一、服务

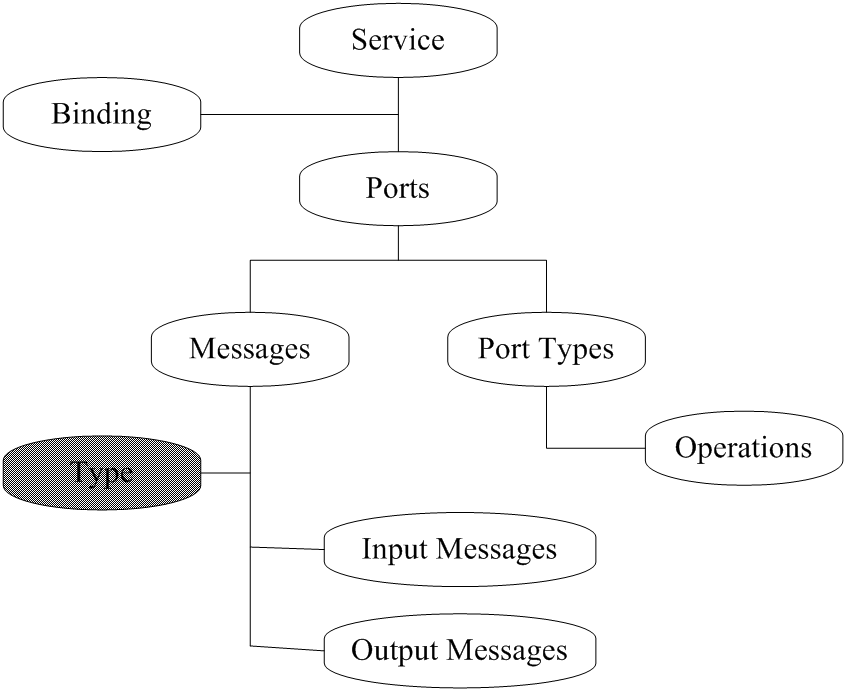
服务的定义：代表了一种基于至少一个服务供应商和一个服务消费者关系的交互,以达到某一种商业目标或解决方案的目的。

服务的生命周期：（1）咨询和策略计划（向第三方咨询公司咨询） （2）服务接触 （3）服务递送 （4）服务运作（service operation, monitoring, management） （5） 服务收费（长期性） （6）服务管理

服务自治系统:1垂直服务:从消费者的角度有许多服务可以被同时使用或独立使用,分为纯IT服务和IT服务

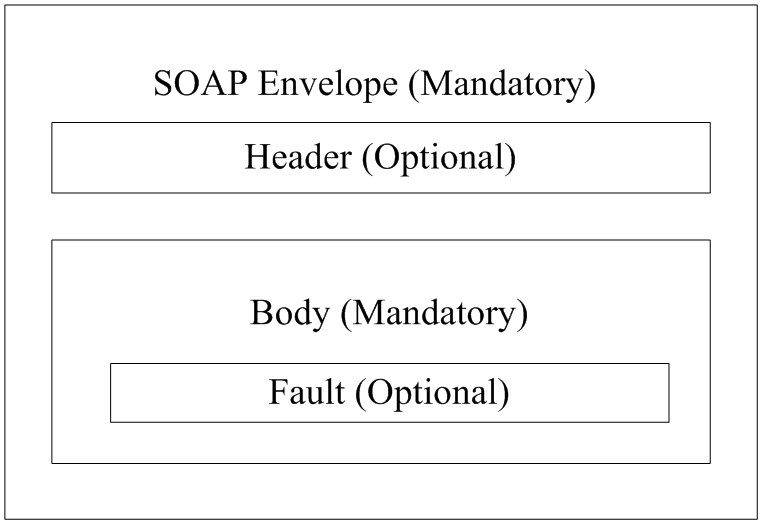
2 水平服务:垂直服务被构建成可重复利用的跨行业的共同服务,分为常见商务服务和IT服务

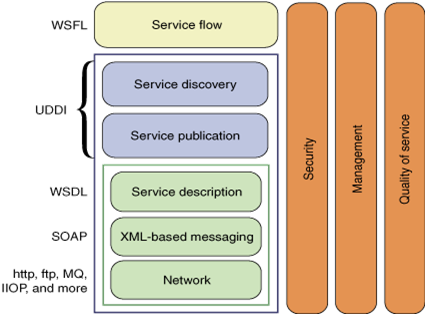
服务起因：1复杂的服务体系2灵活的服务体系3专业化和外包模型4计算环境的演变5IT专家和商业专家的距离6新的增值和创新性职能

二、服务的建模、发布、发现

WSDL：是一种基于XML，用来描述Web服务和说明Web服务之间如何通信的描述性语言。是一种XML格式,描述的网络服务作为一套端点操作,这组操作包含面向文件(异步)或面向过程(同步)的信息。它定义了一个web服务的公共接口,封装了服务的功能点和怎样调用他们.它同时定义了与其他web服务交互的信息格式和协议绑定。信息格式定义了怎样翻译信息中的数据格式,协议绑定定义了怎样把信息映射到一个具体的网络传输端口。

WSDL结构：WSDL定义了一系列端口（服务接入点），每一个端口有一系列的端口类型，端口类型和一个具体的网络地址相连，端口类型是相关操作的逻辑集合。操作是服务所提供的动作的抽象描述。

SOAP：一种简单，轻量级的协议，用来在Web服务之间交换结构化的信息。是基于XML的,并独立于任何操作系统,编程语言或分布式计算平台，需要绑定到现有的互联网协议,如的HTTP,SMTP 支持的两种交互模式:远程过程调用和面向文件的模式。

 SOAP结构：有四个元素。外层封装（包括SOAP的根元素和XML的文档头），SOAP报头（包括控制信息，如作者等），主体部分（包括请求回应信息），以及错误信息。其中报头和错误信息是可选的。

UDDI：Universal Description, Discovery, and Integration是一种通过查询海量数据来发布和定位web服务的中继服务。UDDI的定义一种机制来存储用XML信息注册的Web服务的描述。包括四个关键元素:商业实体 商业服务 邦定模版 技术模型 使用XML Schema描述其结构。

