**RUP**

1、一个对象是一个具有明确界限并且封装了**状态** 和 **行为**的统一体

– 状态主要表现为属性和关联。

– 行为主要表现为操作，方法和状态机

2、**包**：包是分组的通常手法。它是可以包含其他模型元素的。一个通用机制，用于将元素组成组，是一个模型元素，能包含其他模型元素。

**类**：对具有相同职责、关系、操作、属性和语义的一组对象的说明。

3、**子系统**：包（包含其他模型元素）和类（拥有行为）的结合。实现定义在行为中的一个或者多个接口。

**包与子系统的对比：**

包：不提供行为、不完全封装其行为、可能不易于替代

子系统：提供行为、完全封装其行为、易于替代

4、**组件**是设计的物理实现。子系统能够表示设计中的一个组件。

5、**需求工作的目标包含:**

（1）提供一种与客户在系统功能方面进行沟通并达成共识的方式

（2）使开发者能够更准确的理解系统的需求

（3）确定系统的边界

（4）提供了对迭代过程中的技术内容进行计划的基础。

（5）为系统开发的成本估计提供一个基础

（6）定义出系统与用户之间的交互接口

6、用例模型：使用用例描述了系统的功能需求；模型化表示了系统的功能（用例）和系统的环境（角色）。（优点：交流、标识、验证）

7、分析和设计的目的是:

（1）将需求转化为系统未来的设计

（2）使系统引进更加合理的架构.

（3）使设计和实现环境相匹配，做性能设计

8、**分析**：关注问题的理解，理解问题；理想化设计；行为；系统结构；功能需求；小的模型。

**设计**：关注解决方案的理解；操作和属性；性能；接近实际的代码；对象的生命周期；非功能性需求；一个大的模型。

9、**软件架构**包含着一套关于这个软件系统组织的重要结论。

（1） 组成系统的结构元素和它们的接口

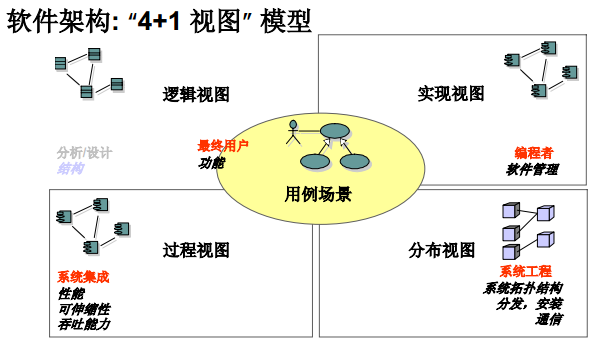
（2） 元素中协作的特定行为

（3）将结构和行为元素结合成一个大的子系统

（4）体系结构风格支配了组织结构

架构=元素+形式+数据

10、**4+1view**



（1）逻辑视图：当采用面向对象的设计方法时，逻辑视图即对象模型。

（2）实现视图：描述软件在开发环境下的静态组织。

（3）进程视图：描述系统的并发和同步方面的设计。

（4）部署视图：描述软件如何映射到硬件，反映系统在分布方面的设计。

（5）用例视图：指明子系统应该做什么。

12、**模式：**提供在一个环境下的共同问题的共同解决方案。

13、**框架：**确定结局问题的通用方法，提供一个概要解决方案。

区别：范围和规模上，框架是描述背景，不能缺少细节。而细节可以运用不同的分析、设计模式来填充。

14、**分析机制**：分析机制用于在分析过程中向设计人员提供复杂行为的简短表示，从而减少分析的复杂性并提高分析的一致性。种类：（1）分析机制（2）设计机制（3）实现机制

15、**用例分析步骤：**（1）补充用例说明（2）对每一个用例实现：– 从用例行为中查找类– 将用例行为分配给类（3）对每一个得到的分析类：– 说明职责– 说明属性和关联– 限定分析机制（4）统一分析类（5）检查点

