


基于企业架构（EA）的企业信息化建设模型

肖建国
2010 年



中国 [选择] | 使用条款

dW 全部内容

搜索

首页 | 产品 | 服务与解决方案 | 支持与下载 | 个性化服务

developerWorks 中国 > SOA and Web services >

developerWorks.

developerWorks 中国

本文内容包括：

- SOA 和 DW 概念
- SOA 和 DW 结合的企业架构
- 基于 IBM 产品体系的实现
- 总结
- 参考资料
- 参考文献

面向服务体系架构 (SOA) 和数据仓库 (DW) 的思考

基于 IBM 产品体系搭建基于 SOA 和 DW 的企业基础架构平台

级别：初级

肖建国, IT 咨询顾问, 浪潮软件

2009 年 10 月 19 日

文档选项

- 打印本页
- 将此页作为电子邮件发送

面向服务体系架构 (SOA) 和数据仓库 (DW) 的思考

最初由 IBM developerWorks 中国网站发表, 网址 <http://www.ibm.com/developerworks/cn>
本文 http://www.ibm.com/developerworks/cn/webservices/0910_soa_datawarehouse/index.html
博客: <http://xiaojg.spaces.live.com>

肖建国

2009-10-16

摘要

当前业界对面向服务体系架构 (SOA) 和数据仓库 (Data Warehouse, DW) 都介绍的很多, 提出了很多优秀的解决方案, 但是一般是把 SOA 和 DW 单独考虑, SOA 和 DW 有着共同的目标—系统整合, 由于基于不同的技术思路, 提出了不同的方案。本文将围绕 SOA 和 DW 相结合的思路, 基于 IBM 的产品, 规划统一的数据库, 搭建企业级的技术架构。

SOA 和 DW 概念

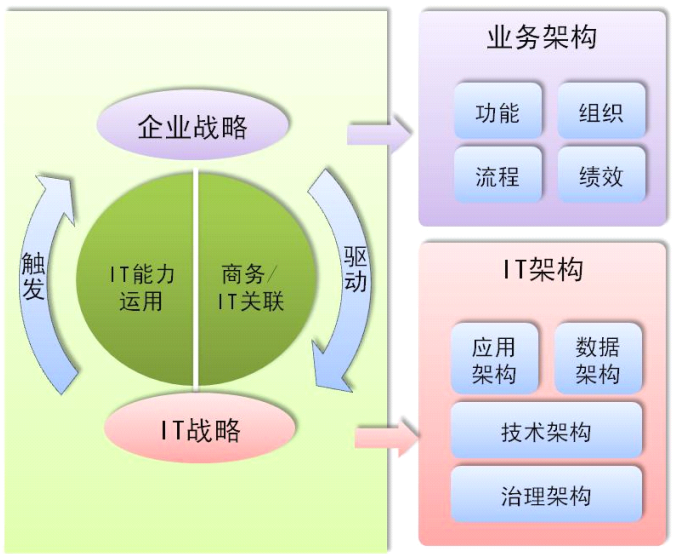
随着 IT 技术的发展, SOA 和企业架构 (Enterprise Architecture, EA) 逐步融合, 形成了新的架构理论, 但是与 DW 之间还没有很好的集成。下面首先来看看 EA、SOA 及 DW 概念。

企业架构的概念

企业架构 (**Enterprise Architecture, EA**) 的概念产生于 1987 年, 在 IBM 的一个内部刊物上发表的一篇文章 "A Framework for Information Systems Architecture" by J.A. Zachman (扎克曼) 中提出。概念的提出是为了应对日益复杂的 IT 系统, 以及高投资、低回报的问题。他认为使用一个逻辑的企业构造蓝图 (即一个架构) 来定义和控制企业系统及其组件的集成是非常有用的。为此, Zachman 开发了信息、流程、网络、人员、时间、基本原理等 6 个视角来分析企业, 也提供了与这些视角相对应的 6 个模型, 包括语义、概念、逻辑、物理、组件和功能等模型。随着 EA 的发展, 产生了很多的流派, 当前主要的 EA 架构包含: 通用框架 Zachman、TOGAF(The Open Group Architecture Framework)、以及适用于政

政府和军方的美国联邦政府的标准架构 FEA、美国国防部的 DoDAF 等。这些模型主要分成两派，如今正在逐步的融合在一起。随着企业架构的不断进化，企业架构理论越来越与战略和业务相融合，逐步形成了企业战略、业务架构、IT 战略、IT 架构等四个层次的 IT 规划方法论。IT 架构包含数据架构、应用架构、技术架构和 IT 治理等四个方面的内容，其中技术架构包含集成平台、公共服务平台、基础平台（软件和硬件）和安全平台等，如下图所示：

图 1. 企业架构（EA）示例



本文以下主要从技术架构中的集成平台角度来看如何搭建应用集成平台和数据集成平台。

面向服务的体系架构

面向服务的体系架构（Service Oriented Architecture, SOA）是一种架构 IT 系统的方法，它将应用和 IT 功能划分为单独的业务功能和模块，即所谓的服务。用户可以构建、部署和整合这些服务，且无需依赖应用程序及其技术平台，从而提高应用的灵活性。这种业务灵活性可使企业和机构加快发展速度，降低总体拥有成本，及时、准确地获取信息，同时有助于实现更多的资产重用。而建设 SOA 体系架构就需要建立一个一致的架构框架，在这种框架中，可以快速地进行开发、集成和重用应用系统。而对于原有的应用系统来说，可以采用合适的技术手段进行平滑的优化与过渡。

数据仓库

数据仓库（Data Warehouse, DW）是一个面向主题、集成、时变、非易失的数据集合，是支持管理部门的决策过程（W. H. Inmon）。数据仓库具备以下四个关键特征：面向主题(Subject Oriented)的数据集合；集成(Integrated)的数据集合；时变(Time Variant)的数据集合；非易失(Nonvolatile)的数据集合。根据数据仓库所管理的数据类型和它们所解决的企业问题范围，一般可将数据仓库分为下列 3 种类型：操作型数据库（ODS）、数据仓库（DW）和数据集市（DM）。

- **操作型数据库（ODS）**既可以被用来针对工作数据做决策支持，又可用做将数据加载到数据仓库时的过渡区域。与 DW 相比较，ODS 有下列特点：ODS 是面向主题和面向综合的；ODS 是易变的；ODS 仅仅含有目前的、详细的数据，不含有累计的、历史性的数据。
- **数据仓库（DW）**为通用数据仓库，它既含有大量详细的数据，也含有大量累赘的或聚集的数据，这些数据具有不易改变性和面向历史性。此种数据仓库被用来进行涵盖多种企业领域上的战略或战术上的决策。
- **数据集市（DM）**是数据仓库的一种具体化，它可以包含轻度累计、历史的部门数据，适合特定企业中某个部门的需要。

主数据

主数据（Master Data, MD）指系统间共享数据（例如，客户、供应商、账户和组织部门相关数据）。与记录业务活动，变动较大的交易数据相比，主数据（也称基准数据）变化缓慢，一般每年的变化在 20% 左右。在正规的关系数据模型中，交易记录（例如，订单）可通过关键字（例如，订单或发票编号和产品代码）调出主数据。根据主数据管理实施的复杂程度，参照 Jill Dyche, Evan Levy 的观点大体可以把主数据管理可以分为五个层次，其中 Level 3（通过集中的总线处理，类似于翻译器）可以实现企业内任意两个系统交换数据。Level 3 是将数据转换逻辑集中化和标准化，它支持主参照数据的分布式存在（即分布的主数据存储，集中而标准的主数据转换），Level 3 打破了各个独立应用的组织边界，使用各个系统都能接受的数据标准统一建立和维护主数据(MDM)。而最高级别 Level 5（企业数据集中），当主数据记录的详细资料被修改后，所有应用的相关数据元素都将被更新，本级别可以通过 SOA 的架构平台实现。

通过对以上几个概念的简单分析，可以发现，SOA 虽然解决了系统之间的数据实时交互的问题，但是数据的集中，大数据量的数据同步以及主数据管理等问题还没有解决，即使建立了 SOA 的应用架构，仍然需要进一步进行数据仓库的建设和主数据的管理等。于是有了建立统一的应用集成平台和数据（信息）集成平台的概念。

SOA 和 DW 结合的企业架构

把数据和服务作为企业的资产

通常，软件重用（Software Reuse）是利用事先建立好的软产品创建新软件系统的过程，早在 1968 年的 NATO 软件工程会议上就已经提出可复用库的思想。重用形式上可以分为二进制、源代码、设计、分析等四个层面的重用，其中基于二进制代码的重用最为重要，从当前来看，基于二进制和源代码实现重用的方式主要有函数库（面向过程）、对象（面向对象）、服务（SOA）等方式。基于设计和分析实现重用的方式主要是通过模板进行定制，当前主要采用 MDA 的方式，由设计文档直接生产代码。

要实现基于二进制代码的重用，最重要的一个方法就是实现分离，通过分离，将共用的或者相关性不紧密的功能分离出来，如操作系统、数据库、应用中间件、公共组件等，都是在软件发展历史过程中逐步从一个完整的一体化的系统中分离出来的。这也符合社会分工

的要求，每个企业有多个开发商，即使同一个开发商，由于需要多人的分工合作，这时分工不同部分之间的标准就显得尤为重要。因此，可以说软件的核心是重用，方法是分离，关键是标准。围绕这个思路，在本架构中将企业的所有数据独立出来，基于数据架构，建设数据集成平台；将企业的所有应用组件分离出来，基于服务总线，建设应用集成平台。数据集成平台和应用集成平台共同组成企业的业务基础平台。未来的企业只有一个数据库，一个应用，所有的用户登陆一个系统可以完成所有的工作，系统的开发则可以基于一个业务基础平台，由多个厂商共同完成，形成一个企业的“云”，其中数据和 Web 服务是最核心的两个资产。

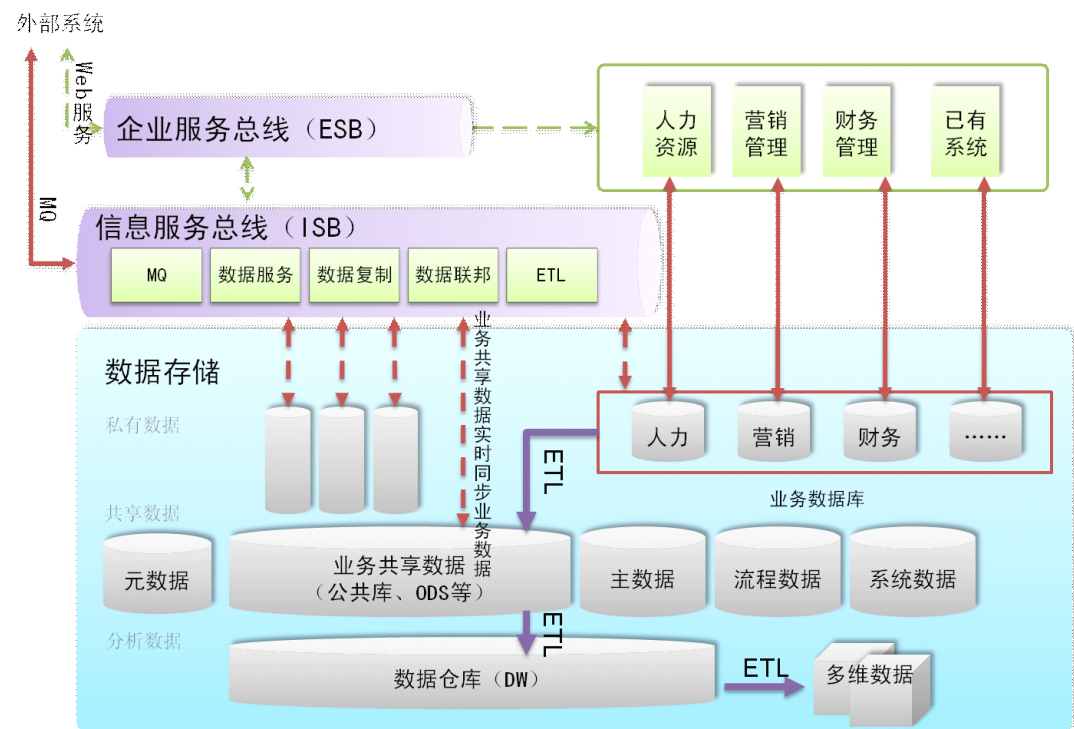
应用集成平台由企业服务总线（ESB）和公共的业务组件组成。ESB 将不同的组件互相衔接起来，在两个或更多的组件或系统之间实现无缝集成，使整个信息系统就像一个整体一样。通过统一购买、开发或者封装已有的组件和系统建设公共服务组件，形成企业共享的公用服务平台，避免重复开发、重复购买、标准不一致等问题。其中公共服务组件包括了门户组件、统一认证组件、工作流组件、GPS 组件、GIS 组件、BI 组件、通用报表组件等。

关于企业服务总线（ESB）和业务组件，相关的介绍很多，本文不做过多的描述，以下重点对数据存储层进行规划设计。

数据集成平台

对于一个企业来说，最理想的状况只有一个数据库，一个应用，但是需要考虑不同业务系统间的相互影响，例如：一个业务系统的性能问题会影响到其它系统，业务的新增、变更都会对其它系统有一定的影响，争抢已有系统的 I/O、CPU 资源等，因此，一个数据库是逻辑上一个库，可以物理上分开部署，包括采用数据库分区和数据库集群等技术；应用可以通过集群的方式解决性能问题，应用集群运行所有的业务组件，组件之间通过数据共享或者 Web 服务调用的方式实现互联。如果是遗留系统或者是产品化的系统，则通过主数据管理、SOA 集成以及数据仓库的 ETL 等实现集成，关于数据集成平台，如下图所示：