右图是UDDI的协议栈：

WSRF（web service resource framework）是一种基于XML的描述资源的方法.定义了几个使用web服务管理和接入稳定资源的系统 包括四组规格:1WS资源性质 2资源生命期 3基本错误 4服务组

Stateful Service：为消费者提供各种个性化、持久的服务,综合服务需要协作;支持更多商务交易，提供更多个性化服务，需要更多代码和过程资源 重量级

Stateless Service：不需要捕获或保持状态 只读web服务 规模小 容错性好 轻量级

BPEL：基于XML的业务流程执行语言。定义多种业务的协调互动,以实现一个共同的业务目标.精确地界定基本服务行为的跨企业的商业协议。关注描述一个商务过程的行为,这个过程基于过程和合作伙伴的互动。建立商务过程模型的两种方式1可执行的过程模型 为商务交互中参与者交互行为建模2抽象模型 显式描述没有显示其内在行为的参与者的行为 用BPEL建立商务过程的两个阶段:创建服务描述 创建商务过程

BPEL的9个关键元素1Partners定义了商务过程中的关系2partner link types 定义了两个web服务的关系3partner links 在商务过程中交互的伙伴服务4business partners 两个商务伙伴之间的关系5endpoint references 动态绑定服务端口数据的机制6activities 如接受 回应 调用7data handling 商务过程的状态8Correlation关联9Scope范围每个活动的上下文

WSDL是对单个服务的建模； WSDL+BPEL组合成复杂的服务。都关注服务的静态信息，WSDL是一个抽象接口，描述了信息的格式和协议，以及服务的地址，而BPEL则是关注众多服务之间的关系。

服务的发布：集中式发布：UDDI；分布式发布：WSIL。

UDDI :1UDDI的可被视为一个传统的黄页目录,分类,并组织出版的Web服务的各种组织形式;2 UDDI的还提供相应的信息服务提供商,UDDI注册提供了一个SOAP接口,使SOAP消息为基础的服务发布，被接入控制划分为公共和私有UDDI;3集中式服务注册)

WSIL:1 WSIL通过常规的WS发现部署调用Web服务,而不需要复杂的服务注册基础设施2Web服务作为一个普通的XML文件发布到Web服务器上.Web服务检查语言( WSIL )是用来聚合现有的服务描述文件参考.参考指针可以用来连接一个发表在UDDI注册的服务或连接到另一个文件WSIL,以形成WSIL链3分布式服务注册)

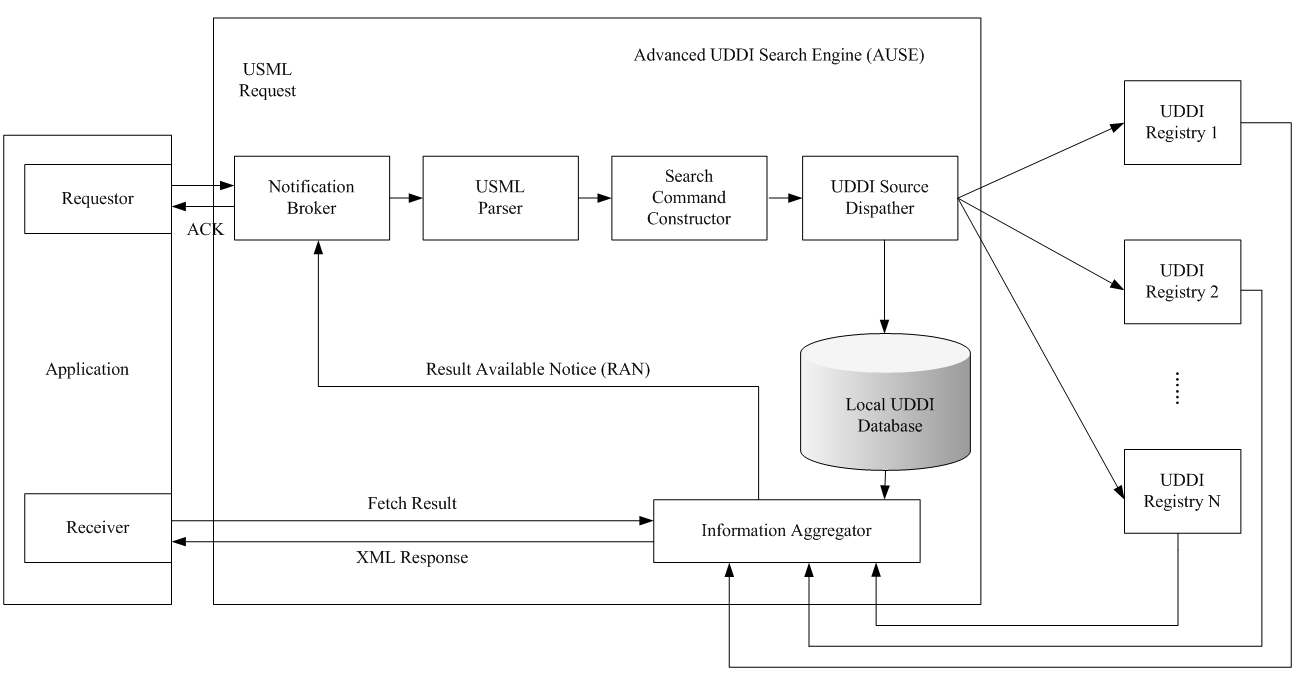
简单的服务发现：

1. UDDI：查询结果太多 必须知道查询目录（如by business, by service, by service type） 必须人工写大量的查询代码，尤其是复杂的查询批准时 标准的UDDI查询只支持在UDDI库中的查询 一个UDDI查询只能查询一种分类，效率低