16、**边界类**：接口和系统外部事物的中间体。包含一个主角的通信的行为。

17、**关联**：两个对象通常被认为是不相关，但是常常被连接在一起。

**聚合**：两个对象通过一个完整关系被紧密的约束在一起。

18、**用例图：**把满足用户需求的基本功能集合起来表示的工具。每一个参与系统的人员都能理解的方式列举出系统的业务需求。对于已经构造完成的系统，用例反映了系统能完成什么功能。

19、**类图：**描述类、接口、协作及他们之间关系的图示，是静态视图的一部分，表达系统内部重要的组成结构。

20、**顺序图：**用来描述对象之间动态的交互关系，着重体现对象间消息传递的时间顺序，重点在消息序列上，描述消息如何在对象间发送和接收。

21、**活动图：**一种表述过程机理、业务过程以及工作流的技术。他可以用来对业务过程、工作流建模，也可以对用例实现甚至是程序实现来建模。

22、**状态图：**用于显示状态机、使对象达到这些状态的事件和条件、以及达到这些状态时所发生的操作。是对类图的补充。在使用上并不需要对所有的类话状态图。

**COM部分**

1、**接口的三个函数：**

2、**不同接口的识别：**

3、**动态链接库实现过程（实现com组件的过程）：**

（1）定义接口及接口标示符，在接口中声明组件中需要使用的方法和变量，声明方法使用虚函数声明，定义的接口都要继承Iunknow。

（2）定义实现接口方法的抽象类。实例化类之后，才会对已经实例化的类对象分配空间。对类中接口的引用，可以通过对象指向接口的指针来调用。

（3）定义指向实例化类对象的指针，并可以调用他的接口返回标识，生成模块，定义文件，并在客户可以获取某个组件接口之前，将DLL装在到过程空间并创建此组件。

（4）创建类厂。在类厂中使用createInstance方法连接到动态链接库接口，并用指针指向这个接口，用DLLgetClassObject函数创建类厂，并根据类厂的接口标示符返回类厂指针。

（5）实例化类厂。使用cocreatInstance实例化类厂，生成一个实例类厂的对象，这个对象是可以被用户调用的。

4、**动态链接：**指用户在应用程序的运行中能够将组件替换掉的技术，使用组件的优点，使来源于用户可以动态的将组件插入或者卸载出应用程序，为在运行是能够改变组件，我们需要将组件动态的连接起来。

5、**com组件的结构：**由接口和实现两部分组成，实现部分对组件的客户来说是黑盒，客户能看到的是组件的接口，接口部分由若干接口组成。每一个接口是有序排列的一组函数指针，可以简单理解为一组函数。每一个接口相当于该组件对外的一个窗口，客户可以从不同的视角看到，获得该组件通过该接口提供的服务。对客户来说组件就是接口集，只能通过接口与组件交互。

6、**导出函数：**是已经编号的com组件供客户程序使用的函数。定义导出函数时需要在前面加上“extend c”的声明，再加上def函数，来告诉连接程序需要输出什么函数，客户程序再通过导出函数名获取此DLL中的导出函数地址，赋予相同参数和返回值的方法指针。获取类厂接口的导出函数的返回值应该是一个bool型，用来判断是否得到类厂接口。某一类厂只是针对某一组件的设计，所以需要判断参数中组件ID是否匹配。匹配，则实例一个类厂，调用查询方法查询该类厂接口ID是否存在，存在则返回一个类厂接口指针赋予最后一个参数，客户程序通过这个指针生成iunknown接口实例，在通过queryInterface查询其他接口调用其方法。

7、**类厂接口的作用：**

（1）客户可以通过类厂创建其他组件；

（2）类厂组件的唯一功能就是创建其他组件；

（3）创建组件的标准接口是IClassFactory，用CoCreateInstance创建的组件实际上是通过IClassFactory创建的；

（4）可以给客户提供一种能够控制组件创建过程的方法。在建立好一个组件之后，可以控制将组件装载到内存中何处或检车客户是否有权限；（5）类厂的特性：类厂的一个实例只能创建同某个CLSID相应的组件，类厂可以知道并确实具有它所创建的组件的一些特殊知识。

