# 第14章 组件级设计

## •什么是组件?

•OMG统一建模语言规范[OMG01]将组件定义为

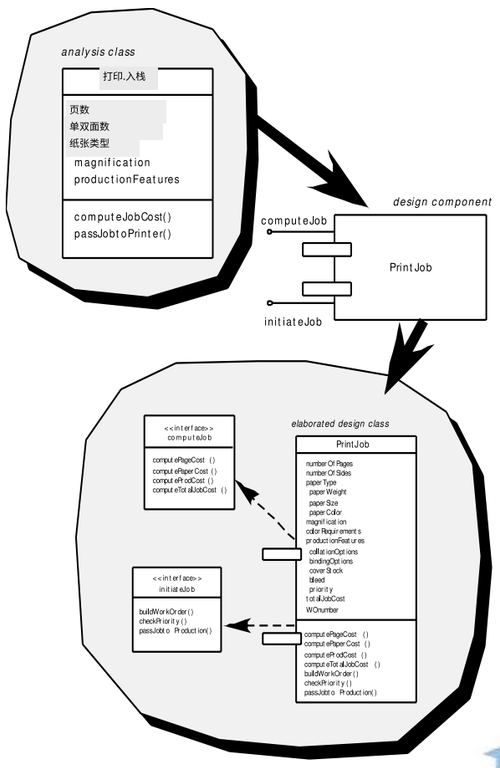
•".....系统中模块化、可部署且可替换的组成部分
它封装实现并暴露一组接口。"

•面向对象视角：组件包含一组协作类

•传统观点：组件包含处理逻辑、实现该逻辑所需的内部数据结构，以及调用组件并传入数据的接口。

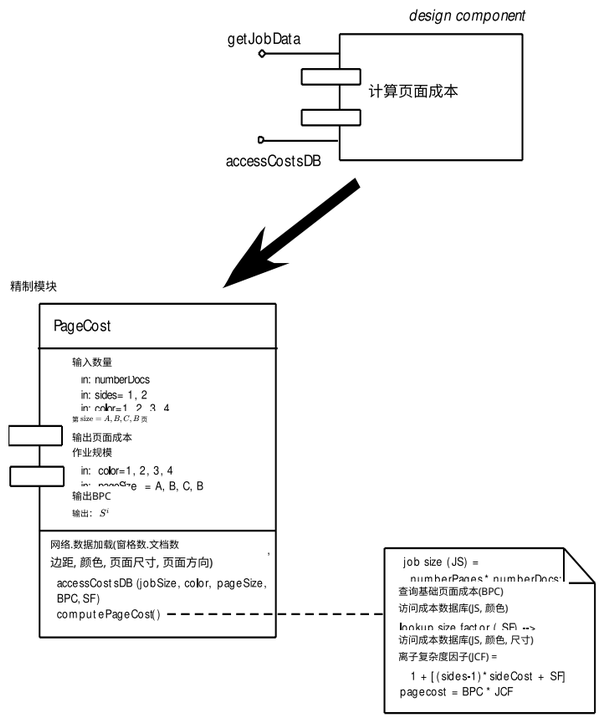


## •面向对象组件






## •常规组件






## •基础设计原则

•开闭原则(OCP)。"模块[组件]应对扩展开放，对修改关闭。

•里氏替换原则(LSP)。"子类必须能够替换其基类。

•依赖倒置原则(DIP)："依赖抽象，而非具体实现"

•接口隔离原则(ISP)："多个专用接口"
优于单一通用接口

•发布复用等价原则(REP)："复用的粒度就是发布的粒度"

•共同封闭原则(CCP)："共同变化的类应该放在一起"

•共同复用原则(CRP)："不被共同复用的类不应归为一组"