图 2. SOA 的 DW 架构



数据存储层规划

为了保证业务系统的性能同时实现数据的共享、数据分析的需要，将数据存储层的数据分为三个层次：私有数据层、共享数据层和分析数据层。

(三个层次准确的命名应该是私有操作数据层、共享数据层、私有分析数据层)

1、私有数据层：由一组业务专用数据库组成，这一层的数据集主要用于支撑企业的运营，是典型的操作型数据环境。包括经营、财务、人力资源、资产管理等已有系统的数据库，其主要职能在于支撑企业日常的经营和管理活动的运营需要。未来新的业务组件基于共享数据层进行开发，不在共享数据层的数据也在私有数据层，特别是新的业务组件自己内部处理过程中产生的数据。私有层的数据，其他的系统或者业务组件除了通过 web 服务调用之外不能直接访问。

2、共享数据层：该部分整合所有企业内外数据源，基于企业完整的数据架构进行建设，是所有已有系统和业务组件共享的数据库。共享层的数据结构清晰，基于业务对象，易于理解，和 Web 服务一样，是企业的资产，私有数据层的数据通过实时或者准实时的方式同步到共享层或者基于联邦技术直接基于共享层进行开发。共享层基于主题建模的明细数据和部分汇总数据，是新的系统的业务数据库和数据计算、企业报表的数据源，是对外数据的统一接口。

共享数据层包含业务共享数据、主数据、系统数据、流程数据和元数据。其中业务共享数据主要包括基于主题的数据模型，是交易类的数据，特点是数据变化快，数据量大；主数

据是基础数据，数据变化慢，数据量小，但是查询量大；系统数据主要包含用户数据、功能数据、用户权限数据等，和门户系统结合；流程数据是跨系统的流程数据，是未来 workflow 处理、流程监控，和审批平台的公共流程数据。

业务共享数据库定位为运营数据存储和业务支持，负责收集企业各系统中的数据，统一存储在业务共享数据库中，作为企业的共享数据。业务共享数据库采集了企业各系统的业务数据，对源系统的数据质量进行审计监控，同时按照企业统一的主题域数据模型对数据进行整合转换，并为其它业务应用系统提供跨域的数据共享。由业务共享数据库统一向各业务系统提供数据共享服务，同时业务共享数据库也可以作为新的系统开发的业务数据库。考虑到业务共享数据库是比较中立的数据，在基于业务共享数据库进行新的系统开发的时候，建议跟业务无关的数据单独建表，并和业务共享数据库的数据区分开，作为新系统自己的数据，存放在私有数据层。对于现有的系统，在升级改造的时候将现在的通过 ETL 抽取的方式改为由原系统直接集成的方式，将数据存放到业务共享数据库中。业务共享数据库也作为 DW 的理论唯一数据源，为 DW 提供高质量的数据；DW 中的挖掘和分析结果也要回写到业务共享数据库中。

3、分析数据层：该层的数据集主要用于支撑企业的管理和决策需要，是典型的分析型数据环境。它从数据仓库中提取数据并整合管理决策所需要的数据集。通过数据仓库的建设，对现有各类分布数据源进行数据集中，经过进一步对数据的清洗、过滤、转换后加载至数据仓库中，为数据的集中存储管理和分析利用提供数据支撑环境。

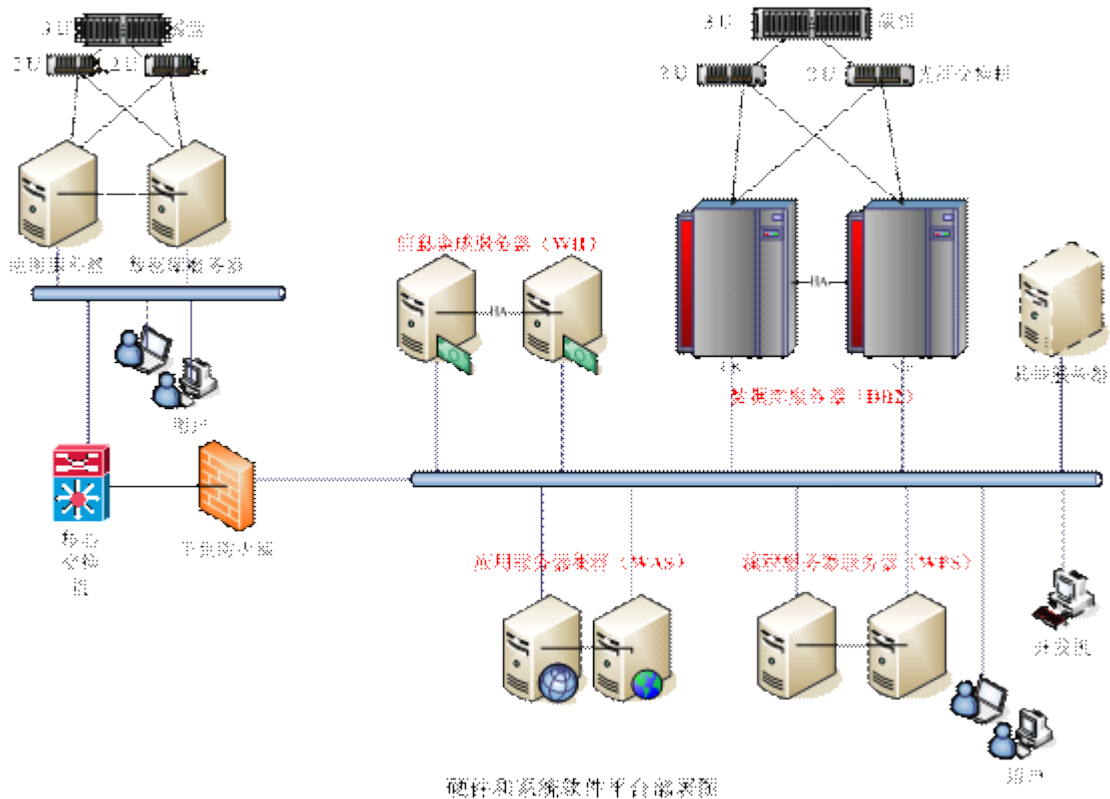
以上通过基于软件重用的思想，描述了需要构建数据集成平台的概念，并对数据平台的功能和数据存储做了说明，通过分层的数据管理，建立企业统一的数据集成平台。

基于 **IBM** 产品体系的实现

技术架构概述

根据前文提到的业务基础平台，基于 IBM 的产品体系，在实际搭建的时候，需要的服务器包含数据库服务器（产品：DB2）、应用服务器（产品：WAS）、流程整合服务器（产品：WPS，实现服务总线 and 流程编排）和信息集成服务器（产品：WII，实现数据的 SQL 复制或者 Q 复制），如下图所示（本方案是建议方案的一种，如 ESB 可以选择 WESB、DataPower 或 WebSphere Message Broker 等，需要根据不同企业特点进行选择）：

图 3. 网络拓扑图



产品实现

数据库服务器 (IBM DB2 Enterprise Server, DB2)，DB2 在本方案中主要实现数据存储层的建立，包含私有层数据库、共享层数据库和分析层数据库。如前文所述，考虑到遗留系统、产品化软件，特别是性能问题，在实际部署的时候需要考虑私有层数据库可以单独在一个表空间甚至是一个独立的磁盘中进行存储，保证业务组件能够不受其他业务组件的影响，数据存储层能够灵活的扩展，适应大数据量、多系统的处理。

DB2 提供多平台支持，因此能够在各种系统平台上发挥该产品的功能。DB2 能够管理多种数据类型，并能联合各类 IBM 系列数据库上的关系型数据。DB2 允许数据库管理员把数据库划分成若干称作表空间 (table spaces) 的部分，表空间可以单独管理，这就大大增强了管理特大型数据库的能力，它能包含上兆字节数据。DB2 在 SMP (对称多处理器) 或 MPP (并行节点机) 环境下，都可充分发挥其并行处理能力，特别是当数据量累计到一定程度，通过 Cluster (集群)，作为并行关系数据库它允许把单个数据库映像散布到多个系统上，从而能利用所有系统的处理能力以满足用户对数据的需求。DB2 可以在并行处理的多个节点上同时运行某一查询，从而提高查询性能，必要时它可以重新编写查询以优化性能，从而保证了可以实现一个数据库的设计理念。

通过多种数据类型、多平台、表空间、集群处理、并行处理等技术，可以实现数据存储层的统一管理。

信息集成服务器 (IBM WebSphere Information Integrator, WII) 信息集成服务器实现信息总线功能, 主要包含数据联邦和数据复制功能, 通过数据联邦和数据复制实现私有层数据和共享层数据的同步。

联邦技术能够统一地访问以任何格式(结构化的和非结构化的)存储的任何数字信息。WII 支持多种数据源, 包括DB2/390, DB2/400, DB2 UDB、Informix、Oracle、Sybase、SQL Server等关系型数据库, 也包括非关系型的文本文件和Excel文件。通过对各种数据源的支持, 可以在逻辑上实现不同数据库的统一视图。

数据复制, 在多个厂商的数据库产品之间实现复制, WII 使用基于 SQL 的复制架构, 该架构可在管理计划、转换和分发拓扑结构中最大限度地提高灵活性和效率。在 SQL 复制中, WII 使用基于日志或基于触发器的机制捕获变更, 并将其插入到关系分级表中。然后应用过程异步处理目标系统的更新。WII 可以广泛应用于填充仓库或集市、在不同应用程序之间维护数据一致性或者在总部和分支机构或零售商之间高效管理分发和合并。该技术采用经典的数据库复制体系结构实现, IBM 的基于队列的新复制架构(简称Q复制技术), 是一种高吞吐量、低延迟的复制方法, 它使用WebSphere MQ的消息队列在源数据库与目标数据库之间, 或者在源子系统与目标子系统之间传递事务。Q Capture程序通过读取DB2的恢复日志来获取你所指定的复制源表的变化情况; 继而, Q Capture程序将事务作为消息, 通过队列发送; 最后, Q Apply程序从队列中读取这些消息, 并将其应用于目标表。

通过WII, 将分散的数据库以数据联邦或者数据复制的技术集成起来, 实现数据层面的集成。

应用服务器 (IBM WebSphere Application Server, WAS), WAS所有业务组件的运行的容器, 实现统一的应用集成平台。应用集成平台需要主要考虑应用集群技术。

WAS完全提供了独立于硬件系统的集群功能。在WebSphere应用服务器中, 提供了两种方式的Cluster: 垂直的方式和水平的方式。它们分别是为了适应不同的应用环境而设置的, 垂直的Cluster允许在一台机器上实现动态负载均衡; 而水平的Cluster允许在多台机器上实现动态的负载均衡。二者可以有机的协调工作, 为复杂环境的应用实现负载均衡提供了强有力的保证。WebSphere使用的是一种Nanny -- Administration Server -- Application Server 的体系结构, 使用该结构, 所有的Application Server(应用服务器)将由一个 Administration Server(管理服务器)来监视管理, 一旦由于一个特殊的原因(例如进程被杀掉)导致应用服务器停止, 管理服务器将自动重新启动该应用服务器; 同样, Administration Server (管理服务器) 由一个“Nanny(保姆)”进程管理, 该“保姆”进程一旦检测到管理服务器停止也将自动重新启动它。这样WebSphere从体系结构上就最大限度地保证服务器程序的可靠稳定运行。

通过WAS的独立硬件的集群功能, 实现垂直或者水平的两种方式, 从而保证业务组件统一管理。

流程整合服务器 (IBM WebSphere Process Server, WPS) 服务总线和流程编排, 实现基于Web服务的数据同步, 业务组件之间的服务调用等。

WPS 是一个基于 WAS 的全面的面向服务的体系结构的集成平台。面向服务的体系结构提供了动态开发和修改集成应用程序的功能。通过面向服务的体系结构, 可以将现有应用程序与更新的应用程序相集成, 以便它们透明地协同工作。WebSphere Adapters 包括特定于应用程序的适配器(例如: 用于 Siebel、SAP 或者 PeopleSoft 的适配器), 以及技术适配器(例如: 用于关系数据库或者平面文件的适配器)。WebSphere Adapter完全符合JCA 1.5 的架构, 可以方便地在WPS 中使用。通过IBM提供的丰富的WebSphere Adapter产品,

可以轻松地操作已有业务系统中的数据，从而实现对现有IT资产的重用。WebSphere Process Server 中的唯一体系结构允许将业务功能封装到各个模块，然后单独进行更新。例如，可以使用包含用于实际审批的人工任务的审批模块，随后使用包含业务规则的另一个审批模块替换它。这一更改对于该模块的使用者是完全透明的。此外，封装的概念确保了数据和接口定义在使用它们的位置封装。例如，可以隐藏如何在模块内的后端系统中表示使用者的细节，而模块本身将具有一般业务对象的通用接口作为数据公开。这一规范的数据表示还启用了任何给定集成应用程序中的高度重用。

通过 WPS，搭建企业服务总线，实现基于 Web 服务数据同步和组件之间的整合。

以上通过对技术架构总体概述以及采用对应的IBM相关产品的具体实现，根据IBM产品所具有的产品特性，论证了SOA和DW结合的企业架构实现的可能性。

总结

本文基于企业架构（EA）、面向服务体系架构（SOA）、数据仓库（DW）、主数据（MD）等概念，构建了数据集成平台和业务集成平台共同组成的企业级业务基础平台，重点描述了基于 SOA 的数据集成平台中数据存储层的建设方案，提出了将数据和服务作为企业资产管理的思想，方案实现了数据和应用一体化设计，并在最后给出了基于 IBM 产品体系建设的建议方案和产品的定位。