8、**新版本替换旧版本组件的影响：**

9、**怎么区分两个组件：**

10、**com组件标准的目标**：易于动态组装；组件的实现与语言无关，对客户透明；可以以二进制形式发布；组件的升级不影响视图继续使用它的老客户；组件可以透明的在网络上进行位置分配，对远程组件的使用和对本地机器上的组件的使用对客户来说无差别。

11、**com组件的规定：**（二进制性、接口的不变性、多态性）（2、3、4、5可以用来说**其他接口**）

（1）接口部分由若干接口组成。每个接口是有序排列的一组函数指针。客户通过接口获得组件提供的服务。

（2）所有组件都有一个称为Iunknown的接口，该接口一次给出三个函数：QI作用是对组件的接口进行查询；AR和release通过计数器共同完成引用计数器功能，实现组件的卸载。

（3）一个组件的其他接口可以看做是Iunknown接口的子接口。任何接口的前三个函数都相同。客户获得一个接口后可以查询其他接口是否存在。

（4）一旦公布了一个接口，该接口将永远不能发生任何变化。包括函数在接口中的顺序。改变接口的唯一方式，为接口增加新的接口。目的是兼容，不影响老用户对接口的使用。

（5）每个接口都有唯一的接口标识。一个组件也有唯一的一个标识。对接口中函数的确定，本质上是通过位置确定的。

12、**中间件：**中间件是位与平台之上的、应用软件之下之间的通用服务，这些服务具有标准的程序接口和协议。针对不同的操作系统和平台，他们可以有符合接口和规范的多种实现。两种性质：（1）完成一种或几种领域无关的功能；（2）必须要有规范的接口。

13、**ORB完成**了网络传递、压缩解压包等过程。用户可以只针对业务编写程序，而不需要知道它的内部具体实现的细节。 **EJB容器完成**：事物的相关工作，使用者万千不用知道具体的实现过程，这些全部由容器来完成，容器能够根据组件部署描述，插入所需要的服务。

**EJB**

1、**EJB部署描述文件的作用：**

（1）让容器认识自己是一个组件，介绍自己；

（2）申请服务：一些与组件无关的事情，指明需要帮助的事情；

（3）与组件种类相关的事情；部署和开发分离。

（4）定义若干量的值。

EJB部署文件是EJB的一个重要部分，它是EJB的说明文件，服务器容器按照部署文件来管理EJB，借助于部署描述文件，EJB组件能够声明其以来的中间件服务。然后EJB容器将通过部署描述符了解组件待使用的中间件服务，EJB部署文件是标准的xml文件，它遵守xml和相关文档类型DTD语法规范。

**部署描述文件**描述部署单位内容，并相对于应用环境配置组件和应用程序。同时对他们与组件之间的关系做出了一定的处理，从而可以在无需变成的情况下对这些关系加以治理。部署工具通常会自动生成部署描述文件，所以开发人员不比直接编辑和治理他们。

2、**定义bean类需要完成的工作（bean类中包括哪些种类的方法）：**

（1）与组件的业务接口有关：把组件业务里对外的每个业务流程或业务逻辑都给出实现；

（2）与home接口有关：home接口中的create操作在bean中也要有一个实现，home接口也是支持远程调用的；

（3）与另外一个接口有关：另外一个接口表明正在做的组件是三种bean中的哪一个，对应三种组件都有一个接口来进行约束，这个另一个接口就是用来表明态度，要实现的是哪一种EJB组件种类。