## •设计准则

组件

•应为架构模型中指定的组件建立命名规范，并在组件级模型中进一步细化和完善

•接口

•接口为通信与协作提供关键信息（同时帮助我们实现开闭原则）

•依赖与继承

•建议采用从左至右表示依赖关系，自下（派生类）而上（基类）表示继承关系的建模方式



## •内聚性

•传统观点：
•模块的"单一性"——面向对象
观点：
•内聚性意味着组件或类仅封装彼此密切相关且与类/组件本身紧密关联的属性和操作——内聚层次

•功能层

•通信层-顺序层-过程层

•时序效用



## •耦合度

•传统观点：
•组件与其他组件及外部环境的关联程度
外部环境
•面向对象视图：
•类间连接程度的定性度量

•耦合级别
•内容
•通用
— 控制
•印章
•数据
•例行调用
•类型使用
•包含或导入
•外部

•组件级设计 - 第一部分

•步骤1. 识别与问题域对应的所有设计类

问题域相关的设计类

•步骤2. 识别与基础设施域对应的

所有设计类

•步骤3. 细化所有尚未完善的设计类

作为可复用组件获取。

•步骤3a. 当类或组件协作时指定消息细节

组件协作。

•步骤3b. 为每个组件识别适当接口

组件。



•组件级设计 - II

•步骤3c. 详细阐述属性，定义实现这些属性所需的数据类型和数据结构。

•步骤3d. 详细描述每个操作内部的处理流程。