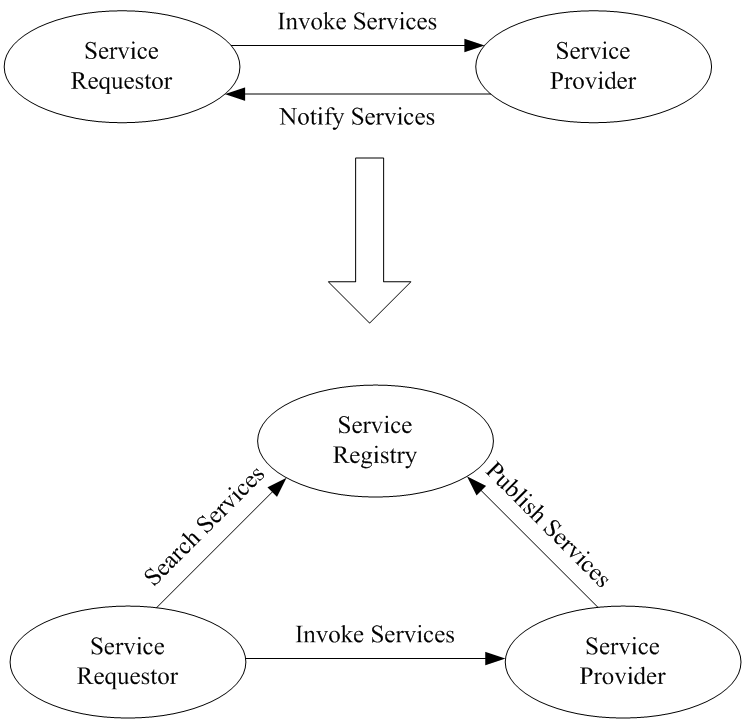
2. WSIL：缺少将查询WSIL链过程中发现的结果记录下来的能力

USML：UDDI Search Markup Language 一种基于XML的综合的描述语言，旨在为了形式化UDDI搜索查询以及提供一种在UDDI库中的复杂的UDDI搜索查询。 一般能多查询，并将查询过程中的结果记录下来

AUSE：USML-Based Advanced UDDI Search Engine 目标是能够自动的解释输入的USML查询请求，并将这些查询请求投递到相应的UDDI库中，记录不同UDDI库中查询的结果，并以USML的形式返回给查询方



DSDF：WSIL-oriented Dynamic Search Discovery Framework 提供了一种机制，该机制能够自动从WSIL链中查询，记录从每一个WSIL文档中的结果，并实时的反馈给查询者。

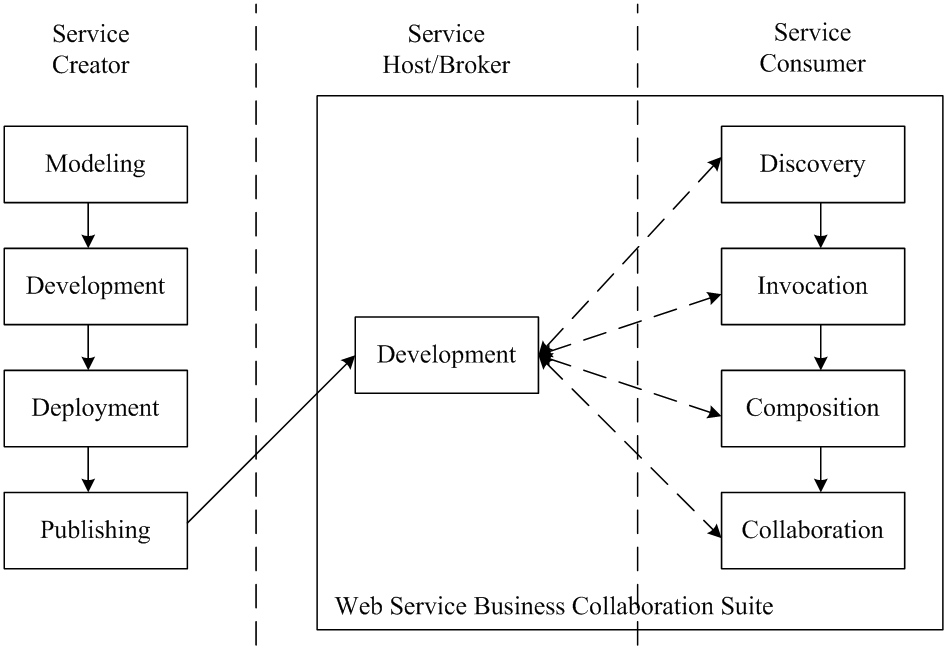
Agent-base Advanced Web Services discovery：代理商机制

三、 SOA

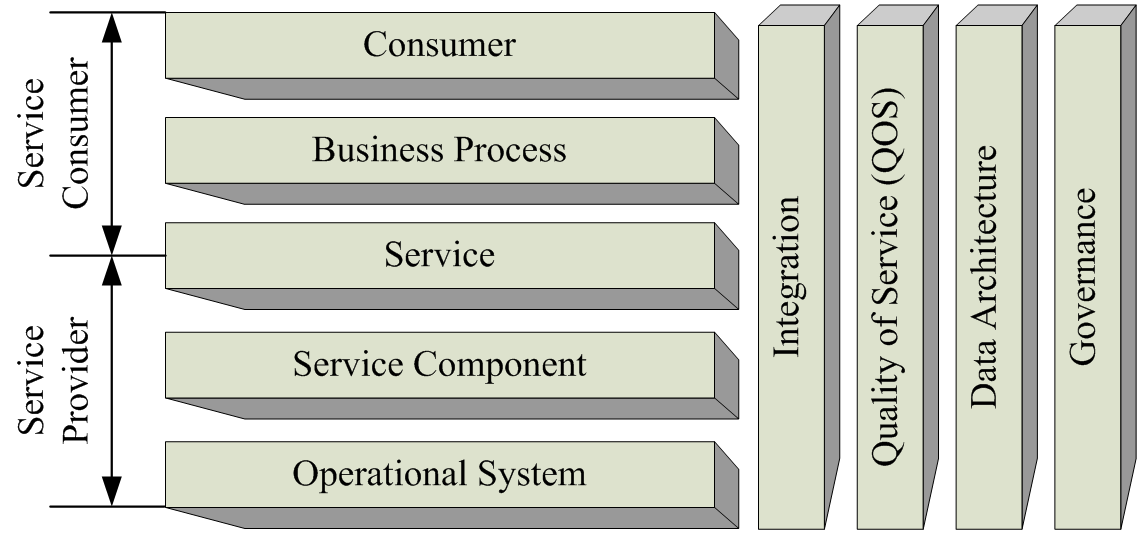
SOA：是一个概念级别上的架构模型，是建立在现有服务之上来构建推动业务流程的应用的商务IT一体化的解决方案.

