统的接口更是至关重要。笔者在项目实践中通过研究，从基于 SOA 的设计出发，提出了解决这一问题的一种思路和实现方案。

2 SOA 概述

2.1 SOA 的定义

SOA 是英文 Service – Oriented Architecture，即面向服务架构的缩写。简单来说，SOA 是一种架构模型[6]，它可以根据需求通过网络对松散耦合的粗粒度应用组件进行分布式部署、组合和使用。服务层是 SOA 的基础，可以直接被应用调用，从而有效控制系统中与软件代理互联网络的人为依赖性。本质上说，SOA 体现的是一种新的系统架构，SOA 的出现，将为整个企业级软件架构设计带来巨大的影响。

2.2 SOA 的关键性特征

（1）独立运行（standalone）。服务（service）与组件（component）的根本不同，在于 service 是独立于调用者自行运转的，这也是 service 的优势所在。而 component 则需要其他 component 的协助才能运行。

（2）异步调用（asynchronous）。内在的异步特性是 SOA 包容真正的商业智能的关键所在。异步调用的好处在于，它使我们能同时处理多件事务。当然 SOA 并不排斥同步调用。

（3）基于消息（message based）。基于消息的调用方式是分布式系统的一种内在要求。消息是一种数据，它并不是远程对象指针。消息机制是异步调用和松散耦合的最好实现方案之一。

（4）纯文本协议 + 元数据（Plaintext Meta）。SOA 架构中基于纯文本协议是一个非常关键的技术抉择。说到纯文本的元语言，xml 无疑是这一概念最强势的候选者。

（5）松散耦合（loose couple）。这是 late binding（discovery），standalone 和 message based 等多种技术策略综合作用之后所达到的一种效果，这为外部灵活的流程配置做好了铺垫，也是众多 SOA 实施者所要追求的目标。

3 CRM 与 SFS 接口的设计与实现

3.1 业务需求

SFS(Services Fulfillment System)即服务开通系统，是指在电信运营商(现在从 CRM 系统中)受理客户的服务请求后，根据请求产生客户定单即申请单，然后系统自动生成或分解为服务定单和工单，通过网络资源的调配/调度或新建来最终完成客户服务请求的过程。

CRM 即客户关系管理系统。其完全以客户和产品为核心，具有高度的定制性，用户可以很大程度上自行定义系统的数据属性，因此传送给服务开通系统的数据格式具有高度的不稳定性。因此一旦数据格式发生哪怕是很小一点的变化，都意味着相关系统重新设计、代码开发、重新测试等相关工作，这为系统的维护带来了极大的难度。也降低了设计的可维护性和可读性。因此，有必要在体系架构上通过某种设计模型来解决这一接口的通用性问题。

3.2 设计的基本思想

由于采用基于 SOA 原则的连接方式，流程引擎跟远端业务系统之间通过 MQ 协议进行交互，需要在流程引擎管理界面中，配置相应的 WebSphere MQ Provider、连接工程和队列。当流程引擎需要调用远端业务系统时，需要通过 JMS 接口向引擎配置的队列发送消息报文，如图 1 所示。