•步骤4. 描述持久化数据源（数据库及文件），并确定管理这些数据源所需的类。

•步骤5. 为类或组件开发并细化行为表示

。

•步骤6. 细化部署图以提供更多

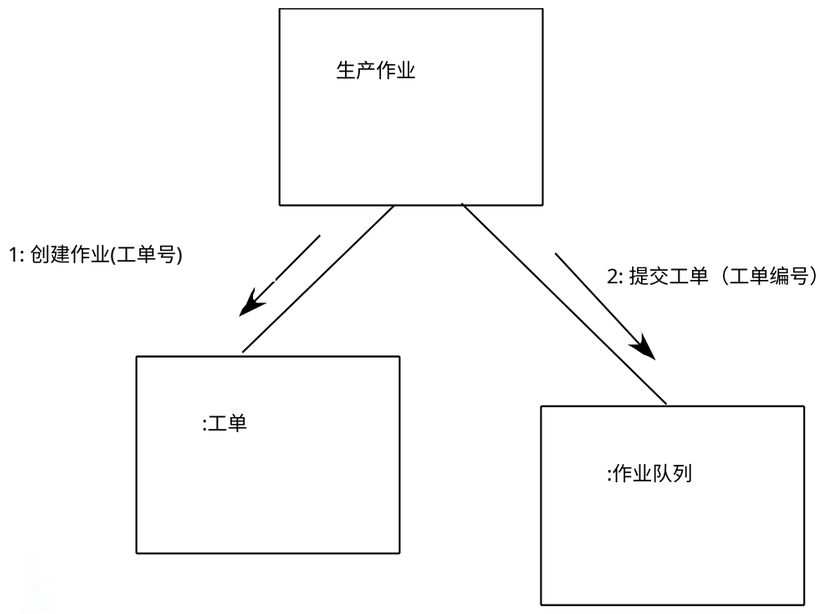
实现细节。

•步骤7. 分解每个组件级设计表示

并始终考虑替代方案。

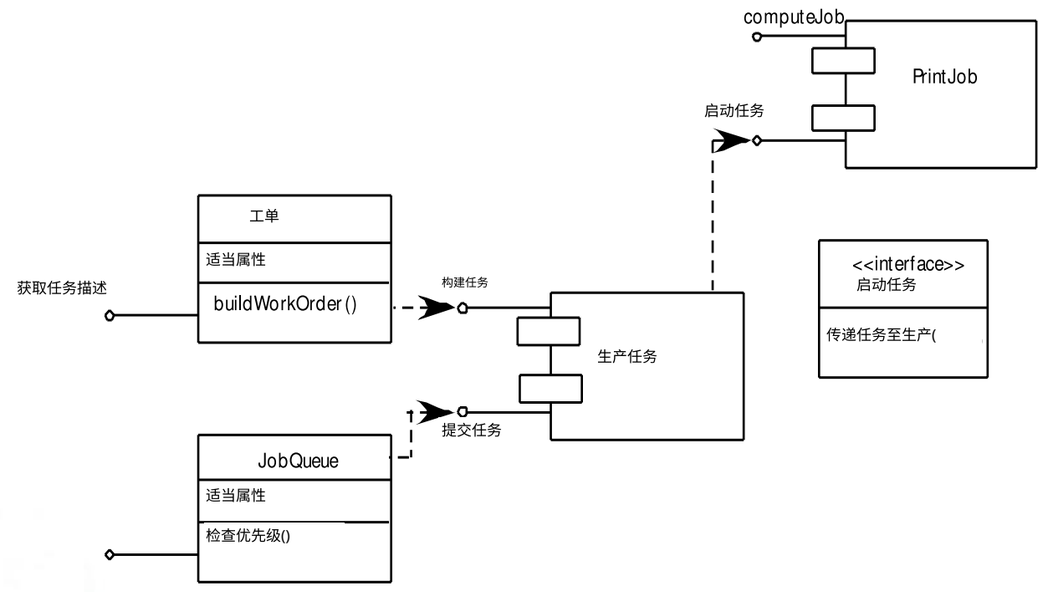


•协作图



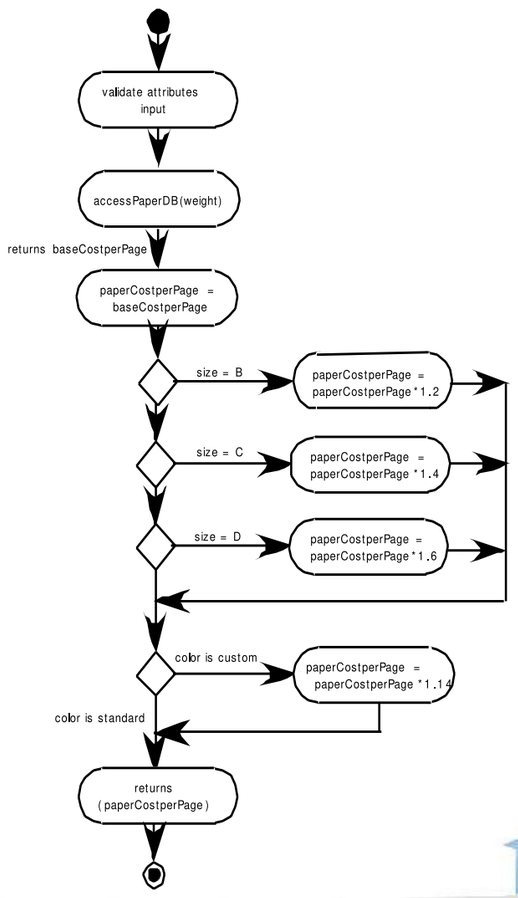



•重构



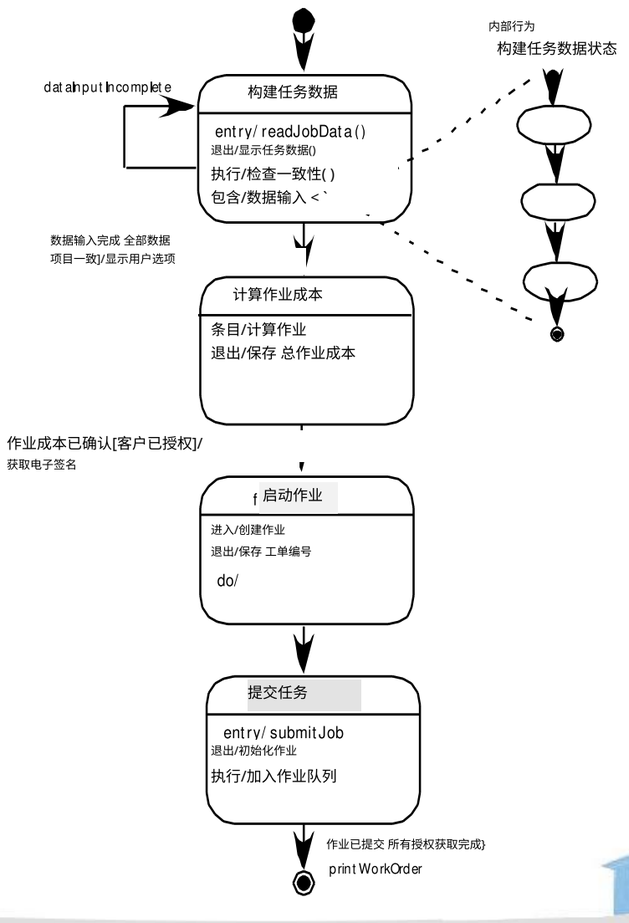



•活动图






状态图






•Web应用组件设计

•Web应用组件是

•(1) 一个定义明确的内聚功能，用于操作

内容或为终端用户提供计算或数据处理，

或

•(2) 一个内容与功能的内聚包，

为终端用户提供某些所需能力。

•因此，Web应用的组件级设计

通常融合了内容设计与

功能设计的要素



Web应用内容设计

•聚焦内容对象及其

面向终端用户的呈现包装方式

•考量基于网络的视频监控功能

居家安全保障网

•可为视频监控功能定义潜在内容组件：

监控功能：

•(1) 表示空间布局的内容对象（平面图）