“二级”到“三级”的转化：传统的服务模式是一种端对端的服务递送模式，服务的提供者通知请求者有可用的服务，服务请求者向提供者调用服务。而SOA框架下的服务模式是三角的，包括三种角色：服务的提供者、服务的请求者、服务注册处。1. 服务提供者向服务注册表中发布服务2. 服务请求者在服务注册表中搜索服务3. 服务请求者调用发现的服务。三个功能:发布,搜索,调用;三个主要的协议:WSDL, UDDI, SOAP。

四个层次上的SOA：1. programming level: SOA用来指导低级别的IT技术,包括SOAP,二进制SOAP传输,SCA 2. middleware level: SOA用来指导普遍产品和开源软件的设计与开发3. process level, SOA用来指导业务流程的集成与管理以及事件驱动架构的设计4. enterprise level SOA用作指导企业的组件化以及支持高级的传输咨询。在每个级别,SOA用top-down的方法指导大的单元分割成小的以服务为中心的单元,用bottom-up的方法组织小的单元集成用以提供新的服务。



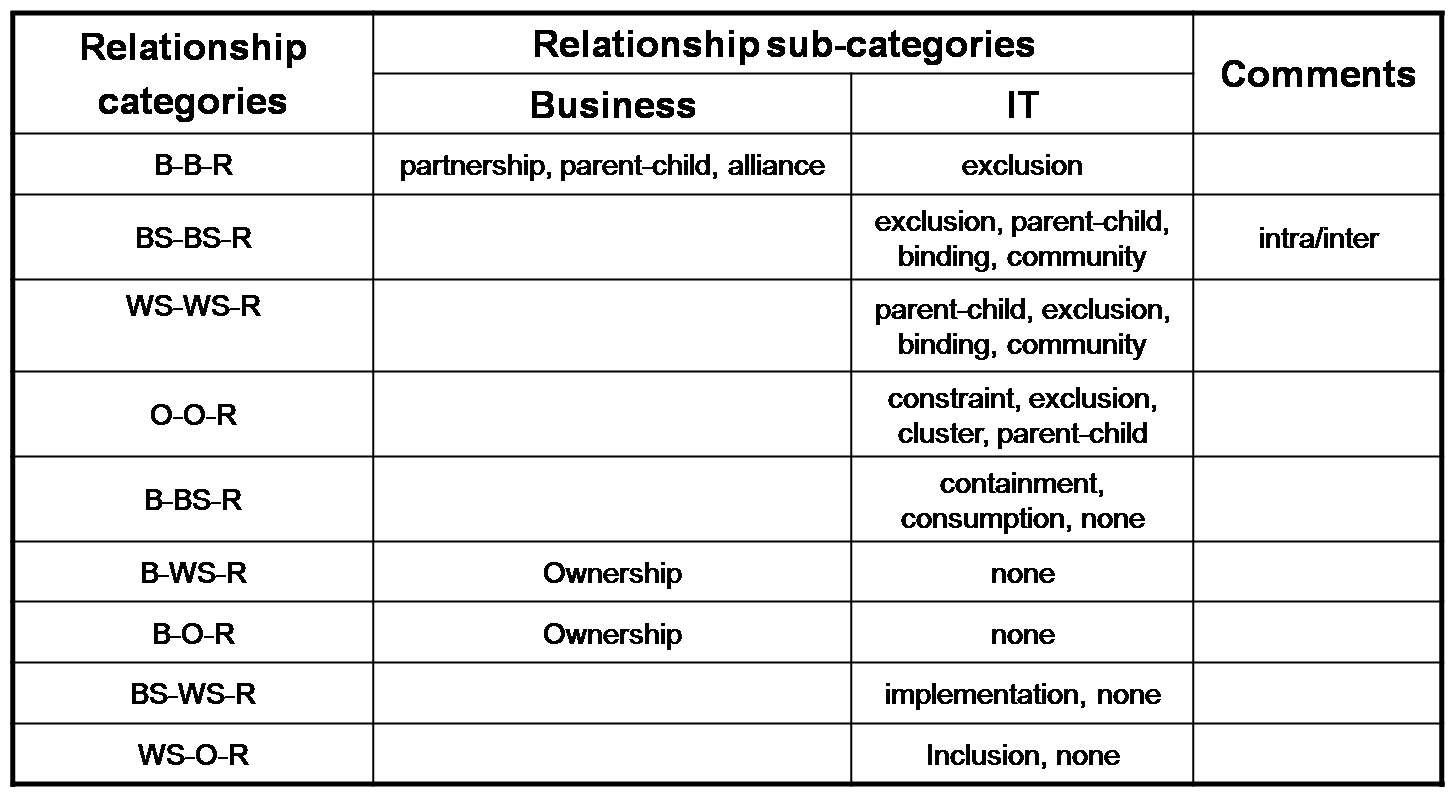
SOA的生命周期：服务的生命周期是从一开始的构想一直到服务不再使用，整个过程SOA都参与了。主要包括服务创建(建模,开发,部署（根据服务的类型部署到相应的服务器上）,发布),服务消费(发现,调用,组合,协作（众多组合在一起的服务之间通过信息交流以达到协作的上的）)以及服务的监控和管理。

SOA-RA：SOA参考模型 把一个SOA系统划分为一个拥有五个水平层（实现功能需求）和四个垂直层（提供了便利与可行系统支持）的二维体系架构。四个垂直层:1integration layer:提供服务提供与请求者之间的协调,路由,服务要求的传输的关键支持2QoS layer:在多个方面提供了解决方案级别(不是服务级别)的QoS管理3data architecture layer:提供了与特殊领域数据架构集成的统一化的表示框架来推动价值链集成4governance layer:提供了能够保证SOA解决方案架构设计的指导。 五个水平层1. operational system layer:包含了具有特定功能的现有的解决方案包,客户应用以及一流系统 2. service component layer:提供了在服务层定义了的服务实现3. service layer:扩展SOA的三角模型为一个允许及推动服务注册,拆分,发现,绑定,接口聚合以及生命周期管理的综合性的逻辑层4. business process layer:管理所有被用作服务组合与拆分的业务逻辑5. consumer layer:为企业流程层,服务层等层说明快速建立服务用户接口以满足客户需求.