中国 [选择]

dW 全部内容

搜索

[首页](#) [解决方案](#) [服务](#) [产品](#) [支持与下载](#) [个性化服务](#)[developerWorks 中国](#) > [SOA and Web services](#) > [文档库](#) >

developerWorks®

面向服务的体系架构 (SOA) 和业务组件 (BC) 的思考

肖建国, IT 咨询顾问, 浪潮软件

发布日期: 2010 年 3 月 08 日

面向服务体系架构 (SOA) 和业务组件 (BC) 的思考

最初由 IBM developerWorks 中国网站发表, 网址 <http://www.ibm.com/developerworks/cn>本文 http://www.ibm.com/developerworks/cn/webservices/1003_xiaojg_soabc/博客: <http://xiaojg.spaces.live.com>

肖建国

2010-03-08

摘要

在基于面向服务体系架构 (SOA) 中, “组件化” 是一个很重要的概念, 如何进行 “组件化” 开发是搭建企业级业务基础平台时需要考虑的一个重要课题, 本文通过建立业务组件 (BC) 接口模型及内部结构模型, 提供了一个在新开发系统环境下基于 Web 服务和 OSGi 标准的组件化开发模型。

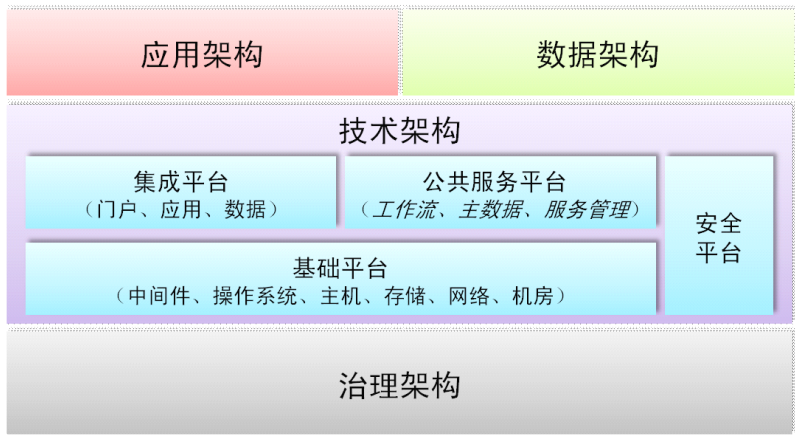
什么是业务组件 (BC)

组件化、模块化是软件开发中一个很重要的概念, 基于面向服务体系架构 (Service Oriented Architecture, SOA) 下, 如何实现组件化, 有各种实现方式, 下面通过对各种组件概念的对比, 从技术角度提出业务组件 (Business Component, BC) 定义, 并结合对总线模式的分析, 给出企业服务总线 and 类总线的实现方案。

企业架构 (EA)

关于企业架构 (Enterprise Architecture, EA) 和面向服务体系架构 (SOA) 在《面向服务体系架构 (SOA) 和数据仓库 (DW) 的思考》(以下简称《SOA 和 DW》) 一文中做了介绍, 企业架构包含企业战略、业务架构、IT 战略、IT 架构四个部分, IT 架构如下图 IT 架构模型所示, 包含数据架构、应用架构、技术架构和治理架构等四个方面, 其中技术架构包含集成平台、公共服务平台、基础平台 (软件和硬件、网络) 和安全平台等, 《SOA 和 DW》着重对如何构建数据架构特别是数据存储做了详细的阐述, 本文基于《SOA 和 DW》进一步对如何搭建 SOA 体系进行研究, 将着重描述如何基于可扩展的、灵活的企业级的集成平台、公共服务平台进行组件化开发。

图 1. IT 架构模型



业务组件 (BC)

当前，提到组件 (Component) 的有很多概念，比如分布式组件 DCOM、J2EE、CORBA 等，IBM 的业务组件模型 (Component Business Model, CBM)，SOA 中的服务组件架构 (Service Component Architecture, SCA) 等。本文提到的业务组件 (Business Component, BC) 定义为一个可以独立运行的系统或者模块，业务组件的目的是以方便业务组件独立升级和减少不必要的组件之间的交互为基本原则，通过一定程度的分离，实现《SOA 和 DW》中提到的重用 (SoftWARE Reuse)。

如果业务组件是共用的，是其它业务组件需要重用的，称之为公共业务组件 (简称公共组件)，所有的公共组件组成企业架构中技术架构的公共服务平台，比如主数据管理、系统管理、统一认证管理、通用报表等，这些都是公共组件。

组件业务模型 (CBM)

组件业务建模 (Component Business Modeling, CBM) 是 IBM SOA 构建的一个方法论，通过将组织活动重新分组到数量可管理的离散、模块化和可重用的业务组件中，从而确定改进和创新机会，把业务从领导，控制和执行三个方面进行模块化分析，从而有效的实现业务的有组织的提供服务的能力。CBM 的价值是提供一个可以推广的框架，用来创造顺应组织战略的可以运营的指导方向，同时 CBM 也用来按照业务和资源的优先级别和相互关联的程度来构建和顺应战略的发展方向，其中包括建立一个沟通的机制来理解整个业务发展的方向。通过 CBM 建立了 SOA 的规划的方向，为实施 SOA 奠定基础。

本文所提到的业务组件在粒度上基本对应着组件业务模型 (CBM) 的粒度，但是本文中的业务组件 (BC) 更多从技术实现角度考虑，或大于，或小于业务组件模型 (CBM) 提到的业务组件概念。

服务组件框架（SCA）

服务组件框架（Service Component Architecture, SCA）由 BEA、IBM、Oracle 等中间件厂商联合制定的一套符合 SOA 思想的规范。服务组件框架（SCA）提供了一套可构建基于面向服务的应用系统的编程模型，它的核心概念是服务及其相关实现。SCA 组件组成程序集，程序集是服务级的应用程序，它是服务的集合，这些服务被连接在一起，并进行了正确的配置。在 SCA 标准下，SCA 由域（Domain）、组合构件（Composite）、构件

（Component）三个级别组成，构件对应着细粒度的 Web 服务，域对应着粗粒度的 Web 服务。SCA 程序集运行在两个级别：第一种情况，程序集是“大规模编程”（Programming in the Large 或 Megaprogramming）的一组松散连接的服务组件；另一种情况，程序集是“小规模编程”（Programming in the Small）内的一组松散连接的组件。二者的区别在于，“大规模编程”对应着应用，“小规模编程”对应着模块，一般来说，服务组件对应着“小规模编程”，即模块的概念。

本文所提到的业务组件（BC），是比 SCA 组件更大范围的概念，这几个概念的颗粒度从大到小的排列顺序如下：**系统**（每个企业只有一个系统）、**应用**、**业务组件（BC）**、**模块**、**SCA 组件**（粗粒度服务）。

总线模式（Bus）和 SOA、OSGi

总线（Bus）：一般指通过分时复用的方式，将信息以一个或多个源部件传送到一个或多个目的部件的一组传输线。基于总线模式的有很多应用，在微机的技术中，有三种总线，地址总线 Address Bus，数据总线 Data Bus，控制总线 Control Bus。在通信架构下，交换机也是一种总线，在 SOA 中，总线一般指企业服务总线（Enterprise Service Bus，ESB），企业服务总线可以连接所有协议的各种接口，但是最理想的是基于 XML 的 Web 服务标准。

OSGi——Open Service Gateway Initiative，1999 年 OSGi 联盟成立，旨在建立一个开放的服务规范，为通过网络向设备提供服务建立开放的标准，是开放业务网关的发起者。

OSGi 是一个 Java 框架，该框架能装载以 Bundle 为单位的资源。Bundle 能提供服务或响应处理请求，而他们之间的依赖都是被管理起来的，正如一个 Bundle 能从容器中获得它所需要的管理。每个 Bundle 都可以有它自己的内部类路径，所以它可以作为独立的服务单元。所有的这些符合 OSGi 规范的 Bundle 理论上都可以安装在任何符合 OSGi 规范的容器中。

OSGi 具有可动态改变系统行为，热插拔的插件体系结构，高可复用性，高效性等等。在 J2EE 环境下，基于总线（Bus）模式的思考，可以进一步推广到 Java Class，基于 OSGi 的微内核，建立一个类总线（Java Class Bus，JCB）。

通过以上概念的分析，我们可以看到，本文提到的业务组件（BC），是指具体的一个软件实现，业务组件（BC）跟 IBM 的业务组件模型（CBM）中提到的业务组件有一定的对应关系，但是一般来说，业务组件（BC）可能包含 CBM 中的多个业务组件或者一个 CBM 的业务组件封装成多个业务组件（BC）。另外 CBM 更多的是从业务角度来考虑，是业务上的概念，业务组件（BC）则是从技术实现角度考虑。服务组件框架（SCA）定义的粒度和业务组

件（BC）相比来说，SCA 划分的还是很细，业务组件（BC）是更粗粒度的一个软件实现概念。

业务组件（BC）模型

根据业务组件的作用不同，可以将业务组件分成公共业务组件和普通业务组件，公共业务组件包含统一用户组件、统一认证组件、门户组件、流程组件、报表组件、BI 组件、GIS 组件等，这些组件的共同特点是多个业务组件或者系统会用到这个业务组件。

组件的粒度和对外接口设计决定了组件的可复用和松耦合（Loose Coupling）特性。粒度过大，灵活性小，难以实现复用，粒度过小，管理成本提升，使得复用性也很难改善；接口和实现的分离，保证各项业务组件在提供标准化的服务接口的前提下可以替换各种可选的实现，而不会影响系统其它部分的实现，接口设计不当，对于组件的耦合会有很大的影响。

业务组件的粒度

业务组件的粒度根据需要可以不同，既可能是独立运行的子系统，也可能是程序模块。业务组件是提高应用系统灵活性和复用的重要基础。业务组件粒度太小，造成组件数量多，组件之间交互多，管理困难，性能低下；业务组件粒度粗，功能复杂，功能之间关系紧密，升级困难（可以独立升级往往会作为确定一个业务组件范围的重要因素），很难实现重用。因此找到一个合适的业务组件粒度是很重要的事情。

根据前文所定义的业务组件定义，我们把整个企业的所有软件称之为系统，即一个企业只有一个系统；系统下面划分成若干应用，每个应用完成一个相对独立的业务功能，比如财务管理、人力资源管理，一般来说是一个厂商独立完成（后文还会提到，如果是基于一个业务基础平台，多个厂商可以在一个应用中）；应用下面划分成若干业务组件，业务组件是相对独立的功能，其可以进一步划分成若干模块，从而形成了系统—应用—业务组件—模块这样四个层次的模型。根据 SCA 的定义，模块下面可以进一步划分成程序集为更小的粒度。从软件复用角度来看，业务组件是独立部署的最小颗粒，模块是复用的最小颗粒。

除了业务组件需要粒度控制外，Web 服务的粒度控制也是一项十分重要的设计任务。通常来说，对于将暴露在整个系统外部的服务推荐使用粗粒度的接口，而相对较细粒度的服务接口通常用于企业和机构系统架构的内部。从技术上讲，粗粒度的服务接口可能是一个特定服务的完整执行，而细粒度的服务接口可能是实现这个粗粒度服务接口的具体的内部操作。虽然细粒度的接口能为服务请求者提供了更加细化和更多的灵活性，但同时也意味着引入较难控制的交互模式易变性，也就是说服务的交互模式可能随着不同的服务请求者而不同。如果暴露这些易于变化的服务接口给系统的外部用户，就可能造成外部服务请求者难于支持不断变化的服务提供者所暴露的细粒度服务接口。而粗粒度服务接口保证了服务请求者将以一致的方式使用系统中所暴露出的服务。

业务组件的松耦合设计

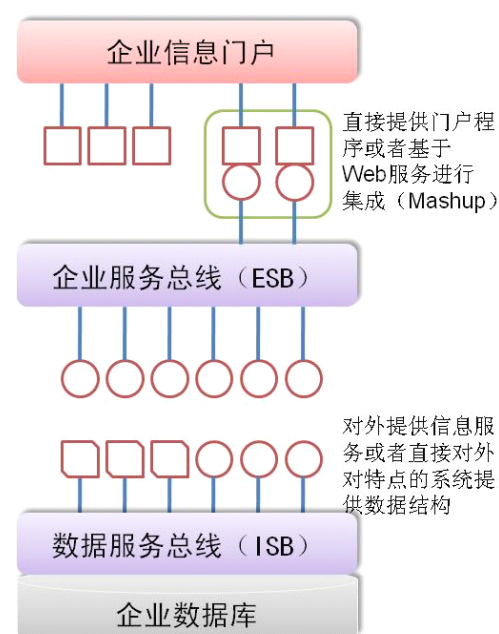
耦合性（Coupling）是程序结构中各个模块之间相互关联的度量，它取决于各个模块之间接口的复杂程度、调用模块的方式以及哪些信息通过接口。耦合性由松到紧可以分成七

种：非直接耦合（Nondirect Coupling）、数据耦合（Date Coupling）、标记耦合（Stamp Coupling）、控制耦合（Control Coupling）、外部耦合（External Coupling）、公共耦合（Common Coupling）、内容耦合（CONTENT COUPLING）。非直接耦合是指两个模块之间没有直接关系，这种耦合的模块独立性最强。数据耦合，彼此之间是通过数据参数（不是控制参数、公共数据结构或外部变量）来交换输入、输出信息的，模块之间的独立性比较强。标记耦合是指一组模块通过参数表传递记录信息，就是标记耦合，这要求这些模块都必须清楚该记录的结构，并按结构要求对此记录进行操作，应尽量避免这种耦合，它使在数据结构上的操作复杂化了。在业务组件设计模型中业务组件之间尽量实现非直接耦合（总线模式，推荐使用）和数据耦合（共享库模式，控制使用），通过定义清晰的 Web 服务进行交互，业务组件内部的模块之间可以通过标准化的 Web 服务或者数据表来进行共享。