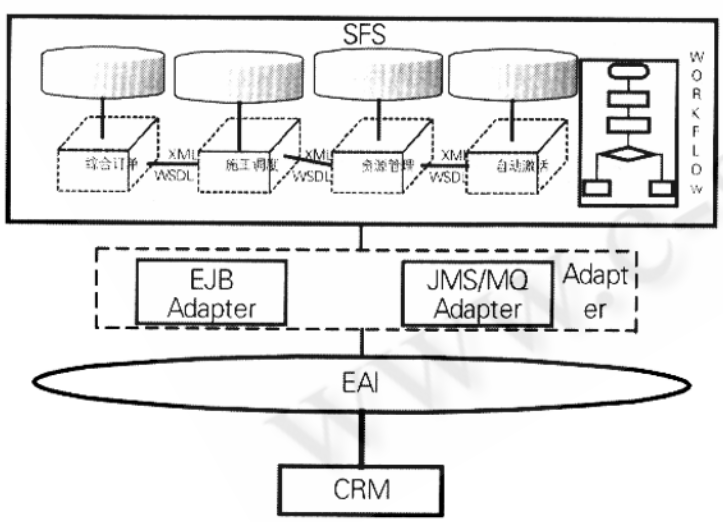

图 1　接口设计示意图

3.3 消息报文[4]的结构设计

根据 SOA 的原则，消息交互应该尽大可能的消除平台依赖性，使得交互更方便。由于 XML 具有简单、规范性，可扩展性，自描述性，互操作性等特点。因此报文全部内容封装在一个 XML 报文中，XML 报文分为两大部分：XML 头部分和 XML 体部分。XML 头 <HEAD>… </HEAD> 部分用于标识 XML 报文的基本属性，在现阶段包括版本号、源机源、消息目的地、应用名称等基本信息，用于标识当前数据报文规范版本、交易发起机构、交易目的机构、业务种类名称。XML 体 <MSG>... </MSG> 部分用于存放具体的交易报文。其内容由具体应用的交易种类决定。格式示例如下：

```
<? xml version = "1.0" encoding = " gb2312" ? >
  <CFX >
    <HEAD >
      <SRC > CRM </SRC >
      <DES > SFS </DES >
      <APP > 新装 </APP >
      <VER > 1.0 </VER >
    </HEAD >
    <MSG > {1:交易报文头块内容}{2:交易业务头块内容}{3:交易业务明细块内容}
    </MSG >
  </CFX >
```

3.4 CRM 和 SFS 系统交互的业务流程实现

CRM 前台根据客户选购商品进行业务受理，生成多条服务定单，把服务定单、服务定单间的关系、定单关联的客户等信息组织成消息包，并写入消息表，如图 2 所示。

SFS 系统通过后台服务程序从消息表中读取定单消息，将消息包解析并写入 SFS 数据库，然后激活工作流，由工作流来进行工单调度处理。如图 3 所示。

3.5 服务的封装

按照 SOA 的原则将 CRM 和 SFS 都各自封装成为一个服务。基本的实现步骤如下：

(1) 将 CRM 和 SFS 系统基于业务规划进行流程设计，然后对于流程访问的基本原子操作采用服务形式进行封装。

(2) 对于基本的原子业务操作利用 Session Bean 实现，特别是在系统内部基本都可以通过此方法实现；利用 WBISF 的自动 Wizard 工具将被流程调用的 Session Bean 通过 WSDL 进行封装。对于 SFS 子系统间的调用，如果业务允许异步处理，则通过 JMS/MQ 实