SOA－RA的意义:1它是一个关于如何以一系列的逻辑层的结构来构建一个SOA方案的抽象；2它是一个低耦合的体系结构.每一层都是严格和它上面各层分割开来的；3它是一个企业架构模板,指导如何在企业层创建SOA方案

四、 服务关系建模

9大关系：B-B-R, B-BS-R, B-WS-R,B-O-R, BS-BS-R, BS-WS-R,WS-WS-R, WS-O-R, O-O-R



partnership:合作关系

parent-child:从属关系

alliance:联盟关系

exclusion:竞争关系

ownership:拥有关系

binding:绑定关系

community:提供相同或相似功能

cluster:

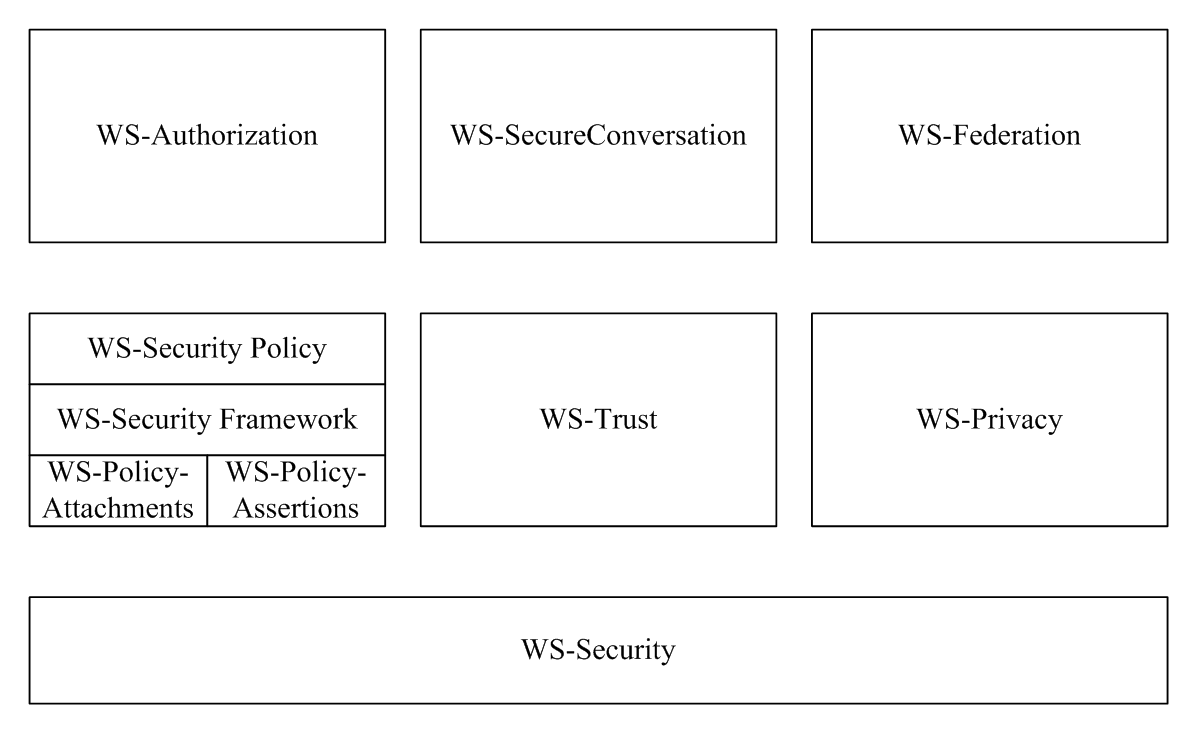
inclusion:包含关系

consumption:消费关系

none:没有关系

注：关系包括各自己产品的关系以及和其它公司的关系。

五、 QoS in SOA

Web服务的非功能特性（QoS）包括：安全性;事务性;消息传递的可靠性;资源的生命周期的管理

(1) 安全性提供了一系列的机制来帮助Web服务的开发者保护SOAP消息的交换.包括认证和用户授权;消息的完整性;消息的加密.安全性 分为6个子成分(WS-Security road map如右图): WS安全策略,WS信赖中心,WS-Privacy,WS认证中心,WS安全对话中心和WS联盟.

(2) 事务性 WS-TM定义了三个传输协议:1WS-Coordination(调和):负责开始并且协调若干个Web服务合作;它定义了:协调者服务和合作的上下文.2 WS-Atomic Transaction(WSAT,原子传输):主要对服务的提交和通知进行控制.3异步服务访问控制.

(3) 消息传递的可靠性: 三种途径来保证信息的可靠性:顺利的传递(通过反馈机制);不能被复制(检查ID把相同ID消息删除);有序的传递(通过序列化数字)

(4) 资源的生命周期的管理:定义当前交换的资源的生命周期以及到期后如何销毁,在生命周期内对资源进行监管

六、 需求驱动的服务组合

为了减少搜索空间,一个两步骤过程被引入：商务需求应该以一致的格式来表示,并且能够被用于自动产生搜索脚本来驱动服务发现引擎.在缩小范围之后,下一步工作就是探 索一个合适的方法来把候选服务组装到商务流以满足商务需求.可以采取全局优化算法.

七、 服务价值链的协作?