附带代表传感器与视频摄像头位置的图标

摄像头；

•(2) 缩略视频截图集合（每个均为独立数据对象）

，以及

•(3) 特定摄像头的流媒体视频窗口

•每个组件均可单独命名

并作为整体包进行操作。

## •Web应用功能设计

•现代Web应用程序提供日益复杂的处理功能：

•(1) 执行本地化处理，以动态方式生成内容和导航功能；

•(2) 提供适合Web应用业务领域的计算或数据处理能力；

•(3) 实现高级数据库查询与访问，或

•(4) 与外部企业系统建立数据接口。

•为实现这些（及众多其他）功能，您将设计与构建形式等同于传统软件组件的WebApp功能模块。



## •Web应用功能设计

•轻量级网页客户端

•仅设备端保留界面层

•业务逻辑层与数据层通过网页或云服务实现

•富客户端

•设备上实现了全部三层架构（界面层、业务层、数据层）

•受限于移动设备性能



## •常规组件设计

•处理逻辑设计遵循算法设计与结构化编程基本原理

•数据结构设计由系统开发的数据模型定义

•界面设计取决于组件需实现的协作关系



## •基于组件的开发

•当面临复用可能性时，软件团队需考虑：- 是否有商用现成(COTS)组件可实现需求？- 是否有内部可复用组件可实现需求？- 可用组件的接口是否与待建系统架构兼容？- 同时他们面临以下复用障碍...