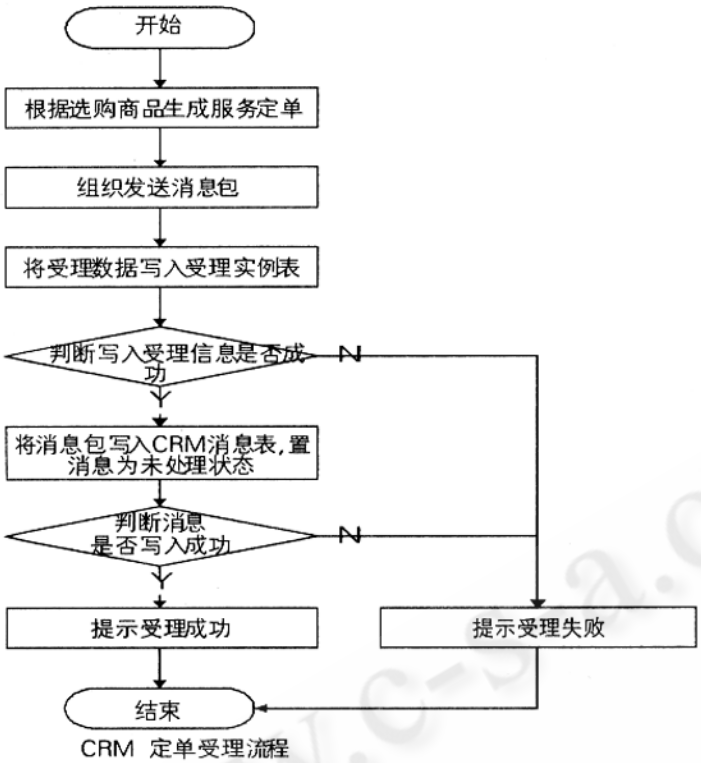

图 2　CRM 订单受理示意图

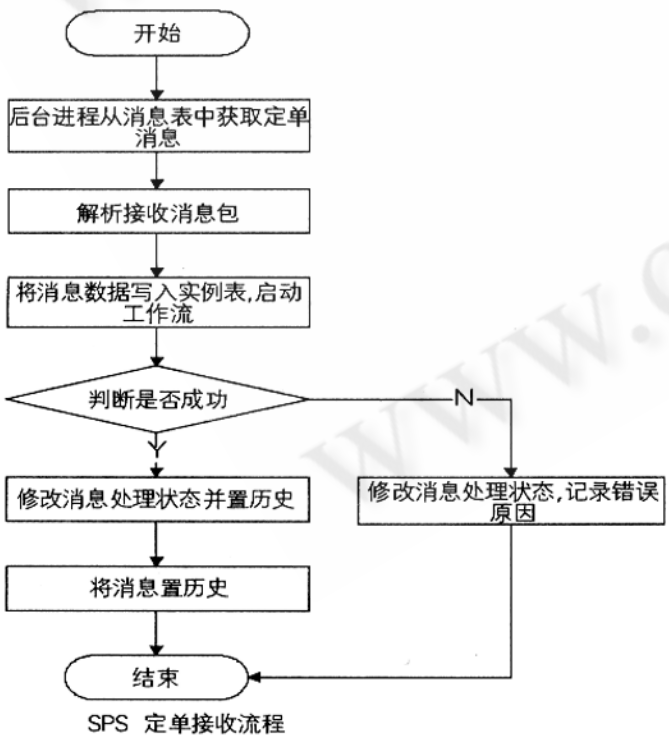

图 3　SFS 订单接收示意图

现，并适当配合 MDBean 使用。

（3）将要传递的信息包括定单号、老版本号、新版本号信息、要触发的参数等信息的 POJO 转换为 XML 包，消息包为一个 LIST 对象，包括多条定单信息。

（4）由数据提供者（CRM 或 SFS 系统）对实体数据进行维护，通过数据库物化视图保证变化的数据同步到数据使用者。数据提供者向数据接收者发送 MQ 通知消息，数据接收者收取 MQ 消息后到物化视图中获取数据进行本系统内部处理。

（5）CRM 和 SFS 各子系统由于采用 SOA 架构设计，彼此间无需定义过于清晰的界面。

举例来说“客户服务功能修改”这个功能可以由“客户信息装载”、“服务功能修改”、“工单数据生成”、“相关数据处理”四个“原子服务”完成，通过将这四个“服务”连接，就实现了“客户服务功能修改”的业务功能。也许业务规则会发生变化，比如，要求在“服务功能修改”之前先检查该客户的信用，应用的修改只要在流程中插入“客户信用检查”，形成“客户信息装载”、“客户信用检查”、“服务功能修改”、“工单数据生成”、“相关数据处理”的新流程，就满足业务部门的要求了。另一方面，其他的业务流程也可以调用“客户信息装载”等“原子服务”，组成其他的业务流程。系统的灵活性还表现在“服务”本身的修改上，只要“服务”的接口定义不变，“原子服务”本身的代码修改和功能实现的改变对业务服务流程都可以不产生影响。

在这里之所以将 SOA 中的服务称之为原子服务[6]，是因为在实际的业务支撑系统当中，一个原子服务是可以被多个流程所调用，例如一个检查客户信息状态的原子服务就会出现在多个业务流程当中。而几个原子服务打包能成为一个“服务”，例如上面所说的由四个原子服务组成的“客户服务功能修改”这个服务，如果对里面的四个子服务不作修改，那么这个服务就可以被其他流程直接调用。

CRM 和 SFS 系统的接口为消息队列，同时提供消息/数据传输的可靠性保障。业界领先的消息中间件同时提供同步、异步两种通讯方式。使用消息队列，消息系统可以管理很多通讯细节。此种接口方式为典型松耦合模式，符合 SOA 原则的设计，也是 EAI 技术普遍使用的方式之一，可以实现接口的重用能力。

3.6 消息 HUB

由于两个系统之间存在着大量的异步消息接口，通过点对点的应用接口完成应用的集成复杂度高，花费大，严重地限制灵活性。消息 HUB 连接需要交互的各个应用系统，减少了应用系统接口的数量、相互的技术依赖性。通过消息 HUB，实现消息的自动路由、消息的传送控制、消息日志的监控和管理。对于消息 HUB 就两部分进行说明：