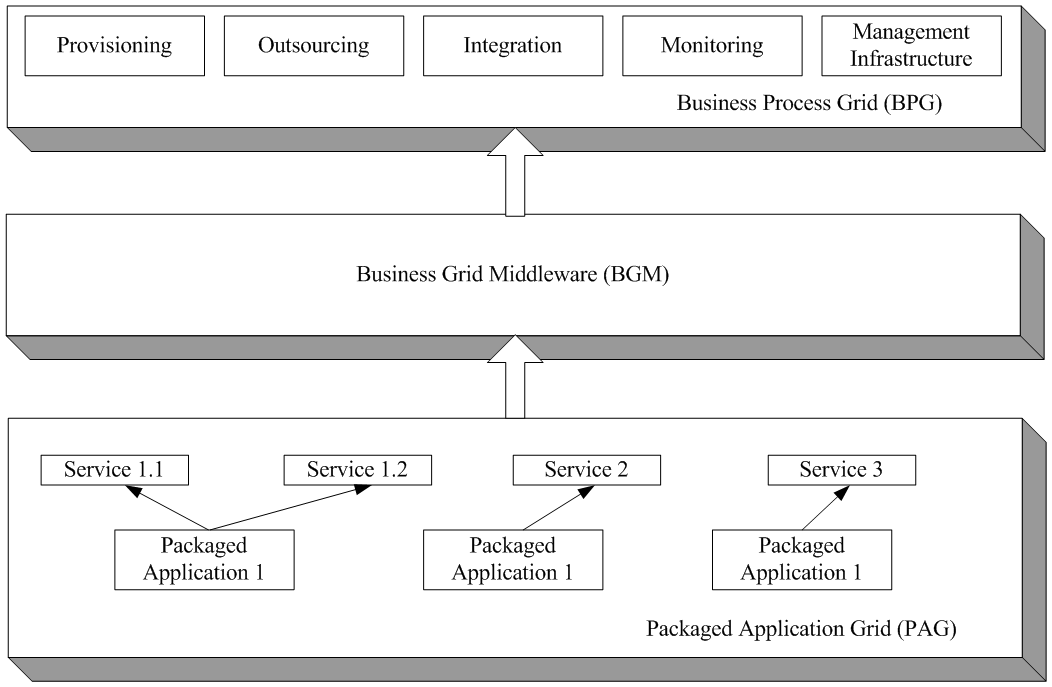
九、 商务网格

网格：为了使得当计算负载很高时能够使网络中连接上来的几点可以使用的资源实现最大化,每一个网格,也就相当于一个应用程序,来执行不同方法来协调多个计算机上的资源(计算能力、存储能力等等)

OSGA定义Open Grid Services Architecture OGSA的)是一种互动和分布式计算架构,面向网格服务,用WSRF表示

目的是确保在异构环境中的互操作性,从而使不同类型的系统可以交流和共享资源,它采用了Web服务技术,推动传统的网格计算朝面向服务的架构发展。OGSA最核心的思想是利用Web服务的概念来描述网格服务。OGSA的主要组成部分有服务的制造，注册，发现，生命周期，查询，通知以及可靠的调用。

逻辑的网络基础



Provides an IT-level infrastructure to provide support facilities

Organizes and leverages existing application services and wraps them into Grid services to support new business processes

Web服务和网格服务的工作流模型

十、 企业模型

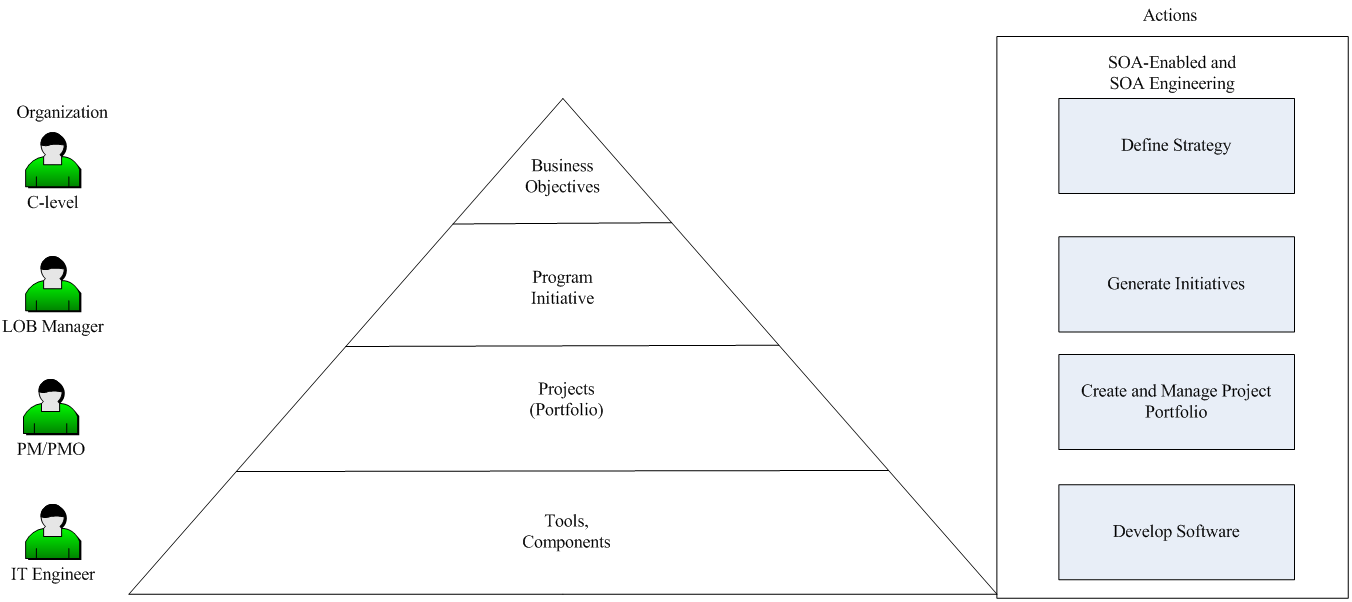
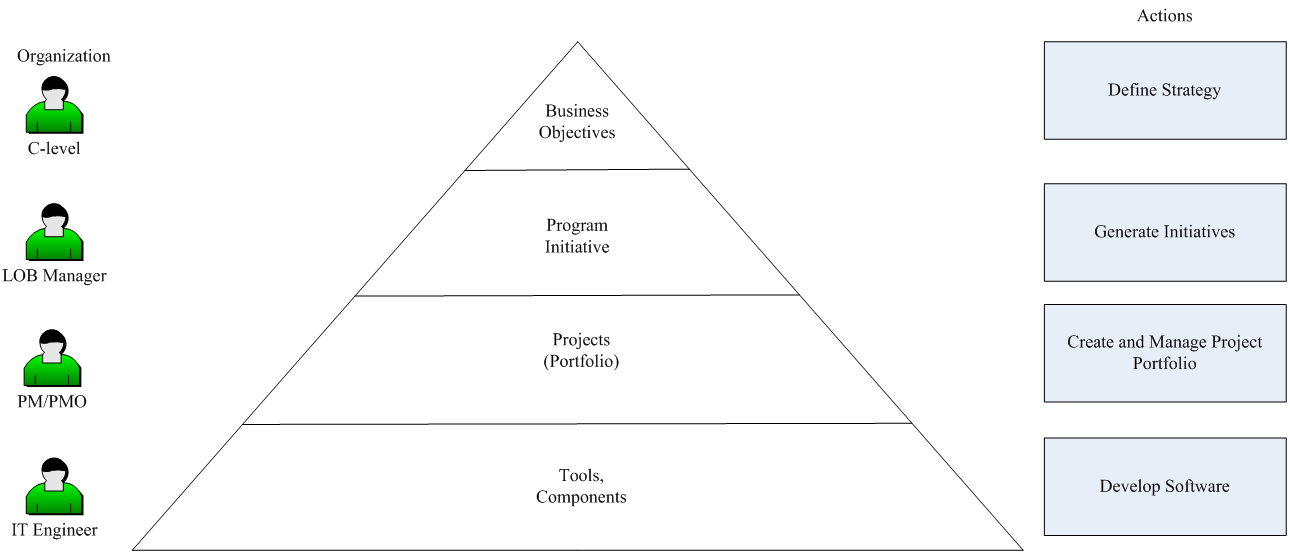
企业建模的方法：