3、**考虑远程访问客户程序的情况下，构造bean需要的工作：**

（1）定义并实现提供业务方法的类：+上题；

（2）定义两个接口：home接口（工厂接口、类厂接口等）和组件业务接口（远程接口）；

（3）创建一个xml部署描述文件；

（4）把程序部分形成字节码+xml文件，做成一个jar包文件（bean，接口和部署描述文件放到一个ejb.jar文件中），作为EJB物理载体。

（5）用服务器开发商提供的工具把bean部署到服务器上

4、**容器：**各层的中间件在规范中乘坐容器。各层中有不同的组件。

（1）是J2EE的运行环境，为组件提供所需要的服务。

（2）J2EE的组件不直接相互作用，他们使用容器提供的协议和方法进行交互。

5、**EJB容器：**

（1）EJB容器是EJB运行环境，在EJB服务器上实现，管理EJB系统级服务。

（2）提供规范中定义的接口，提供了EJB类访问所需的服务，容器厂商也可以在容器或服务器中提供额外服务接口。

作用：为EJB组件实例提供生命周期的维护。加载EJB组件并管理其运行，可以创建EJBObject类。根据接口自动生成这个对象及相应的桩，容器还创建home类，实现EJB提供者定义的home接口，并产生home桩类，供客户端与home对话。

完成：事务处理、安全、并发、网络资源管理，持久存储管理，消息传递，及其他领域无关，业务无关的工作。

6、**容器能够自动定义并实现哪些类，能够维护哪些种类的对象：**Context

EJBObject类（实现组件业务接口）

EJBObject桩类（实现组件业务接口，并知道如何与EJBObject类对象进行联系）

Home类（实现home接口）

Home桩类（实现home接口，并知道如何与home类对象进行联系）

Context对象、Home对象

EJBObject对象（实现远程接口，并且客户端通过它访问EJB实例的商业方法）

7、**容器回调：**出现一个特殊时刻，bean不知道，容器调用了bean的一个容器回调方法是它才知道。容器回调出自两个接口中的方法（home接口，sessionbean接口）。

8、**容器回调函数的理解：**

回调函数就好像是一个中断处理函数，系统在符合你设定的条件是自动调用。为此，要做三件事：（1）声明（2）定义（3）设置出发条件，即在你的函数中把你的回调函数名称转化为地址作为一个参数，以便DLL调用，实现容器回调函数可以为空代码，一般需要保留Context

在应用程序使用会话bean时，对某些特定实例及用例，可能需要精确控制，如对象创建，对象销毁等时间。会话bean在创建时，可能需要执行一些数据库初始化操作，而在销毁时，需要关闭一些数据库连接，在此，通过回调方法，程序就能精确控制bean生命周期的每个阶段，而EJB容器将在bean生命周期的适当阶段对其进行调用。

9、**EJB构建描述（EJB有哪几个接口，每个接口的实现功能）：**

一个enterprise bean由接口和类组成；

（1）客户程序通过enterprise bean的home和远程接口来访问enterprise bean的方法；

（2）home接口提供了创建、删除和定位enterprise bean的方法；

（3）远程接口提供了业务方法；

（4）在部署时，容器由这些接口来创建类，使客户能够创建、删除、定位或调用位与enterprise bean上的业务方法。

10、**部署：一个EJB组件部署到EJB容器后，EJB有可能自动生成哪些与该组件相关的对象？作用分别？**

EJB组件负责实现home对象，远程组件EJBObject对象，以及context对象。Home对象就是用来创建EJB对象，主要功能是：创建，查找，删除对象。EJB对象承担了请求拦截远程对象的任务。

11、**在实现一个被远程访问的EJB组件是，bean类不直接实现远程自检业务接口的原因：**

因为组件业务接口继承了EJBObject接口，该接口的对象由容器生成，是可以远程调用的方法，客户在得到该对象的桩后，通过该对象的远程调用业务方法，EJBObject对象得到业务方法调用，容器介入，在bean上调用相应的业务方法。客户是不能远程直接调用bean类的，如果bean类实现了组件业务接口，那么客户调用时，就绕过了EJB容器直接调用了业务方法。

12、**EJBObject对象的作用：**

EJBObject实现远程接口，并且客户端通过它访问EJB实例。EJB开发者定义远程接口，容器开发商提供产生相应的EJBObject方法，客户端不能得到EJB实例的引用，只能得到他的EJBObject实例的引用，当客户端调用一个方法，EJBObject接受请求并把它传给EJB实例，同时提供进程中必须的包装功能，可以把EJBObject对象看成是bean类的代理。