业务组件接口模型

业务组件作为一个可以独立的运行的模块，通过一系列的接口，可以独立完成本业务组件的功能。业务组件通过接口，可以组装到和基于标准的**集成平台**上，并和其他的组件或者系统共同完成企业的业务。基于标准的企业**集成平台**包含企业信息门户（门户标准）企业服务总线（Web 服务标准）和数据服务总线（含数据模型），如下图所示：

图 2. 企业集成平台模型

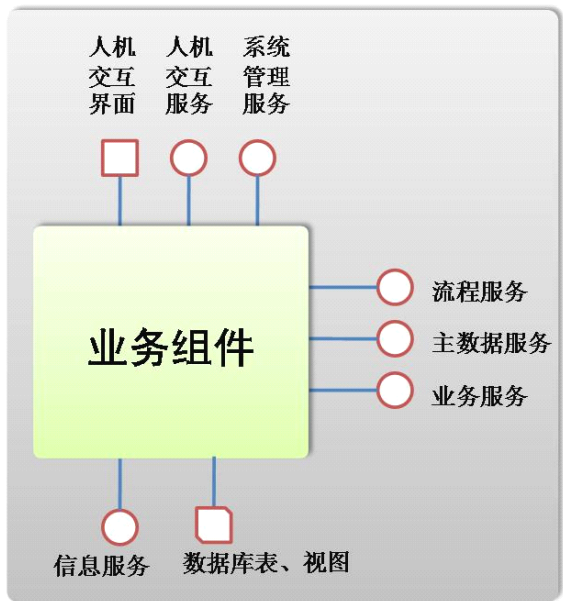


当企业**集成平台**建立之后，基于标准，业务组件就可以简单的以“插拔”的方式组合到整体应用中。

为了实现业务组件间松耦合、业务组件内部高内聚，可以将所有的业务组件抽象成一个模型，该模型的业务组件对外提供三种类别的接口：界面、服务、数据。界面主要是实现业

务组件和人之间的人机交互界面，服务是业务组件和业务组件或者系统之间的交互，是信息系统之间的交互界面，数据是业务组件和共享数据库之间的交互界面，其中服务根据作用又可以进一步分成三类：和人机交互相关的服务、和业务组件之间的交换以及和数据库之间的交换。如下图所示：

图 3. 业务组件接口模型



- **人机交互界面：**采用 Portlet 标准，对外提供标准的门户程序，通过门户集成平台进行门户集成进行。对外的门户程序可以以两种方式提供，一种是完全独立的门户程序，可以任意的集成到任何一个独立的门户界面，但是如果所有的界面都定制，考虑到性能和定制工作量比较大，可以采用另外一种方式，即把多个界面定义到一个门户程序中，可以将一系列的界面在一个门户程序中完成，减少配置以及管理的工作，使得系统更加易于集成。比如可以把客户信息展示作为一个简单的门户程序，仅仅实现客户信息展示，也可以把客户维护，客户信息展示、客户拜访管理、客户分类管理等所有客户相关的信息在一个门户程序中实现，并且在门户程序中以菜单的方式进行选择，相当于是内嵌了一个小的应用功能界面。
- **服务接口：**服务接口按照类型可以分为 6 种，其中人交互服务和信息服务比较特殊，分别实现人机交互和数据交换的功能，是以服务的方式提供人机交互界面和数据接口内容。
 - **业务服务，**业务组件为实现的业务组件核型功能的对外相关服务，是业务组件核心服务，主要用于本业务组件和其他的业务组件之间的业务交互，使得多个业务组件或者系统共同完成企业的业务流程。为了保证业务组件的高内聚，松耦合，要合理的规划业务组件对外提供的服务的粒度，即能保持灵活性，又可以保证对外提供的服务不至于太多，不宜管理。业务组件的 web 服务结构是企业业务组件规划中的最为重要的标准化功能，用于确定不同业务组件之间的边界。
 - **主数据服务，**主数据相关的服务，是共用的服务，主数据管理业务组件也是属于企业公共服务平台管理范围，是企业级的公共业务组件。

- 流程服务，涉及工作流程的服务，相关信息提供到 workflow 引擎，是共用的服务，流程管理业务组件也是属于企业公共服务平台管理范围，是企业级的公共业务组件。
- 系统管理服务，是由系统管理公共组件提供的服务，主要包含用户认证、权限管理等相关的服务，是共用的服务，系统管理相关业务组件属于企业公共服务平台管理范围，是企业级的公共业务组件。主数据服务、流程服务和系统管理服务是企业架构平台提供的公共服务，是集成平台的核心内容。
- 人机交互服务，将人机交互内容以服务的方式提供，通过处理后在界面层次统一展示，通过这种方式，可以实现将不同的业务组件的服务混搭（Mashup）成一个门户程序，而不是通过两个门户程序进行整合。人机交互服务和 Portlet 的差异是采用的标准不同，前者基于 Portlet 标准，后者基于 Web 服务标准；前者直接以界面的方式对外提供，后者主要提供数据（可以同时展示方式），通过前端的定制工具实现界面展示，通过这种方式，各组件或系统把需要展示的内容以服务的连接到 ESB，然后通过门户系统的界面整合，将不同系统的数据在界面进行统一展示，比如可以将财务系统的人员工资信息、人力资源信息等分别以服务的方式对外提供，然后在门户的界面整合工具在门户中统一进行展现，而不是通过 portlet 的方式实现。
- 信息服务，和数据库相关的服务，直接从数据库获取相关信息。由于业务组件的数据是私有的，为了保证业务组件的数据能够得到更好的应用，需要将业务组件的数据公布出来，基于企业的数据模型，把业务组件的私有数据公开为企业数据中的数据。信息服务可以采用实时、或者准实时的方式对外提供。在某些特殊情况下，可以认为业务组件不存放数据，业务组件仅仅是获得数据，处理数据，然后将数据在放到企业数据库中。
- **数据接口：**基于视图或者表直接对数据库进行操作，即业务组件对外提供一个直接访问数据库的接口，如果数据库结构是按照这个接口设计的，这个业务组件可以直接访问数据库，而不需要通过其它的服务，需要明确每个组件对表的读写权限，并进行严格管理，通过数据接口的方式，核心是需要建立企业数据架构，建立共享的数据结构。通过数据联邦、数据复制实现。一般来说，不建议这样实现，特别是跨应用的数据访问，尽量通过信息服务实现。

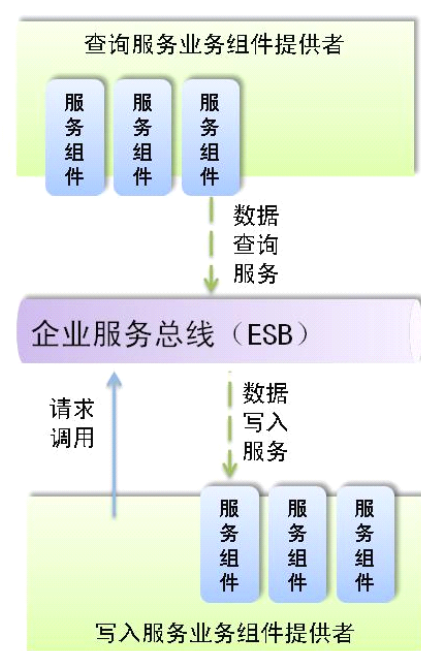
以上通过对业务组件模型对外提供的接口类型进行分析，规划了一个业务组件接口模型，所有的业务组件将对外提供以上三类对外的接口。

业务组件接口通信方式松耦合实现方式

Web 服务根据 Web 服务内部数据的流向可以分成两类，一是获取服务，即别的业务组件或系统调用 Web 服务，对外提供数据；二是写入服务，即其他业务组件或系统调用本业务组件或系统 Web 服务写入数据。前者比较简单，只要提供服务即可，但是对于后者，在实现过程中将会有两种实现方式：一是主动到 ESB 调用 Web 服务，读取数据，然后直接写入系统，二是提供 Web 服务到 ESB，等待业务组件或系统调用写入。采用第一种方式，不需要对外提供 Web 服务，直接调用 ESB 的 Web 服务即可，但是服务的调用以及写入是通过代码固化到系统中的；后者对外提供服务，不管调用系统是谁，只要调用既可写入数据，不会

受到外部系统的变化的影响。如果是采用直接到 ESB 调用服务，然后用代码写入的方式，一旦业务流程发生变化，比如改为别的系统或者模块直接写入，这种方式就无法适应，需要重新增加一个写入的 Web 服务，因此比较好的解决方式是提供对外的一个写入 Web 服务，写入 Web 服务是独立的，满足别的系统主动写入的要求，如果是自己主动写入，则采用先到 ESB 上请求获取数据（本服务仅仅是通知 ESB，要获取数据）然后由 ESB 实现对本次统的写入 Web 服务，这样，不管外部的系统如何变幻，接口是不变的，如下图所示：

图 4. 服务调用顺序



调用者以写入服务方式实现调用

除非是写入服务提供者业务需求非常明确，只有本系统调用才会写入，一般建议按照以上独立的写入服务方式来实现。采用独立的写入服务能更好的适应未来被动写入、或者写入操作需要经过评审或者确认之后的操作。

比如客户信息数据，如果是在 CRM 系统中创建，财务系统需要客户数据，有三种调用方式，一是财务系统直接到 ESB 调用 CRM 的客户信息查询 Web 服务，然后写入系统。二是事件机制，CRM 系统中的数据变化时，对外提供的客户信息变更服务，服务调用中传递的消息就是变更的信息，调用财务系统的写入服务。如果还有其它的系统需要客户信息，可以在 ESB 中定义出发布/订阅关系。三是财务系统先请求 ESB 调用 CRM 的查询服务，然后由 ESB 调用财务系统的客户信息写入服务，写入数据。如果未来业务流程发生变化，改由 CRM 直接将客户信息写入财务系统，则直接调用财务的写入服务即可，需要做的仅仅是配置一下 ESB 即可，现有的程序不需要改变。第一种方式下，如果改成 CRM 写入，财务系统需要重新编码，第二种方式如果别的系统来主动查询客户数据，需要另外增加一个客户信息的查询服务，第三种情况，无论是如何改变化，需要的仅仅是增加一个请求调用即可，对所有的系统影响最小，因此是受外界需求发生变化后影响最小的方式，更好的解决了松耦合的问题。

业务组件接口事务处理—无状态的会话设计

为了保证业务组件的独立和松耦合，SOA 中的具体服务应该都是独立的、自包含的请求，在实现这些服务的时候不需要前一个请求的状态，也就是说服务不应该依赖于其他服务的上下文和状态，即 SOA 中的服务应该是无状态的服务。不同的业务组件之间是采用 Web 服务的方式进行交互，对于事务的控制很困难，应尽量将事务控制在一个业务组件中（关于事务的控制，详见下文关于组件内部松耦合设计的方案），如果是在不同的业务组件，需要事务控制，则考虑采用流程编排的方式实现，基于 BPEL 实现事务控制。

J2EE 架构下业务组件（BC）实现

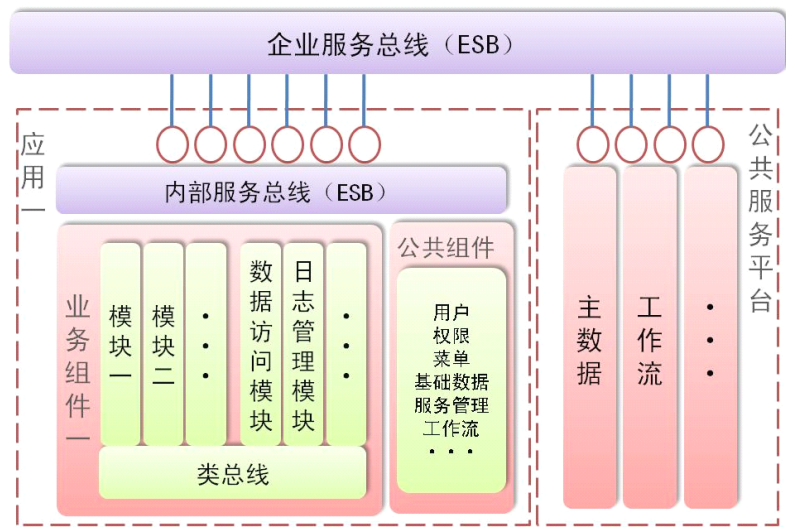
业务组件松耦合设计—OSGi

业务组件以 Web 服务的方式提供接口，通过企业服务总线连接，业务组件内部为了实现高可复用性和高效性，采用基于 OSGi 标准进行构建模块，实现内部模块之间的松耦合，即在业务组件内部基于 OSGi 标准进行模块化设计，将业务组件进一步分解为松耦合的模块（Bundle），使得业务组件本身更加灵活。

基于 OSGi 标准，业务组件内部的模块通过一个具有动态加载类功能的微内核连接，统一管理各个模块，为了便于管理，将不同模块之间的类接口采用服务注册的方式进行管理，具有类动态加载功能的微内核和类接口管理组成类总线（JCB）的基本功能，为了更好的实现重用，有些模块是共用的，比如数据访问模块、日志管理模块等。

在一个应用中，不同业务组件公用的功能，作为应用内部的公共组件，一个应用中部署一个公共组件即可，各个业务组件共用。在一个业务组件中，不同模块公用的功能，作为公共模块，相当于工具类，公共模块需要在每个业务组件中部署。公共服务平台作为企业级的公共服务对外提供企业级的 Web 服务，比如主数据管理等。业务组件构成如下图所示：