1. 平衡打分表（Balanced Scorecard: BSC）：一个简单的二维表格,可以对每个项目进行综合评判,既可以用于评价新项目还可以对已存在的项目进行监督管理.超越了传统的以财务量为主业绩评价模式。缺点是:静态的表格,不能体现之间的关系，只能反映企业内部的共性，不能反映其特性。
2. 策略表（Strategy Map）：是BSC的扩充,与BSC相比，它的每一层下可以分成很多元素，另外策略表是动态的。通过每层之间的关系可以反映高层次组件的KPI（关键绩效指标）的由来，使得企业了解到自己的KPI的分布情况。
3. 业务组件模型（CBM）components business modeling一个二维表格,但是行列划分主要是行业指标,三个横行是商业生命周期的三个阶段（策略，控制，执行）.优点是:比BSC更为灵活每一层的items个数不一定，企业可以自定义组件，组件级别的规划，可以更方便地规划企业的功能。

（产生CBMC的五个步骤:形成一般的CBM;分析企业的核心竞争力;对每一个组件进行财务分析;将当前活动转化为服务组件;实现商业服务）

1. 企业架构（EA）：对整个企业进行分析.企业架构方法学：也是一个二维结构:横向的是EA的各个阶段;纵向的是EA涉及的重要领域.对于EA设计的每一个阶段都是一个迭代过程:每一过程可能对几个领域讨论多次直到都符合要求为止.EA的发展趋势是面向服务的EA)

十一、 企业绩效管理



EPM framework SOA-enabled EPM framework

EPM(企业绩效管理):集成项目计划,策略,资源配置和体系架构管理来为企业获得最大的盈利.EPM框架:从上到下:工作量,资本,人员越来越多.本图中的EPM对若干个PM进行综合讨论,因为PM之间都有关系,不能武断分开,而对每一个PM来讲,它们都需要被放在这个整体中讨论.而传统的PM只涉及到最底的两层,因为它讨论的只是单独的一个项目.

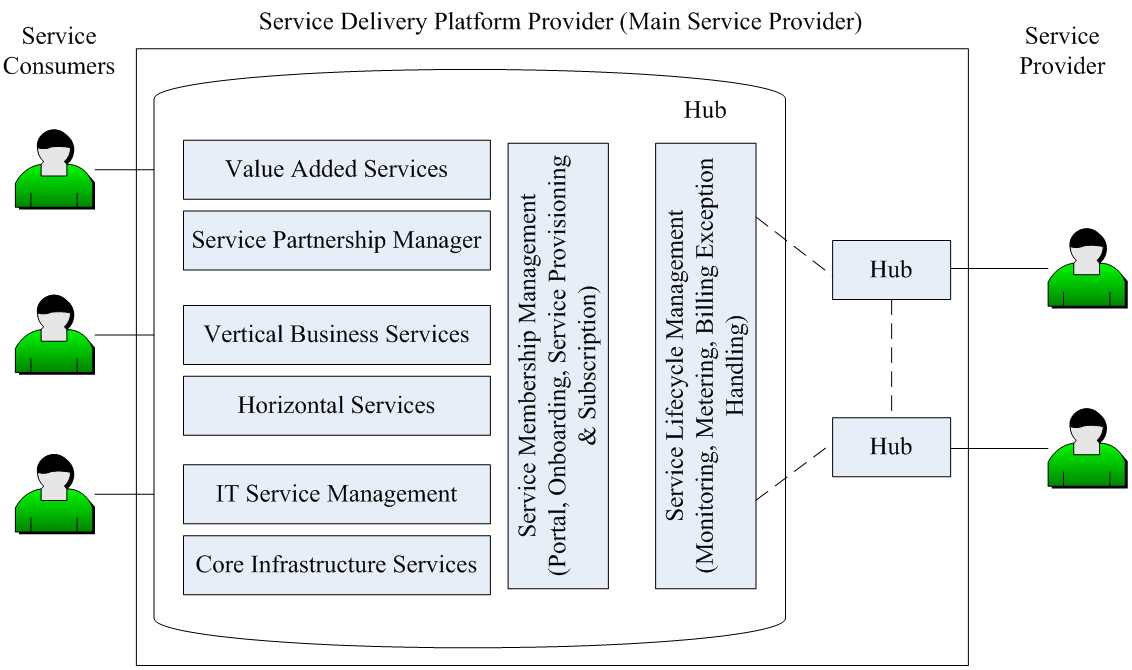
面向服务的EPM(WS-EPM)：SOA支持下的EPM框架:与EPM基本一样,不同的是右侧的“actions”是在SOA的支持下建模的(如使用BPEL对商业流程进行建模)

WS-EPM中的共享服务(WS-EPM资源(主要包括两类:物理资源和非物理资源 ,可分为三类:商业组件, 商业策略,非功能性要求)和WS-EPM中的三类实体)

十二、 服务的递送

服务送达的介绍:服务送达的目标是服务提供者按照合同约定的方式把服务送达给消费者,与此同时保护服务提供者的利益.服务送达的生命周期包括:准备阶段;创建阶段;操作阶段;服务送达主要通过的是Internet的常用的协议将服务送达到客户端，如HTTP等。

端对端的服务送达和方法：两种服务分发的形式:1. 作为服务的软件:把软件放入到web中,使得用户可以调用它,而不需购买它.主要包括两个方面:把软件放入到web中;把软件转换为服务.2. 作为软件的服务:核心是怎样吧领域知识标准化,形式化.