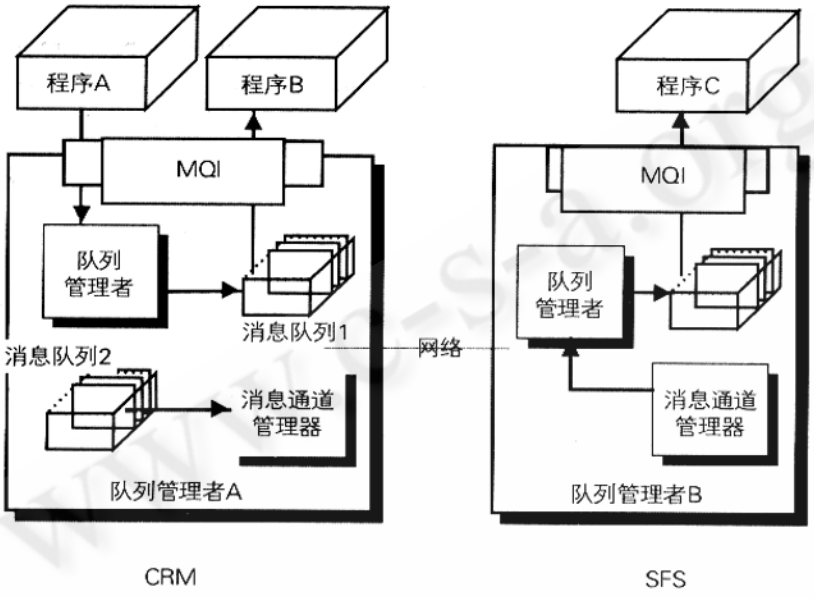

图 4　消息通讯示意图

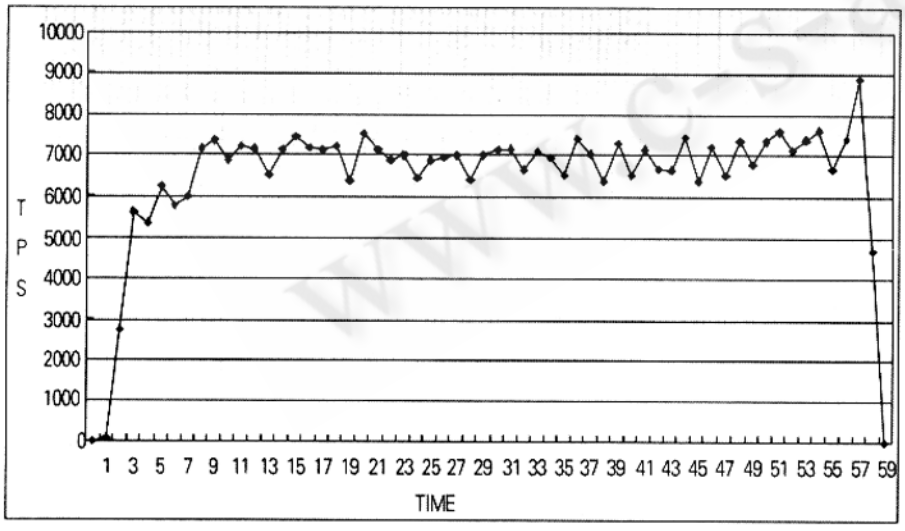

图 5　压力测试的结果示意图

(1) 发送时。后台多线程守护进程轮循消息表，将类型为需要写入 MQ 队列的消息写入 MQ 消息队列中，成功写入 MQ 消息队列后将消息表对应记录修改状态并置历史。如果配置了消息转发对象数据，则根据配置的消息队列数启动多个线程(每个线程处理一个 MQ 队列，另外还有一个处理转发对象为空的线程)。调用 MQ 提供的 API 查看 MQ 中堆积的消息情况，如果 MQ 中堆积消息数超过指定存放消息数或者最大消息数则不予处理该消息。

(2) 接收时。后台多线程守护进程轮循接收消息的 MQ 队列，根据接收到的消息包指定的接口类型而动态调用相关的业务类方法，实现本地业务逻辑。将消息转发对象配置表数据读入内存，以便根据队列数启动线程；如果配置了消息对象数据，则根据配置的消息队列数启动多个线程(每个线程处理一个 MQ 队列)。根据消息的服务类型判断是否需要消息转发，如果配置了多个转发对象则随机选取一个，如果未配置则转发对象为空。

3.7 消息通讯实现

首先来看本地通讯的情况。如图 4 所示，应用程序 A 和应用程序 B 运行于同一系统 CRM，它们之间可以借助消息队列技术[10]进行彼此的通讯：应用程序 A 向队列 1 发送一条信息，而当应用程序 B 需要时就可以得到该信息。

其次是远程通讯的情况(即 CRM 和 SFS 通讯的情况)，如果信息传输的目标改为在系统 SFS 上的应用程序 C，这种变化不会对应用程序 A 产生影响，应用程序 A 向队列 2 发送一条信息，CRM 系统的 MQ 发现 Q2 所指向的目的队列实际上位于系统 SFS，它将信息放到本地的一个特殊队列 – 传输队列(Transmission Queue)。我们建立一条从 CRM 到 SFS 的消息通道，消息通道代理将从传输队列中读取消息，并传递这条信息到 SFS，然后等待确认。只有 MQ 接到 SFS 成功收到信息的确认之后，它才从传输队列中真正将该信息删除。如果通讯线路