图 5. 业务组件模型（公共类—公共组件—公共服务平台）



注：
业务组件中的公共组件和公共服务平台的差异是公共组件是应用内部的，提供应用级别的服务；公共服务平台则是面向企业整个系统的，是提供系统级的服务，两者有时候可以互相替换，主要是看其处于那个级别。

基于 IBM 产品体系的实现

本文提到的集成平台，基于 IBM 的产品共体系在实际搭建的时候需要包含应用服务器（产品：WAS）、流程整合服务器（产品：WPS，实现服务总线和流程编排）等，关于集成平台的详细描述，详见《SOA 和 DW》一文中“基于 IBM 产品体系的实现”的描述。

业务组件（BC）在 WAS 中的部署

首先来看一下在 J2EE 架构模式下文件格式（以 WAS 为例）。在 J2EE 架构下，文件格式有三种，分别是 EAR、WAR、JAR，另外还有一个特殊的 JAR 即基于 OSGi 标准的 Bundle，共四类文件：

- **EAR 文件（file Enterprise Achieve）**除了包含 JAR、WAR 以外，还包括 EJB 组件、部署文件 application-client.xml、web.xml、application.xml 等全部企业应用程序。
- **WAR 文件（file web Achieve）**包含 Servlet、JSP 页面、JSP 标记库、JAR 库文件、HTML/XML 文档和其他公用资源文件，如图片、音频文件等全部 Web 应用程序。在一个 EAR 文件中可以有多个 WAR。（在 WAS 环境下，如果设置在 EAR 中用一个类加载器，这样不同的 WAR 之间可以直接调用 Java Class，WAR 之间是紧耦合的，不建议采用。）
- **JAR 文件（Java Achieve）**按 Java 格式压缩的类包，包含内容 class、properties 文件，是文件封装的最小单元。但是普通的 JAR 文件，只是一个文件的集合，没有具体的意义。

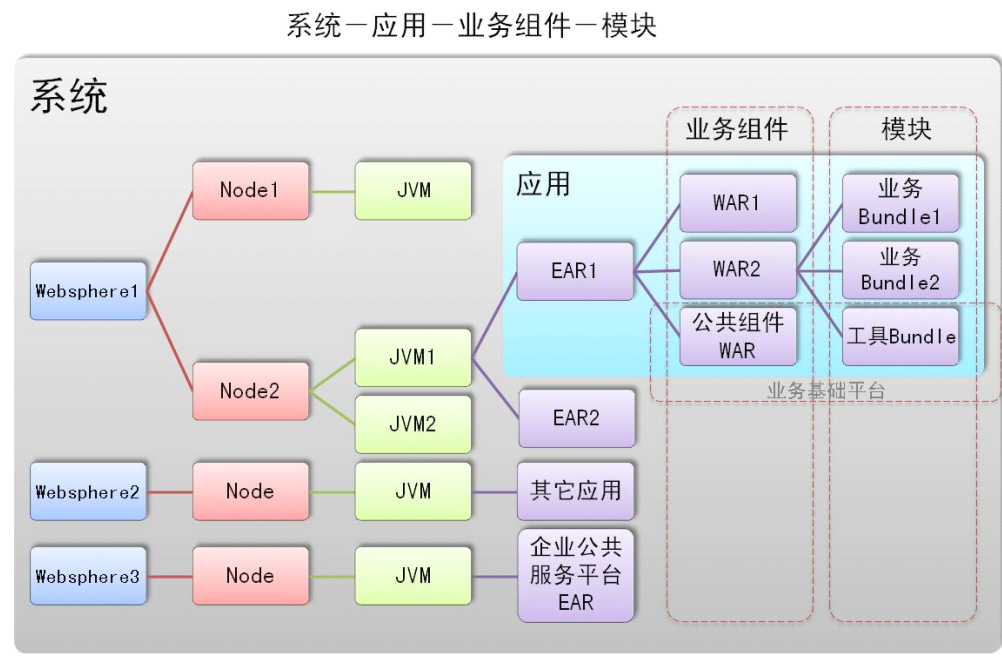
- **Bundle**，基于 OSGi 标准的一种特殊的 JAR 文件，每个 Bundle 也包含一个 META-INF/MANIFEST.MF 文件，这个文件会宣布导出哪些包 (Package) 以及导入哪些包。只有那些导出包中的类才能被其他 Bundle 所使用，而其他包都只面向包的内部成员，包里的类也只能在自身 Bundle 中使用。一个简化的处理思路是直接采用包 (Package)，但是无法实现类的动态加载和对接口进行管理，不具有松耦合特性。（WAS 从版本 6.1 开始支持 OSGi）

从以上文件结构可以看到，JAR 之间可以进行类的调用，很容易实现不同 JAR 之间的事务处理，且具有更高的性能，结合 OSGi，通过“类总线”进行管理所有 JAR (Bundle) 对外开放的接口，从而以“总线” (Bus) 的方式管理不同 JAR 之间的类调用。不同的 WAR 之间各种资源不共享，为了实现重用，需要设置一些公共模块，即公共 Bundle；不同的 WAR 之间交互，需要以 Web 服务的方式进行连接，需要建立企业服务总线 (ESB) 管理所有的 Web 服务，实现 WAR 之间调用。在一个 EAR 中，可以设置不同 WAR 之间共享 Session，从而方便的实现单点登录以及其它的公共信息，这些信息将在这个 EAR 环境中共享。

根据前文所述业务组件 (BC) 的定义，业务组件适合于在 WAR 文件层面进行划分，即一个 WAR 作为一个业务组件，一个或者几个 WAR 组成一个应用 (EAR)，多个应用构成企业的系统；业务组件内部进一步划分为多个模块 (Bundle)，每个模块相对独立，可以独立维护，独立升级和安装，以插件的方式通过类总线进行关联。为了实现重用，在 EAR 层面，将企业级的业务组件单独部署，比如主数据、统一认证、工作流等，建立公共服务平台；在 WAR 层面，各厂商的公共的业务组件单独封装在公共组件 (WAR) 中，如系统管理、系统参数管理等。在模块级别采用公共模块的方式，在各个 WAR 中以工具模块 (Bundle) 方式提供，如数据库访问、日志 (Log4j) 等。公共组件 (WAR)、内部服务总线 (ESB) 和类总线 (JCB)、工具模块 (Bundle) 组成应用系统的业务基础平台 (Business Platform)（如浪潮的 Loushang 平台）。

在系统部署的时候，一台服务器上安装一个 Websphere 实例，一个实例根据主机的性能可以安装多个节点 (Node)，每个节点 (Node) 可以安装多个虚拟机 (JVM)，每个虚拟机可以安装多个 EAR 应用，每个 EAR 有多个 WAR，不同 WAR 之间文件不会冲突，WAR 内部采用 OSGI 标准分成多个模块 (Bundle)。不同公司的系统是不同的 EAR，同一个公司可以有多个 EAR。如下图所示：

图 6. WAS 部署模型（系统—应用—业务组件—模块）



EAR 之间数据交换采用通过企业级服务总线（ESB）的 Web 服务，大数据量数据共享在数据总线上通过企业级的共享数据库进行共享，通过数据总线共享的数据不是实时的，是有一定的延误或者准实时的。共享库是企业的资产，不隶属于任何一个厂商。

WAR 之间数据交换采用通过企业或者内部的服务总线（ESB）的 Web 服务，不同的 WAR 共用一个数据库，但是数据表根据 WAR 的职责进行划分，每个 WAR 可以只读所有的共享的数据表，但是只能写自己控制的表，对其它 WAR 控制的表、表结构很复杂或者易变的表的读作操，都应通过 Web 服务进行，以实现 WAR 之间的松耦合。一个 EAR 中的共享数据库（对企业来说是私有数据层）在各个 WAR 之间结合更加紧密，一般采用直接访问的方式，不在私有数据层存放共享库数据，私有数据仅仅是一些控制类或者跟业务无关，不需要共享的数据。企业级共享库一般从性能、不同厂家便于控制等角度考虑，数据同步是准实时的，数据在各个 EAR 的私有数据层一般会有一个拷贝，让各个 EAR 之间相对更加独立。以上仅仅是一般原则，如果是一个厂商，两个 EAR 之间也可以像一个 EAR 共享一个数据库。EAR 和共享库之间的关系如下图 EAR 和共享库的关系所示：

图 7. EAR 和共享库的关系



Bundule 之间数据交换可以类似 WAR 的 Web 服务调用，也可以直接通过类总线，调用 Bundule 对外发布的接口，特别是需要具有事务处理能力和更高的性能要求的情况。一个 WAR 内原则上不再划分数据表的控制，由 WAR 自己内部进行管理。

一般来说，WAR 是相对比较稳定的，不需要完全进行替换升级，日常的变更是通过 Bundle 来实现的，由于 Bundle 在应用方面是基于插件的方式进行开发的，数据方面由于业务组件本身是基于标准信息服务或者作为企业资源的开放的数据表、视图（Web 服务和数据模型，作为企业的两个资产），因此可以实现热插拔，保证了系统可以连续运行，而不至于因为系统升级而影响到业务的运行或者最小程度的营销业务的运行。

在实际部署的情况下，有可能会出现整个企业只有一个 EAR（功能简单，最极端情况），或者一个厂商把应用分别部署在不同的 EAR，需要根据企业规模，主机部署等，从性能、易于管理等角度考虑。由于 WAR 之间是松耦合的，基于企业服务总线（ESB）和信息服务总线（ISB），可以实现灵活的组装，因此一个或者多个 WAR 可以进行灵活组装部署。

以上基于 J2EE 的应用服务器 WAS（IBM WebSphere Application Server，WAS），对 J2EE 架构模式下的文件体系进行分析，针对前文提到的业务组件模型实现提供一个实现建议。

业务组件（BC）实现举例

一般企业常用到的应用包含人力资源、协同办公、财务管理、营销管理、生产管理等几个主要的应用，根据前文所述架构，基于 J2EE 环境按照以下方式进行构建：首先，人力资源、协同办公、财务管理、营销管理、生产管理可以认五个应用，一般来说可能由五个厂商分别提供，因此可以作为五个独立的 EAR，分别进行部署，考虑到为减少不同厂商之间的影响，一般会分别安装在五台服务器上，或者安装在不同的虚拟机上，（有时候可能是一个厂商提供，会在一个 EAR 中，如财务管理和人力资源在一块，因此 WAR 是可以灵活选

择进行部署的)。五个系统之间如果需要实时交互,则完全通过企业的集成平台中的 ESB 进行交互。五个应用之间可以建立企业级的共享数据库。

对于营销管理应用(EAR)来说,其内部可以进一步划分成客户关系、供应商管理、购销存管理等多个业务组件,作为三个独立的 WAR,三个业务组件之间如果需交互通过营销管理应用 EAR 自己的 ESB(从企业角度来说,需要建立联邦 ESB)进行交互,或者通过企业的集成平台中的 ESB 进行交互。三者共用一个数据库,逻辑上把表划分成三个部分,分别由三个业务组件进行管理,每个业务组件有自己的控制表,对于其它业务组件控制的表只读,如果需要访问,如前文所述,通过其它业务组件提供的服务进行访问。

客户关系管理还可以进一步划分成市场管理、销售管理、服务管理等多个模块,为了便于管理,可以在此基础上再划分的更细一层,比如市场管理可以进一步划分成客户分类管理、客户群管理、客户拜访管理等模块(Bundle),模块(Bundle)之间可以通过类直接调用,不同的模块之间的调用通过类总线实现。从管理和系统升级调度考虑,模块不宜太少,也不易太多。

以上就一般企业常用到的功能按照前文所描述的模型基于 J2EE 架构上具体实现做了说明,可以作为组件化开发的一个参考。

总结

本文通过对业务组件(BC)的定义,特别是描述和当前相关的 CBM、SCA 等组件概念的关系,描述了基于 SOA 的组件化编程在技术上的具体实现,并对业务组件在对外接口、内部结构构成等技术要求做了说明,从而搭建一个灵活、易于扩展、易于维护的组件化开发模式,为企业搭建**集成平台**之后如何进行全新的组件化开发提供了一个参考。

基于 SOA 的业务流程管理（BPM）和工作流（WF）

基于业务组件模型的工作流模块设计

肖建国

摘要

当前基于 BPLE 的业务流程管理（BPM）以及基于 XPD L 的工作流（WF）都有成熟的理论和相应的产品支持，特别是在国内，工作流（WF）的应用十分广泛。本文从流程入手，结合业务流程管理、工作流、绩效管理、个人门户等概念，将业务流程管理和工作流结合起来，搭建企业级的跨系统的流程整合架构。

什么是流程

在面向服务体系架构（Service Oriented Architecture, SOA）中，流程是一个很重要的概念，其中业务流程管理包含了人工任务等，结合《面向服务体系架构（SOA）和数据仓库（DW）的思考》（以下简称《SOA 和 DW》）以及《面向服务体系架构（SOA）和业务组件（BC）的思考》（以下简称《SOA 和 BC》）中关于共享库、业务组件的设计，本文进一步给出了关于如何进行工作流管理组件的设计方法和实现。

流程和作业

流程（Process）是产生某一结果的一系列作业，是多个人员、多个作业按照一定的规则的有序组合，它关心的是谁做了什么事，产生了什么结果，传递了什么信息给谁。流程一定是体现企业价值的，没有价值的流程是没有意义的，因此每个流程都有其特定的绩效目标。在信息系统中，流程由若干作业（Operation）按照一定的规则组合而成，可以用业务流程图来描述，其目标通过绩效指标体现。作业是为了实现一个可定义的目标而进行的一系列活动，是业务流程的基本单元。在信息系统中，作业的前端表现为若干界面，后端由若干个服务按照一定的规则组合成一个个功能单元。