基于SOA的服务送达平台(如web2.0)1. 服务送达平台的分层 (水平层:核心构造服务层,IT服务管理层,水平服务层,竖直服务层,服务合作关系层,增值服务层.竖直层:服务成员的权限管理层(负责控制入口的访问,商业实体的登陆,服务准备和订制),服务生命周期管理层;服务合作关系管理:负责组合服务来满足消费者的需求. 服务生命周期管理:竖直服务:满足某个业务的特定需求水平服务:提供公共服务

十三、软件即服务VS服务即软件

软件即服务是把软件看成服务来对待，通过Internet提供软件的模式，将存在的软件封装成一个服务放在网上，将传统的用户购买软件模式改成了用户租用Web服务的形式，用户无需对软件进行维护，服务的提供者对软件进行全权的管理和维护。

服务即软件是把某一领域的专业知识，经验，过程封装起来，这些优质的服务我们将其作为软件组件卖给用户方，用户将这些软件组件按需组合起来，形成一个适合自己企业的专业的服务。

十四、 面向服务

面向服务的特点：

1. 松散耦合
   * 数据抽象、OO、基于组件的软件工程（CBSE）
2. 服务发布、注册和发现
   * 发布/订购模式
   * 工作流、协作模型、应用模板
3. 涵盖软件需求、设计、实现、验证与确认、维护及演化的整个生命周期
4. 涉包括服务、工作流、服务协同、应用在内的各类软件资产
5. 具有较高抽象层次
6. 强调开放性和标准化

OO泛型和SO泛型：

|  |  |  |
| --- | --- | --- |
| 特点 | OO泛型 | SO泛型 |
| 方法论 | 从构造函数——通过类或模型——到系统设计 | 从系统模型到服务模块，从服务抽象定义到服务实现绑定。通过搜索获得可用的服务实现 |
| 协同 | 应用程序开发委托给一个独立的小组，它对整个开发进程负责。该小组需要知道应用程序逻辑并具备编程知识 | 开发工作划分为三个不同部分：应用程序开发者，服务提供者和服务代理。应用程序开发者需要理解应用程序逻辑但是不需要具备编程知识，服务提供者可以编程但是不需要理解应用程序逻辑 |
| 抽象 | 类层次的抽象 | 将应用程序开发和服务开发分离。服务比类的抽象层次更高。应用程序在服务组装模型上进行抽象 |
| 代码重用 | 通过继承的方式，实现类的重用。代码在应用程序或平台内获得重用 | 借助于独立的服务代理，可实现跨越不同平台、不同应用的服务共享 |
| 动态绑定 | 在运行时分配变量和函数的地址空间 | 可以通过服务目录动态查找需要的服务，激活并使用远程服务 |
| 重组 | 多在设计时决定导入的组件 | 可以动态改变应用系统中服务的组合关系，以及服务定义与服务实现之间的绑定关系，即实现动态地添加、修改、删除各个服务节点 |
| 组件通讯和接口 | 与平台和语言有关，例如C++程序难以直接和Java程序通信 | 与平台和语言无关。组件间通过标准协议通信，如XML，WSDL和SOAP |
| 系统维护 | 用户需要升级软件 | 通过互联网升级系统，因为服务多运行在远程服务器上，用户通过互联网进行访问。维护对用户透明 |
| 可靠性 | 在设计时决定可靠性的方法 | 对于服务提供者，每个服务相对简单，更加可靠。对于应用程序存在多个满足同一需求的服务，可用过将故障服务的节点断开并重新绑定到备选服务节点上，获得不间断的应用系统 |
| 软件拥有 | 软件作为产品销售，为用户所拥有 | 软件存在并执行于独立的服务提供商的电脑上，用户按照每次对服务使用付费，而不是按照软件产品付费 |

面向服务的分析与设计：

基本原则：重用（服务、流程、应用模板、测试用例等）

为变化而设计

业务流程相对稳定，用工作流进行抽象

具体业务实现可变性较高，采用服务封装，并在运行时动态绑定

面向服务的测试的特点：

* 基于规约的测试
  + 服务用户可能没看到或看不到源码
  + 服务提供者虽然看的见源码，可是不知道源码会被谁使用以及如何使用
  + 服务代理了解服务规约，但是依然不知道具体的使用方法和源码
* 动态性测试
* 协同测试
  + 协同验证与确认（CV&V）
  + 独立验证与确认（IV&V）

|  |  |  |
| --- | --- | --- |
| 比较项目 | IV&V | CV&V |
| 方法 | 软件验证和确认小组通常独立于开发小组，以保证测试的客观性和全面性。软件的验证和确认大多由软件提供商完成 | 验证和确认是通过服务提供商、客户和独立的服务代理协同完成的，它强调实时、在线的测试和利用动态收集的数据进行评估 |
| 测试的地理位置 | 集中的多步测试 | 分布式、远程的、多代理和多步骤的测试 |
| 操作性测试 | 离线的领域测试和仿真 | 在应用程序环境下的在线即时测试 |
| 回归测试 | 离线的回归测试 | 利用动态收集的数据进行在线的回归测试 |
| 集成测试 | 静态的配置信息和系统必须在集成测试前链接 | 动态的配置和系统在运行时被链接和验证 |
| 测试覆盖 | 输入范围、结构性（白盒）和功能性（黑盒）覆盖 | 服务提供商可以使用全部的传统覆盖，服务中介和用户只能使用黑盒覆盖（如WSDL定义的服务规约） |
| 测试用例记录 | 静态记录 | 利用分散的代理手机的数据进行动态记录 |
| 可靠性建模 | 基于输入域的可靠性增长模型 | 基于动态记录和社团测试的可靠性模型 |
| 服务资格证明 | 静态服务资格证明中心 | 基于服务历史的动态资格证明 |
| 测试用例仓库 | 静态维护的仓库 | 动态扩展的仓库 |
| 模型验证 | 在源代码和状态模型上实施的模型检验 | 在描述文档和服务规约及组装上实施的即时动态模型验证 |
| 仿真 | 描述文档和模型被模拟执行以验证设计或源代码 | 组合服务的即时动态仿真 |