13、**EJB与J2EE关系：**

（1）都是服务器规范，前者是后者的一个子集；（2）EJB组件无法在EJB容器之外存活，它无main方法，大多数方法由容器调用。

14、**EJB容器什么时候完成消息拦截：**容器拦截来自EJB存根对象的方法调用后，为调用添加事物上下文或者安全限制。

15、**EJB访问过程：**（1）EJB客户端通过JNDI查找EJB的home接口；（2）通过home接口缺省EJB的对外接口；（3）获得对外接口后，调用对外接口中的EJB业务方法，真正的逻辑是bean对象完成的。

16、**双向关联关系：**

关联关系：知道与被知道关系，隐含使用与被使用关系。回调函数是一种简单的关联关系。回调是不平衡的。

**双向关联关系的实现：**

（1）使用两个允引。即双方各有一个允引指向对方。缺点：关系发生改变时不易操作，关系建立，改变复杂。

（2）只保留一方的允引。关系维护简单，设计和实现复杂。

（3）用第三方维护。如登记表的形式，双方各有允引。

17、

**COBAR**

1、**对客户桩stub、骨架skeleton的理解：**

（1）客户桩stub：是server在Client端的代理。每个Client在本地都有一个stub，client要调用某功能，不与远程服务器联系，而是与stub联系，Client将请求发给stub，而由stub跟远程服务器联系。而stub把要执行操作发给服务器端，从服务器端得到结果，再把结果发给Client。因此stub实际上并没有完成操作，只是虚拟上实现了该业务。但对Client而言，业务是在本地由Stub完成了，Client不必考虑远程连接的细节。

（2）骨架skeleton：框架，也是一个代理。在server的本地有skeleton，server与skeleton直接交互，server不考虑远程连接的细节。Server认为是与Client在本地交互。Skeleton也是虚拟的实现了原属于Client端的方法。

2、**RIM过程：**客户对象一旦被调用，客户对象调用stub，而stub连接网络远程段的skeleton，而skeleton调用的是真正server的服务对象。而Client对象认为是直接调用了真正的服务对象。

3、**IDL编译器：**利用IDL定义CORBA组件的接口，由于IDL不是程序设计语言，只是描述语言，描述能力弱，因此，在实现组件接口是，需要利用某种程序设计语言来把接口中的每项规定一一实现。所以需要把IDL语法描述的接口转换为特定的程序设计语言的语法形式。组件的客户程序根据接口约定使用组件，同样需要把IDL语法描述的接口转换为特定程序设计语言的语法形式，这种转换器、解释器、翻译器称之为IDL编译器。

4、**IDL接口描述语言：**定义了某类对象的特征和行为，包括服务器能在这些对象上施加的操作。IDL是接口描述语言，描述能力弱，不能实现程序。

5、**IDL编译器自动生成stub、skeleton的过程：**

IDL编译器的输入是.idl文件，这个文件描述的是接口，接口中描述的有属性和操作。IDL编译器根据.idl文件中接口所描述的属性，分别生成类似get、set两个方法，将这些方法按照要转换成的特定程序设计语言的固定格式，“抄写”在文件中。同时生成ids和readObject、writeObject方法。由此生成stub文件。

同样，对于skeleton文件，把原.idl文件按固定格式“抄写”到skeleton文件，根据选择语句来完成不同的调用请求。每个方法都分配了id，在执行方法中，用选择语句case来完成不同调用。

6、**伪对象：**在corba中有许多伪对象。其中一个是object，在接口中定义了一些几乎在所有对象中都有效的操作，可以认为任何一个corba接口都自动从object继承这些操作，但不需要添加任何代码，这些操作实际由orb执行。“伪对象”接口由idl定义，但不是真正意义的对象，用户不能把伪对象作为操作参数使用，不能使用动态激发方式，调用其操作，在接口仓库中无定义。