在本文中，流程是指企业运作的所有工作流程，企业的所有的活动都可以看作是一个个流程，流程是由若干个作业组成的，在 IT 技术上流程称为工作流，作业称为流程节点。

流程规范 XPD L 和 BP E L

在 IT 技术中，关于流程最早是以 WfMC 为代表的“业务流程开发商”，工作流管理联盟（WfMC）于 1993 年成立，他们主要拥护以 XPD L 作为描述语言来描述业务流程；之后是以 OASIS（Organization for the Advancement of Structured Information Standards，结构化信息标准促进组织）组织为代表的，被 IBM，MicroSoft，BEA 所拥护的 BP E L/BP E L4WS 规范；之后向来以规范著称的 OMG 组织也不甘示弱，联合 BPMI 组织，独辟蹊径以 Notation Specification 为入口，首先推出了 BPMN 规范，进而推出了 BPDM（Business Process Definition Metamodel BPDM）。

2003 年 4 月 BPEL 规范提交给了 OASIS 更名为 WSBPEL (Web Services Business Process Execution Language) 规范。此规范描述如何处理输入的消息, 它不是一个关于业务流程规格化定义的规范。简单的说, 可以将它看作 XML 形式的编程语言, 提供将 WSDL-Services 组合成控制流的能力。由于 BPEL 对于人工活动支持不好, 为此进一步扩展为 BPEL4People (WS-BPEL Extension for People), 从只能编排 Web 服务, 扩展为同时支持对 Web 服务和基于角色的人工活动进行编排。

业务流程管理 (BPM) 和工作流管理 (WFM)

业务流程管理 (Business Process Management BPM), 一般的定义为一套达成企业各种业务环节整合的全面管理模式。BPM 实现了人员、设备、桌面应用系统、企业级后台应用等内容的优化组合, 从而实现跨应用、跨部门、跨合作伙伴与客户的企业运作。

根据 WfMC 的定义, 工作流 (Work Flow) 为自动运作的业务过程部分或整体, 表现为参与者对文件、信息或任务按照规程采取行动, 并令其在参与者之间传递。简单地说, 工作流就是一系列相互衔接、自动进行的业务活动或任务。

工作流管理 (Workflow Management, WFM) 是人与电脑共同工作的自动化协调、控制和通讯, 在电脑化的业务过程上, 通过在网络上运行软件, 使所有命令的执行都处于受控状态。在工作流管理下, 工作量可以被监督, 分派工作到不同的用户达成平衡。

在本文中业务流程管理 (BPM), 是指基于 BPEL 标准的业务流程整合, 主要实现系统和系统之间的整合; 工作流 (WF) 是指人工活动的业务流程, 基于 XPDL 标准或者 BPEL4People 标准, 实现人机交互的整合, 目的是实现系统内部以及跨系统的流程审批。关于业务流程 (BPM), 当前有很多成熟的产品, 不做过多介绍, 本文以工作流管理 (WFM) 为基础, 基于《SOA 和 BC》的方法进行设计, 给出了工作流管理组件的设计模型。工作流和绩效、个人门户、即时沟通等紧密相关, 比如每个岗位流程节点汇总在一起, 在前端展示为个性化门户, 即时沟通平台实现人员之间实时沟通。

业务流程建模符号 (BPMN)

业务流程建模符号 (Business Process Modeling Notation, BPMN) 由 BPMI (The Business Process Management Initiative) 开发是一套标准叫业务流程建模符号, 于 2004 年 5 月对外发布了 BPMN 1.0 规范。BPMN 的主要目标是提供一些被所有业务用户容易理解的符号, 从创建流程轮廓的业务分析到这些流程的实现, 直到最终用户的管理监控。BPMN 也支持提供一个内部的模型可以生成可执行的 BPEL4WS。BPMN 定义了一个业务流程图 (Business Process Diagram), 该业务流程图基于一个流程图 (Flowcharting), 该流程图被设计用于创建业务流程操作的图形化模型。而一个业务流程模型 (Business Process Model), 指一个由的图形对象 (graphical objects) 组成的网状图, 图形对象包括活动 (activities) 和用于定义这些活动执行顺序的流程控制器 (flow controls)

在本文中业务流程管理（BPM），是指基于 BPEL 标准的业务流程整合，主要实现系统和系统之间的整合；工作流（WF）是指人工活动的业务流程，基于 XPD L 标准或者 BPEL4People 标准，主要实现人机之间交互的整合，目的是实现系统内部以及跨系统的审批。关于业务流程（BPM），当前有很多成熟的产品，不做过多介绍，本文以工作流管理（WFM）为主，进行设计，同时考虑到 BPMN 未来将会得到更多的使用，采用 BPMN 来进行画图。工作流是和绩效紧密相关，每个岗位流程节点汇总在一起，在前端展示为个性化门户，除了工作流的沟通之外还有消息平台实现人员之间的协同

工作流管理（WFM）组件设计

企业可以看作是实体对象，包括组织、人员、产品等在不同的环境和条件下的不断的运转的过程，实体对象和运转过程映射到信息系统中，分别对应着数据（可以用 ER 图描述）和业务流程（可以用流程图、业务逻辑和业务规则描述）。数据和业务流程能够全面反映实体对象及其运动的状态。在现实社会中，实体对象的运动体现为一系列活动，在信息系统中，活动表现为一个流程节点，实体对象通过一系列的业务活动直至最终完成任务，在信息系统中体现为数据状态的不断变化，直到数据最终完成。

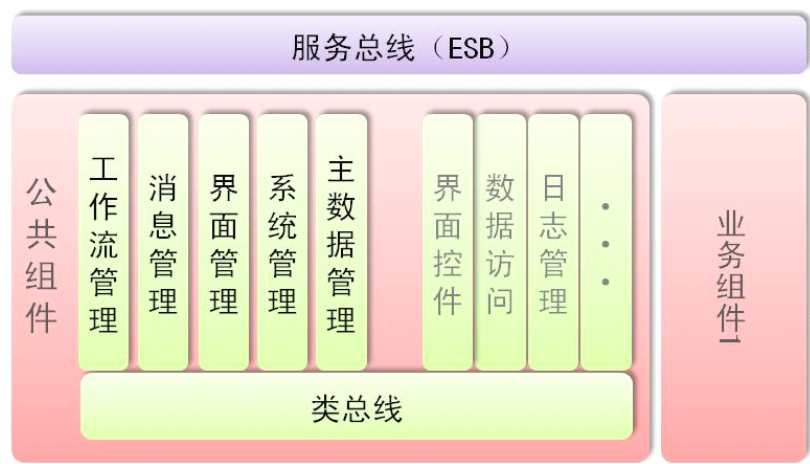
在《SOA 和 BC》一文中提到了关于基于 OSGi 的模块化的设计思路，工作流管理作为其中的一个公共组件的模块，如果是提升到企业级的公共服务平台中，则是独立的工作流管理业务组件，可以实现跨系统的工作流整合，以下结合《SOA 和 BC》的思路，进一步细化工作流管理模块（或工作流管理业务组件）的设计思路。

工作流组件的松耦合设计

传统的办公自动化或者协同办公系统，要实现基于工作流的流转，需要有两个基本的功能：工作流引擎和自定义表单，有了这两个基本功能就可以在一个系统中实现流程的流转。但是如果要实现整合企业所有的应用（不管是什么平台、哪个开发商），要将所有的业务全部整合到一个工作流平台中，就需要工作流组件提供一个松耦合的连接方式，将所有的应用整合在一块，保证现有的系统都能整合到统一的一个工作流中，从而实现统一企业的工作流，全面对企业流程进行监控。

结合《SOA 和 BC》的思路，将工作流组件作为一个独立的公共组件，为了更好的实现和其它业务组件以及公共组件内部的不同模块之间的松耦合，工作流组件对外以 Web 服务的方式对外提供接口，通过 ESB 和业务组件进行调用。同时为了保证性能，可以将工作流引擎内嵌到业务组件中，通过类总线（API）进行调用。这样既可以实现和内部业务组件之间的结合，也可以实现和应用外部的系统进行流程整合。从业务组件划分角度来看，工作流模块可以作为独立的业务组件，从方便管理角度来看，将其和其它的功能模块合并在一起，是公共组件的一个部分。公共组件模型如图 1 所示：

图 1. 公共组件模型— workflow 模块



[注]

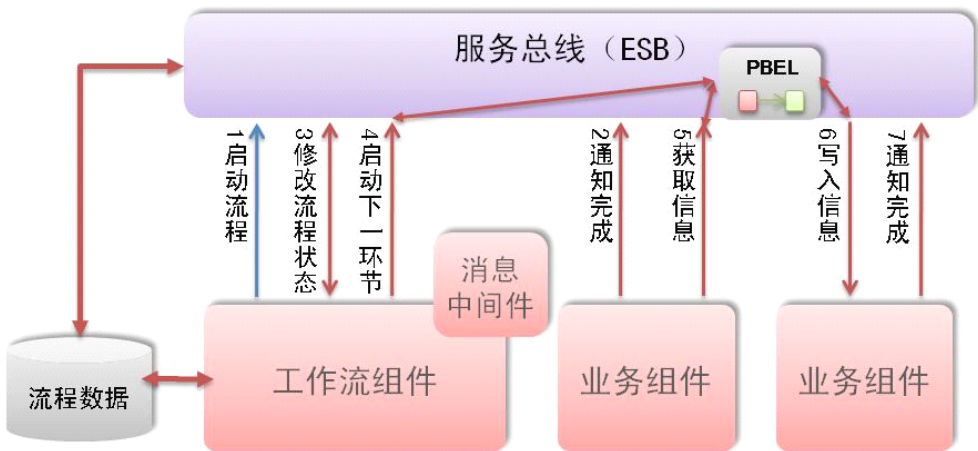
表单自定义功能是界面管理模块的重要功能之一。

本文中的 workflow 管理组件实际上是公共组件的一个部分。

workflow 组件和其他业务组件通过 Web 服务方式调用可以采用异步和同步传输两种模式。同步传输主要适用于两个系统必须同时提交的业务场景，采用同步传输需要等待另外一个系统的反馈信息，对提交 Web 服务的系统有性能的影响；异步传输的方式主要适用于可以异步提交 Web 服务的业务场景，保证了提交 Web 服务系统的性能，异步提交的方式需要考虑接受服务的系统出现宕机或者网络故障的情况下还可以准确无误的将 Web 服务传递到接收系统。异步传输一般通过消息中间件完成。

关于松耦合的 Web 服务的调用顺序在《SOA 和 BC》文中建议尽量采用触发的方式进行调用，基于这种调用方式，业务组件对外只提供查询或者写入服务，而不会直接通过写代码调用服务，实现业务组件的松耦合，这种调用方式同样适应于 workflow 组件中。比如客户注册流程，一开始实在 CRM 系统完成的，但是如果随着业务的发展，客户注册采用网上注册，原来 CRM 系统客户注册流程的任务启动就会发生变化，如果是采用的触发的方式，仅仅修改流程的编排即可（需要由业务流程管理（BPM）实现），而不需要修改 CRM 的程序。在 workflow 组件中，最好的一个实现方式就是由 workflow 组件进行流程的发起，如图 2-1 启动流程所示。

图 2. workflow 组件的松耦合调用



为了实现松耦合，业务组件和工作流组件之间不进行业务数据交互，传递的仅仅是任务信息。业务组件对外提供获取信息或者写入信息的 Web 服务，和普通的业务组件之间的 Web 服务没有什么区别，读取信息和写入信息是标准的业务服务，保证了为工作流使用的 Web 服务具有通用性。如果需要和工作流整合，仅提供一个特殊的 Web 服务“通知完成”将任务的完成状况或者任务的基本信息等传递到工作流组件，任务的基本信息主要是为了痕迹化管理，将修改的信息做一个记录，便于未来的审计。通过这种模式实现业务数据和流程数据分离，工作流组件和业务组件不进行业务数据的交互，简化了工作流整合的难度。工作流组件则提供启动流程、修改流程状态，启动下一环节以及保存任务基本信息等 Web 服务。

为了使流程平台具有良好的扩展性，如果工作流组件需要业务数据，比如需要根据业务数据进行判断业务流转，也是以 Web 服务的方式有业务组件提供一个标准的服务，通过 BPEL 实现，比如为了进行痕迹化，则需要对进行审批的内容进行保存，通过 BPEL 调用查询服务将数据保存到流程数据库中，其调用跟工作流引擎没有关系。实现跨系统的流程流转，也是通过 BPEL 的编排，通过调用业务 Web 服务实现。