## •复用障碍

•极少企业/组织拥有哪怕略微接近全面软件复用计划的方案

•尽管越来越多软件供应商提供支持复用的工具/组件，但大多数开发者并未采用

•鲜有培训能帮助软件工程师和管理者理解应用复用技术

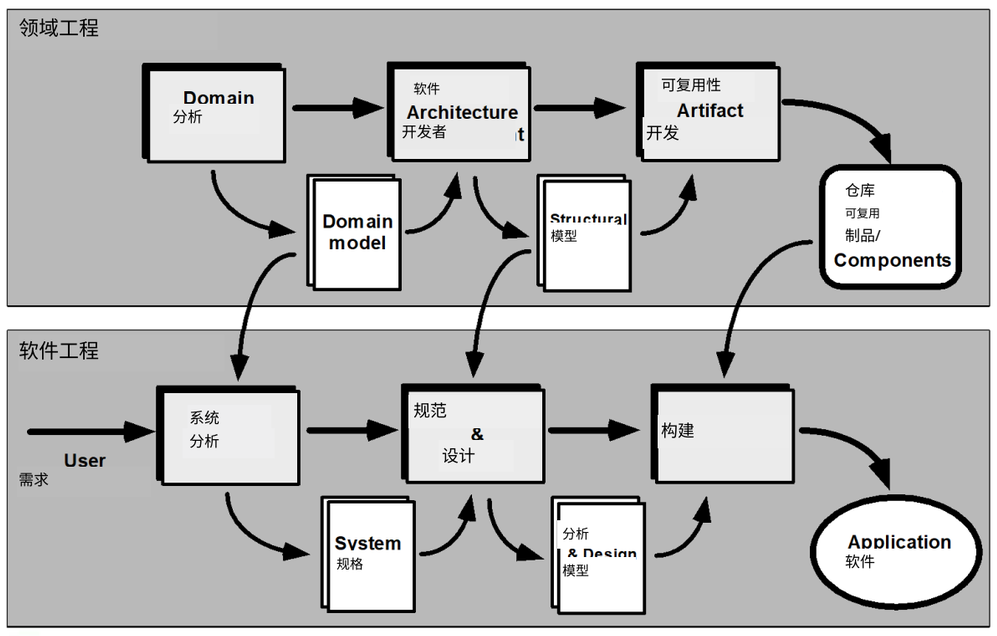
•许多软件从业者仍认为复用"弊大于利"。

•众多企业仍在推行不利于软件复用的开发方法

•鲜有公司对开发可复用程序组件提供激励



•基于构件的软件工程流程






## •领域工程

•1. 确定待研究的领域范围

•2. 对领域内提取的要素进行分类

•3. 收集该领域具有代表性的应用样本。

•4. 分析样本中的每个应用程序。

•5. 为对象建立分析模型。



## •识别可复用组件

•未来实现是否需要该组件功能?
•该组件功能在领域内的通用性如何?
•领域内是否存在该组件功能的重复实现?
•该组件是否依赖硬件?
•不同实施方案间的硬件是否保持不变?
•能否将硬件细节移至其他组件?
•当前设计是否已充分优化以支持下一阶段实施?
•能否通过参数化改造不可复用组件，使其成为可复用组件?
•该组件是否只需微小改动即可在多套方案中复用?变更?
通过修改实现复用是否可行?
•不可复用的组件能否被分解以产生可复用组件?
•组件分解对复用的有效性如何?