不通,或 SFS 不在运行,信息会留在传输队列中,直到被成功地传送到目的地。这样就可以确保信息传输,并且是一次且仅一次(once – and – only – once)的传递。

4 测试与评价

4.1 测试

在本系统上线前对其进行压力测试,模拟单笔实时定单受理交易流程,由 500 个基于 MQ Client 的进程连续发送交易请求,共发送了 4,000,000 笔交易。

我们通过数据库的记录数来度量系统的吞吐率。利用 speed 程序每隔 10 秒重新统计 t_revenue_return 表的记录数,前后两次统计间的记录数之差除以其间隔时间,表示该时间内系统的平均吞吐率。利用 speed 统计得到的系统吞吐率如图 5 所示。

其中,横轴代表时间,单位为 10 秒;纵横轴代表 TPS (Transactions Per Second),单位为笔/秒。需要说明的是,由于测试所用的服务器资源的限制,我们在 Oracle 所在的服务器上同时运行着 250 个模拟定单受理的 MQ Client 进程和用作 MQ Gateway 的两个 MQ Server。所以在系统加压的初期,由于 Oracle 服务器上资源的竞争,导致系统的吞吐率较低。基于上述数据,在系统处于稳定运行的时间内,其平均吞吐率为 6,993 笔/秒。在一个有限规模的硬件配置环境中,在每笔交易的包长 2K、包含 16 个 SQL 操作的情况下,系统的吞吐率高达 6,993 笔/秒。

4.2 评价

测试结果表明,该方案基于 SOA 原则,符合业界整合的潮流。在具有先进性、灵活性、易于开发和维护的同时,也具有优异的性能。该结果远远高于系统的预期目标 (2,100 笔/秒)。同时,该架构具有近似线性的扩展能力,完全可以满足近期和未来的处理能力要求。当然采用这种方式设计接口也有需要注意的地方,比如说消息的轮询策略,必须采用一定的算法来加以控制。还有消息丢失的补偿策略等。这些都有一定的方法来解决,但不是本文讨论的重点。

5 结束语

本文首先从当前软件需求和架构模式出发引出 SOA 的基本概念。然后介绍将 SOA 模式应用到电信 CRM 和 SFS 系统的接口设计当中去。并详细解释了 CRM 和 SFS 如何来通过消息进行交互以及相关的实现方案。最后给出了压力测试的结果和评价。通过项目实践证明,在竞争日趋激烈的市场环境下,该项目利用 SOA 解决方案大大提高了业务流程的处理速度,降低了开发和维护成本,数据传输更加安全、迅速,提高了整个业务系统的工作效率。

参考文献

1 W3C Working Draft. Web Service Architecture Requirements. [S]. http://www.w3.org/TR/2002/WD – wsa – reqs – 20020819,2002.8.

2 Sundaram M, Shim S S Y. Infrastructure for B2B exchanges with RosettaNet[C]. In: Proceeding of 3rd International Workshop on Advanced Issues of E – Commerce and Web – Based Information Systems, San Juau, California: IEEE,2001:110 ~ 119.

3 Tenenbaum J M, Chowdhry T S, Hughes K. Eco System: An Internet Commerce Architecture [J]. IEEE Computer,1997;30(5):48 ~ 55

4 IBM. Patterns: Service – Oriented Architecture and Web Services. (sg246303)[Z]. 2004.

5 Bill St A rnaud, Andrew B jerring, Omar CherKao – ui, et al Web Services Architecture for User Control and Management of Optical Internet Networks[J]. Pr – oceedings of the IEEE,2004,92(9):1490 – 1500.

6 Eric Newcomer , Greg Lomow. 徐涵译, Understand – ing SOA with Web Services 中文版[M], 北京:电子工业出版社,2006.

7 G Erich, H Richard, J Ralph, 等, 李英军、马晓星、蔡敏等译, 设计模式 – 可复用面向对象软件的基础[M], 北京:机械工业出版社,2000.

8 王莉娟、宁汝新、张旭, 基于 Web 服务的虚拟电子仓库的设计与实现[J], 计算机工程与应用,2003,39.

9 潘顺, EAI 模型与架构设计研究, 电信科学,2004,20,(6):15 ~ 18.

10 IBMWebsphere MQ 白皮书. http://www – 900.ibm.com/cn, /software/websphere/products /download/datasheets/Web – Sphere – MQ – V6.0.pdf.