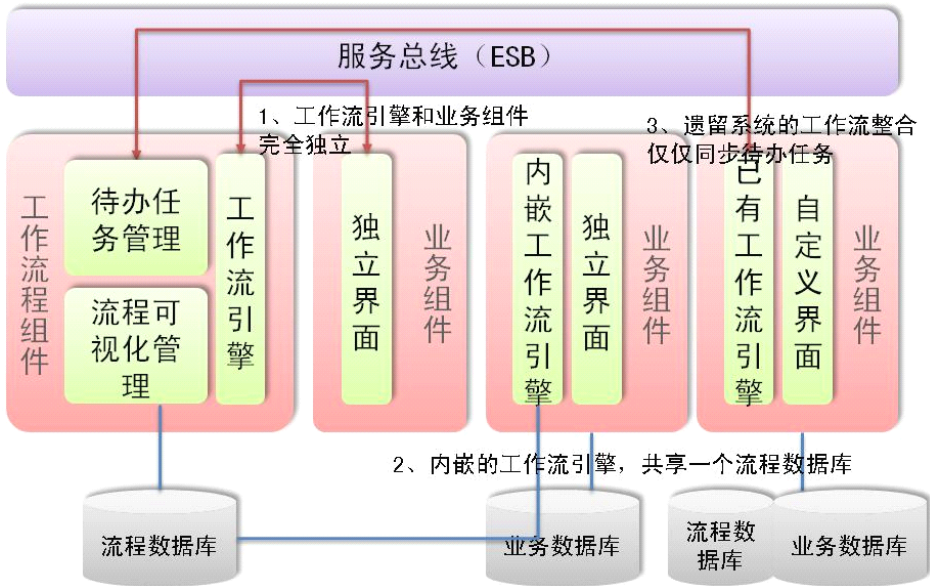
工作流组件组成及和业务组件关系

作为公共组件的工作流模块主要包含三个主要功能：工作流引擎、待办任务管理、工作流可视化管理。工作流引擎是基础模块，可以为所有的系统提供工作流引擎，实现工作流流转的逻辑控制；待办任务管理主要实现人机交互，提供一个统一管理的待办任务管理，整合公共的工作流引擎以及企业已经存在的工作流引擎（如 OA）以及普通业务系统的任务（如没有工作流引擎的 ERP），形成一个统一的待办任务管理；工作流可视化管理，主要用于工作流的可视化展示，用于除了用于工作流定义外，可以实现流程监控、业务绩效指标监控、流程导航等功能。

工作流管理组件和业务组件整合根据结合紧密程度可以分成三种：完全独立的公共工作流管理组件，工作流引擎和业务完全隔离开，流程任务通过 Web 服务方式交互；内嵌的工作流管理模块，业务组件通过 API 的方式调用内嵌在业务组件中的工作流引擎，工作流引擎

和公共的工作流引擎共享数据库；公共工作流管理组件和内嵌工作流模块组合，工作流引擎有自己的流程数据库，包含已有系统的工作流引擎有自己的流程数据库，工作流引擎和业务组件紧密集成，通过 ESB 以 Web 服务方式或者通过数据总线和公共工作流引擎进行数据交换，内嵌的工作流引擎负责任务的处理，公共的工作流组件集成待办任务。如图 3 所示：

图 3. 工作流模块内容及和业务组件关系



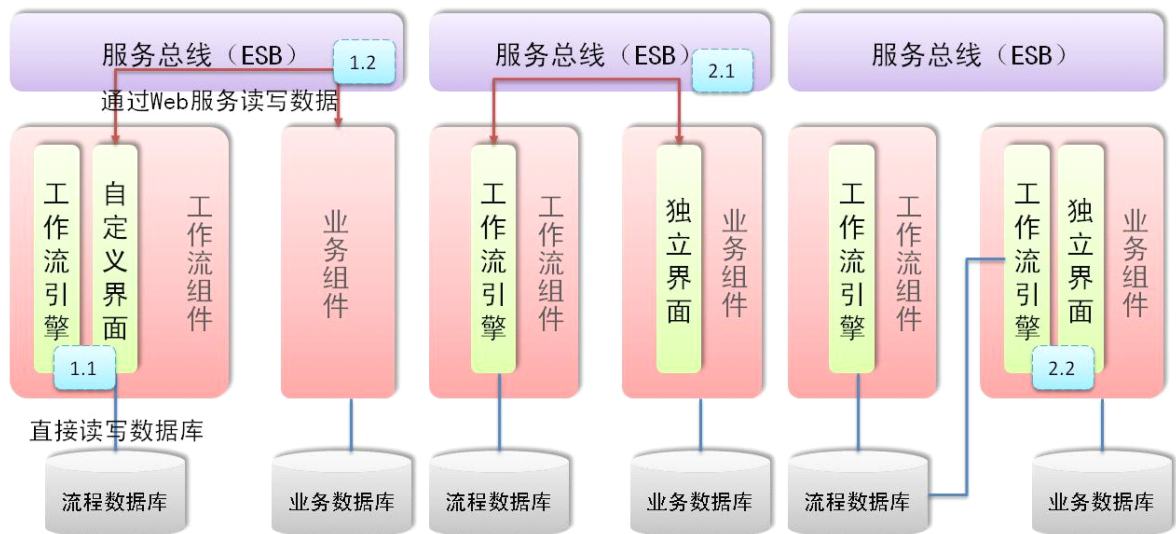
如上文所示，第一种方式是完全松耦合的；第二种方式易于开发，性能更好；第三种方式能够集成已有的各种工作流引擎，适合于集成遗留系统。

工作流的流程节点类型

根据工作流组件跟业务组件的融合程度，可以将工作流组件和业务组件的关系分成以下几种情况：

审批流程节点：直接和工作流组件集成，工作流组件中可以体现待办任务，并带有可以直接操作的表单（含带有附件的表单），典型的例子是协同办公中的公文流转、单据审批等功能。审批流程节点是和工作流组件的耦合性最大，完全依赖于工作流组件工作。审批流程节点可以进行转批、驳回等操作，详细记录流程相关的操作信息。

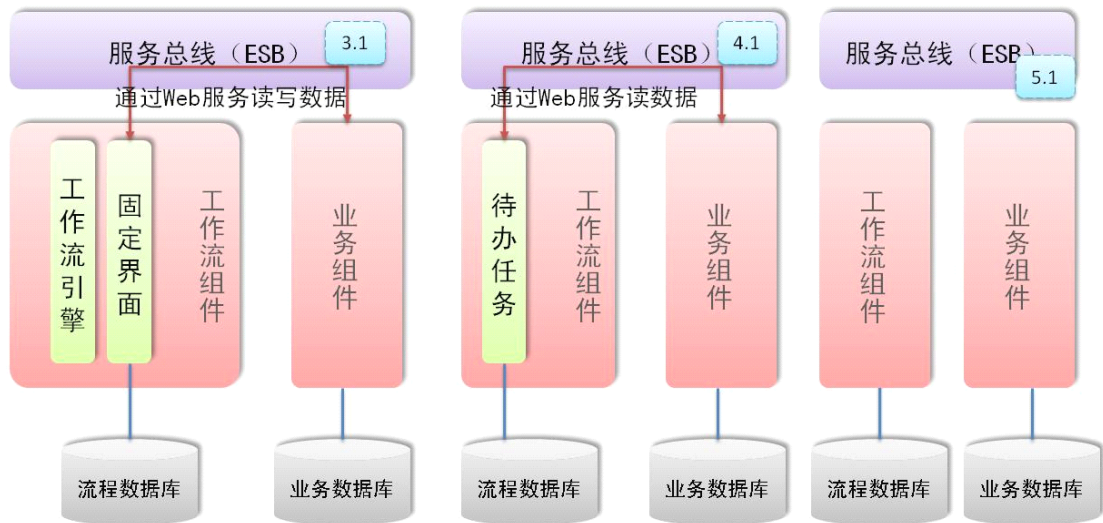
图 4. 审批流程节点带个性化表单的松耦合设计



根据表单是否独立，可以分成两种实现方式，采用自定义表单或独立开发表单。自定义表单中的数据源有两种，一种是由自定义表单直接读写数据库（如上图 1.1），一种是由自定义表单直接调用 Web 服务进行读写，业务组件提供 Web 服务，在自定义表单中进行展示（如上图 1.2）。采用自定义表单的方式，和工作流组件结合紧密，不便于实现松耦合，为了实现更好的松耦合，可以独立开发的表单，然后以 Web 服务的方式和工作流管理组件交互，使得表单和工作流引擎松耦合（如上图 2.1），或者将工作流管理作为一个公共模块，内嵌到业务组件中，通过类总线（API）工作流引擎和业务组件集成（如上图 2.2）。采用 2.1 方式能更加方便的组件松耦合的业务组件。

审核流程节点：和工作流组件应用集成，通过 Web 服务交互，表单显示格式固定，Web 服务可以携带审核信息，但是没有个性化的表单操作，仅仅可以完成审核确认功能。审核流程节点可以进行转批、驳回等操作，详细记录流程相关的操作信息。审核流程节点和业务组件之间是通过 web 服务交互的，因此是松耦合的，但是也限制了其应用，不便于直接操作业务组件的详细内容。由于其模式简单，适合于大规模使用，如订单审批确认、领导审批等功能，（如图 5—3.1 所示）。

图 5. 不带个性化表单的工作流



消息流程节点：和工作流组件应用集成，仅提供待办信息，可以通过 Web 服务调用，根据查询条件直接查询本节点需要处理的任务，然后进入相应的操作界面进行操作，操作完成之后，任务状态发生变化，待办任务完成。通知流程节点仅仅是一个消息通知，适合于大业务量的操作场景，可以实现简单的业务集成，提高系统易用性，简单记录操作过程。

（如图 5—4.1 所示）

审核流程节点和消息流程节点差异主要体现在前者工作流模块为主，和工作流紧密集成，但是只是简单业务操作，后者则以业务组件为主，工作流简单集成，但是却可以完成复杂的业务操作。和工作流引擎模块无关，仅和待办任务管理、流程可视化管理模块相关。

消息流程节点可以包含多种实现场景，由于和它的业务组件是松耦合的，可以对整个系统的所有应用进行整合，主要包括：

- 通知流程节点：根据数据的状态，由应用自动或者被动生成简单的任务信息，以通知的方式通知到工作流引擎，在待办任务中仅提供待办信息数量，具体操作还是在原来的系统完成。自动生成通知信息可以实时通知，但是需要原有的系统进行改造，被动通知则不需要原有的系统变化，仅仅是做一个数据查询的接口即可完成。这一点非常重要，特别是对于性能要求较高业务系统。
- 预警流程节点：有预警模块生成预警信息，将预警和流程结合起来，可以看到那些流程节点出现预警，并根据预警的级别，分成待办任务、确认完成和确认了解三种类型。待办任务需要启动新的流程，对预警信息进行处理（可以采用审批流程节点的处理）；审核完成需要落实预警信息的原因；确认了解则只需要了解即可。消息和预警的差异主要是后者属于异常的信息。

导航流程节点：和工作流简单集成，仅提供一个菜单导航的功能，提高系统的易用性，没有待办任务信息，简单记录操作过程，如操作人、时间、操作的功能，但是无法记录具体操作了什么内容。和工作流引擎模块无关，仅和流程可视化管理模块相关。

人工任务流程节点：和工作流无关，仅仅为了完整流程而做的一个描述。为了进一步提升系统的易用性，可以在所有流程节点上（包含人工流程节点）增加说明文字，作为企业业务流程指导书。和工作流引擎模块无关，仅和流程可视化管理模块相关。

通过以上分析，可以看到，为了搭建松耦合的工作流组件，可以采用通过服务总线（ESB）以 Web 服务方式或者通过类总线以 API 方式进行集成，搭建企业级的公共工作流组件。工作流组件和业务组件之间没有直接的业务数据交换，工作流组件相对更加独立；工作流组件包含了集成各种方式的工作流，特别是遗留系统的工作流引擎；工作流组件的适用范围更广，不仅仅是涉及流程审批，所有的业务系统都可以整合进来，建立一个企业的完整视图的工作流，搭建了一个企业级的技术架构平台。

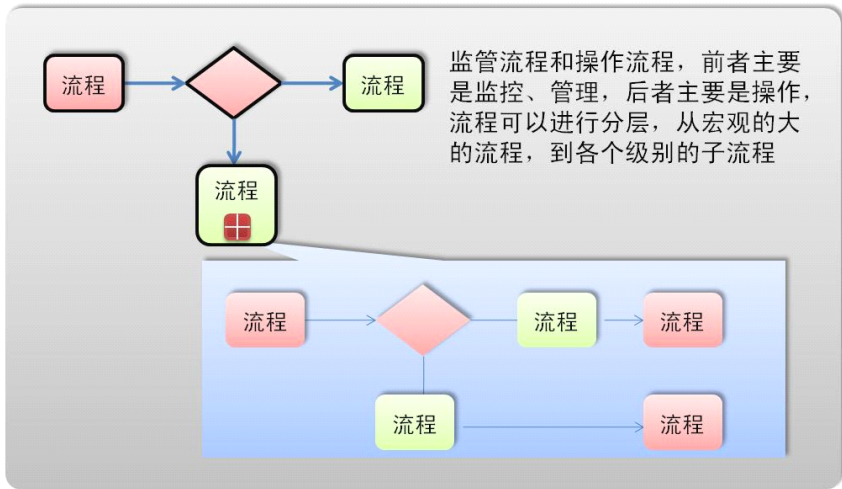
工作流的流程节点绩效和审计

工作流和企业绩效紧密相关，每个企业有一个总体的目标，目标的实现由一个个的工作流来支持，也就是说每个业务流程都是在为了实现企业的某个目标，因此工作流以及工作流的环节都可以对应着企业总体目标细化之后的细化目标，因此，可以通过流程节点 KPI 的监控实现对企业目标实现的监控，实现管理的可视化，从而的实现对企业运行的可视化监控，当某个环节出现问题，超出了预警值之后，就可以进行报警。流程节点主要包含的属性包含开始时间、结束时间、工作内容，操作内容、业务目标（可选，主要是大的流程和环节）待办任务、预警信息等。除了流程跟企业绩效关联的关系之外，工作流本身也是企业绩效的一个关键要素。通过工作流平台可以更加清晰的记录每一个流程节点的时间成本，为监控和优化每一个流程节点提供一个重要的参数。比如一个流程分成 10 个流程节点，通过对每个流程节点操作记录，可以方便的对每一个流程节点进行管理，特别是对于有时间要求的流程节点，可以起到一个很好的监控功能。