•基于组件的软件工程

•必须拥有组件库

•组件应具备统一结构

•应存在相应标准，例如

- OMG/CORBA

•微软COM组件

•Sun JavaBeans组件



•基于组件的软件工程活动

•组件鉴定

•组件适配

•组件组合

•组件更新



**•资质认证**

•Lotus 1-2-3的创始人米奇·卡普尔在《Dobbs博士期刊》上发表了"软件设计宣言"，其中提到：应用程序编程接口（API）

•组件所需的开发与集成工具
•运行时要求包括资源使用（如内存或存储）、时序或速度，以及网络协议
•服务要求包括操作系统接口及其他组件的支持

•安全特性包括访问控制和身份验证协议

•嵌入式设计假设，包括采用特定数值或非数值算法

•异常处理



**•适配**

•"轻松集成"的含义是：

•(1) 库中所有组件均采用统一的资源管理方法；

•(2) 所有组件共享数据管理等通用功能，且

•(3) 架构内部及与外部环境的接口已以一致方式实现。



**•组合**

•必须建立基础设施以绑定组件- 组合的架构要素包括：- 数据交换模型- 自动化- 结构化存储- 底层对象模型



**- OMG/CORBA**

•对象管理组已发布公共对象请求代理架构(OMG/CORBA)。

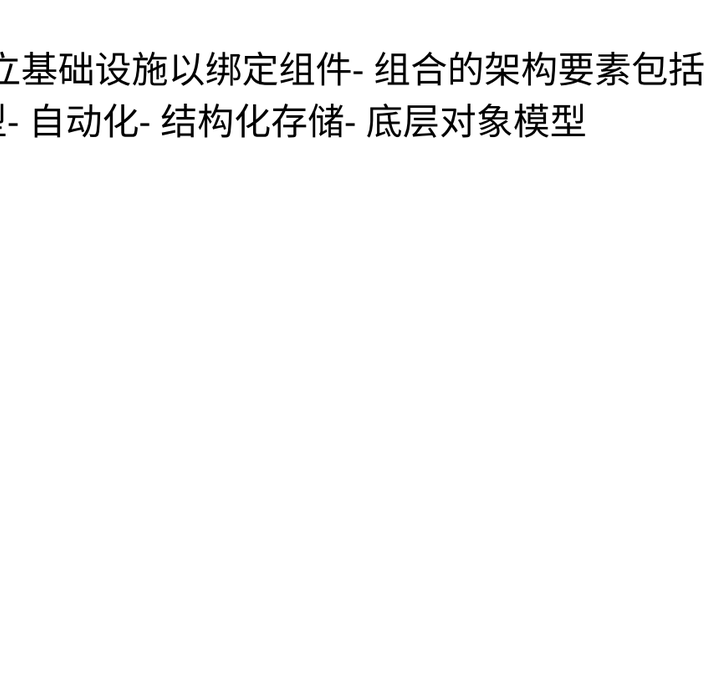
•对象请求代理(ORB)提供的服务使得
可复用组件(对象)能与其他组件通信
组件，无论其在系统中的位置如何。
•若为每个组件创建接口定义语言(IDL)接口，则能确保CORBA组件（无需修改）在系统中的集成。

•客户端应用程序中的对象向ORB服务器请求一个或多个服务。请求通过IDL或运行时动态发出。

•接口仓库包含有关服务请求和响应格式的所有必要信息。



•ORB架构






**•微软组件对象模型**

•组件对象模型(COM)规范了如何在Windows操作系统下的单一应用程序中使用不同厂商开发的组件。

•COM包含两大要素：

•COM接口（通过COM对象实现）

•以及一套用于在COM接口间注册和传递消息的机制。

•**Sun JavaBeans组件**

•JavaBeans组件体系是一种基于Java编程语言开发的、可移植且平台无关的 CBSE 基础设施。

•JavaBeans组件系统包含一套工具集（称为Bean开发套件BDK），可帮助开发者实现：

•分析现有Beans组件的工作原理- 定制其行为与外观- 建立协调通信机制- 开发特定应用场景的自定义Beans- 测试评估组件行为



•**分类**

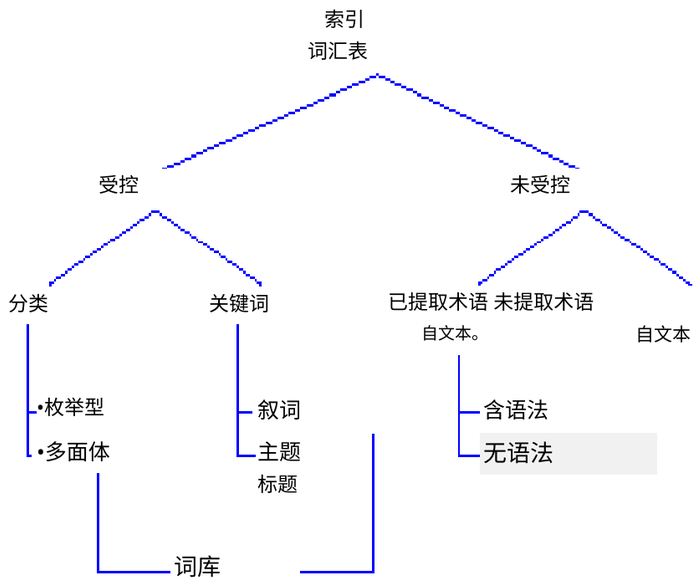
•枚举式分类——通过定义层次结构来描述组件，其中包含软件组件的类及各级子类

•分面分类——分析领域特征并识别一组基本描述性特征

•属性值分类——为领域内所有组件定义一组属性



•索引






•**复用环境**

•一个能存储软件组件及检索所需分类信息的组件数据库。

•一个提供数据库访问功能的图书馆管理系统数据库。

•软件组件检索系统（如对象请求代理），使客户端应用程序能从库服务器获取组件与服务。

•支持将复用组件集成到新设计或实现中的基于组件的软件工程工具。