工作流管和痕迹化管理紧密相关，工作流的操作信息记录之后可以方便的实现工作的痕迹化管理，可以对操作本身以及操作内容痕迹化，工作流通过 ESB 可以对每次操作的数据统一保存一份到流程数据库中，这样就可以很方便的实现操作审计，谁在什么时候修改了什么数据等信息，保证系统的安全性。

因此流程的可视化管理可以进行分成两类，监管流程可视化和操作流程可视化，前者主要是监控、管理，后者主要是操作。为了便于理解，除了监控流程和操作流程的级别之外，流程可以进行分层，从宏观的大流程，到各个级别的子流程，不同类别不同层次的流程节点包含的内容和主要体现的信息是不一样的，即流程环节可以有多个层次的子流程，从而便于流程的查看和管理。。如图 6 所示。

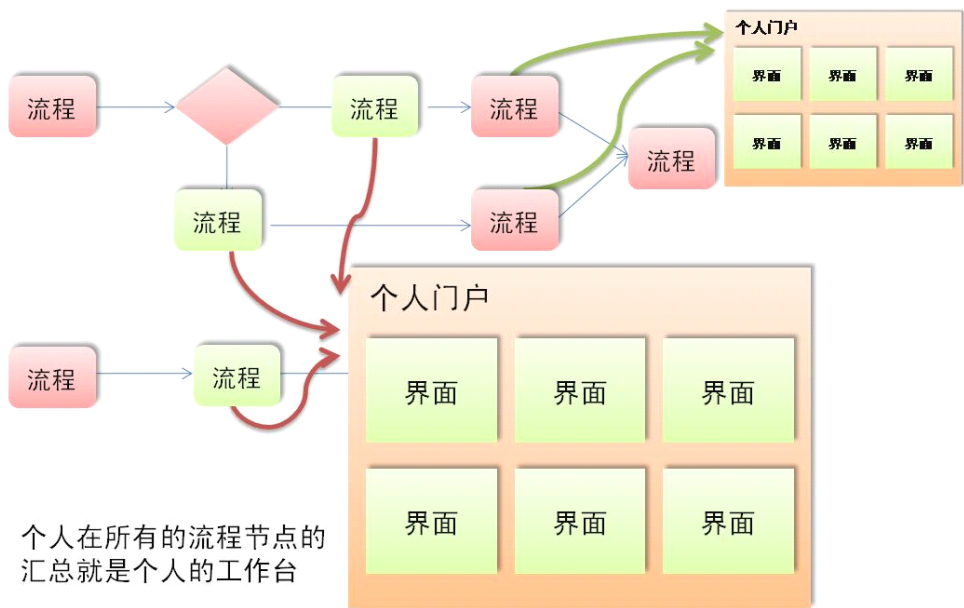
图 6. 监控流程及操作流程



工作流组件和个人门户

企业的业务运转是有若干工作流来完成，每个工作流则是由一个个的流程节点组成，每个流程节点对应这一个岗位，因此如果从具体的一个岗位来看，把这个岗位在所有流程节点需要作的工作汇总在一块，即形成这个岗位的在企业的的所有流程的一个切面，从信息门户角度看，即为一个岗位的个人门户。为了使得系统更加易用，需要以门户的方式，把个人所需要作的工作汇集到个人门户中，工作流和个人门户的关系如图 7 所示：

图 7. 工作流和个人门户



[注]个人门户功能是界面管理模块的重要功能之一。

通过工作流可以让员工从整体上对企业的业务流程有一个整体的认识，使管理者更加清楚的明白业务流程操作细节，使每一个员工对于自己的工作在整个流程节点的位置有一个更加直观的了解，通过个人门户，让每一个员工对于自己需要在整个系统中需要完成的工作有一个清晰的分类，这样通过工作流和个人门户，便于员工在岗位更换之后更好的适应新的岗位。

工作流组件和消息组件

在企业业务运作过程中，具体的业务流转在对于岗位形成一个流程的切面个人门户，主要以任务的方式体现，有人员到待办任务界面查看，除此之外还有一个消息的沟通，包含各种将任务信息通知到具体负责人和即时消息，可以以邮件、即时消息工具、短信等不同的方式通知到人员。业务相关的协同信息来源一般都可以归结到具体的一个流程上，但是除了业务信息之外，还有其它的一些信息，跟具体的流程无关，或者有些信息无法和流程进行关联，以消息的方式进行传播，因此需要建立一个消息处理模块进行处理各种消息，处理的消息的渠道有多种，即时消息、邮件、短信、电话等，需要一个统一的消息模块统一处理各种消息，包括消息订阅、消息发布等功能。

代办任务可以以消息的方式通过消息模块通知到相关人员，跟流程无关的内容可以直接通过消息模块进行相应的沟通。

通过以上分析，可以看到，为了搭建松耦合的工作流组件，可以采用通过服务总线

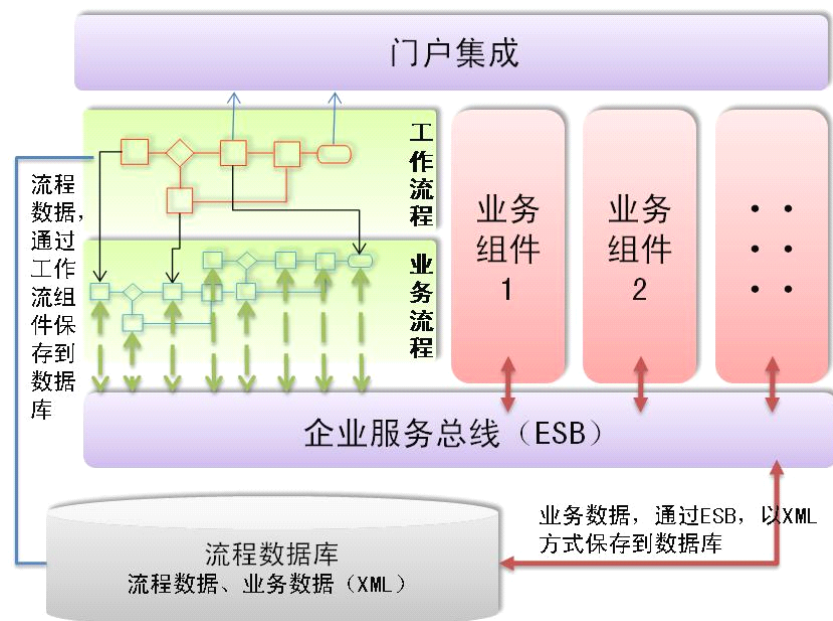
（ESB）以 Web 服务方式或者通过类总线以 API 方式进行集成，搭建企业级的公共工作流组件。工作流组件和业务组件之间没有直接的业务数据交换，工作流组件相对更加独立；工作流组件包含了集成各种方式的工作流，特别是遗留系统的工作流引擎；工作流组件的适用范围更广，不仅仅是涉及流程审批，所有的业务系统都可以整合进来，建立一个企业的完整视图的工作流。工作流和企业绩效、审计、个人门户和消息管理有着天然的联系，共通搭建了一个企业级的技术架构平台。

工作流管理（WFM）组件实现

工作流组件和工作流数据库

如前文所示，工作流组件在应用方面主要包含三个部分：工作流引擎、待办任务管理、工作流可视化管理，在数据库层面，主要包含两个部分：流程数据和业务操作记录，流程数据保存企业所有流程的流程定义、流程运行状态、待办任务以及流程可视化等流程本身的数据，业务数据保存流程过程中的业务数据。由于业务数据结构多样，为了保证对流程操作过程的详细记录，将所有通过工作流的业务操作信息全部直接以 XML 方式存放到流程数据库。如图 8 所示：

图 6. 流程整合平台



基于 IBM 产品的实现

本文提到的集成平台，基于 IBM 的产品共体系在实际搭建的时候需要包含应用服务器（产品：WAS）、流程整合服务器（产品：WPS，实现服务总线和流程编排）等，关于集成平台的详细描述，详见《SOA 和 DW》一文中“基于 IBM 产品体系的实现”的描述。除此之外，本文中涉及到的内容还包括：工作流引擎、基于 XML 的数据库、消息中间等。工作流引擎可以有多种实现方式，比如采用 IBM 的 WPS 或者 FileNet，或者浪潮 Loushang 工作流引擎等，本文以 FileNet 为例。流程数据库采用 IBM 的 DB2 V9。

数据库服务器（IBM DB2 Enterprise Server, DB2）的 XML 主持

IBM® DB2® V9 整合了关系型数据服务器的能力并进行了扩展，使得它能够管理关系数据和 XML 数据，包括对数据的存储，检索，共享，确认等。这使得用户可以拥有一个高性能，高可扩展性的平台来方便的管理关系和 XML 这两类的数据。pureXML 这个突破性的技术结合现有的数据管理技术可以带给用户所有期望从数据库系统中获得的功能特性。它提供高性能和高可靠性，并能够在极大的减小管理成本的同时，提高可用性和性能。DB2 V9 实现了对 XML 数据的索引技术。通过建立索引，给查询检索提供很好的性能。如果没有索引的存在，则任何查询都要遍历 XML 的整个树形结构，其效率就很低了。通过 DB2 V9，建立流程数据库，可以有效解决多种格式的业务数据痕迹化管理带来的数据保存的问题，为流程审计提供数据。

工作流服务器（FileNet）

FileNet P8 是 IBM 新一代的、统一的企业级内容和流程管理平台，它包含广泛的产品和服务，帮助用户在面向服务架构（SOA）的环境中构建，部署，运行和管理企业的内容和

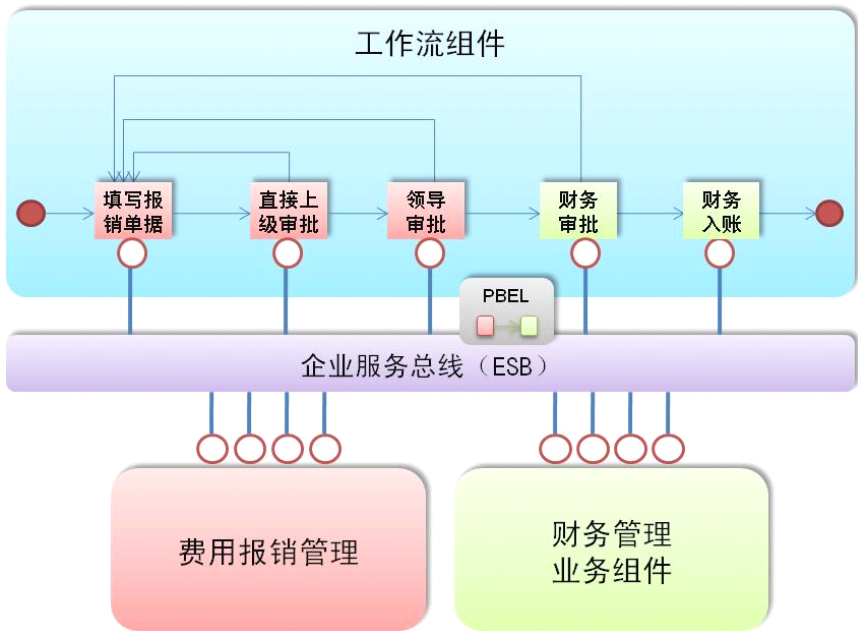
流程。本方案中主要是采用 FileNet 的流程管理 (FileNet Business Process Manager) 功能。流程管理包含流程配置控制台 (Process Configuration Console)，流程设计器 (Process Designer)，流程引擎 (Process Engine)，应用引擎 (Application Engine)等产品和应用。FileNet 的工作流具有分布式(distributed)，可获取性(availability)，可调控性(scalability)，安全，标准化的能力，可以有效的承担公共 workflow 引擎的角色。

工作流模块实现举例

以下以报销流程为例说明工作流组件的使用，报销单据凭证处理都是在财务系统完整的，但是财务系统一般不支持所有员工在财务系统中录入报销单据数据（如发票号码、金额等），同时由于财务软件是产品化的软件，很难进行大的改造，因此实现每个员工填写报销单据，并由上级领导审批报销单据，最后由财务进行审核，需要一个报销管理。报销管理单据的维护、审批处理，信息简单，可以直接用表单定制和工作流管理模块定制出来，包括填写报销单据、直接上级审批、领导审批等流程节点，就像一般的办公自动化系统实现的功能。

领导审批完成之后，启动一个业务流程（采用 BPEL 进行编排），将单据数据自动写入财务系统（只需要财务系统提供写入单据 Web 服务）。由于财务系统无法进行大的改造，在财务审批、财务入账环节，采用消息流程节点处理方式，由财务系统提供一个查询单据状态的接口，这样财务人员就可以在待办任务中看到报销单据的待办，财务人员通过单点登录，点击待办任务进入财务系统进行操作，报销单据的填报者则不需要进入财务系统，在报销系统中就可以看到报销单据的整个流程的状态，这样既可以实现整合所有流程，又对财务系统本身不会造成大的影响。

图 7. 报销流程举例



以上就实现 workflow 组件在数据库设计以及基于 IBM 产品体系如何实现做了简要说明，并以报销流程举例说明实现。

总结

本文通过对业务流程管理（BPM）和工作流（WF）的分析，给出了一个企业级的工作流组件设计，扩大了工作流适用范围，不再仅限于审批，而是扩展到所有的业务流程；提出了建立公共工作流引擎的思路，基于 SOA 的架构思想，整合企业流程；除了流程的整合，还对工作流和绩效、审计、个人门户和消息之间的关系做了简要说明。最后本文给出了工作流的实现思路以及基于 IBM 产品实现的产品支持。

