# 制品

制品存在于比特的物质世界中，因此在对系统的物理方面进行建模时它是重要的构造块。制品是系统中物理的且可替换的部分。

可以利用制品对可能存在于结点上的物理事物（如可执行程序、库、表、文件和文档）进行建模。典型地，制品表示对诸如类、接口和协作等逻辑元素的物理打包。

# 名称

每一个制品都必须具有一个有别于其他制品的名称。名称是一个文字串，把单独的名称叫做简单名；用制品所在的包的名字作为前缀的制品名叫受限名。画一个制品时通常只显示它的名称。与类的画法类似，可以用标记值或表示其细节的附加栏来修饰制品。

# 制品和类

制品和类都是类别。然而，制品和类之间也有一些显著的差别。

1.类表示逻辑抽象，而制品表示存在于比特世界中的物理抽象，简而言之，制品可以存在于结点上，而类不可以。

2.制品表示对在实现平台上的比特的物理打包

3.类可以拥有属性和操作；制品可以实现类和方法，但是它们自身没有属性或操作。

第一项差别是最重要的。在对系统建模时，决定采用制品还是采用类涉及一个简单的决策—若准备建模的事物直接存在于结点上，就采用制品；否则采用类。第二项差别也说明了这个问题。

从第三项差别中可看出类于制品之间的联系。特别是，制品是一组逻辑元素（例如类及其协作）的物理实现。

制品图（Artifact Diagram）

制品必须具有一个唯一的名称。

简单名：   agent.java

受限名：   system::dialog.dll

**制品和类**

 制品和类都是类目。它们的区别：

 类表示逻辑抽象，制品是比特世界的物理抽象

 制品表示对在实现平台上的比特的物理打包

 类可以拥有属性和操作；制品可以实现类和方法，但它们自身没有属性或操作。

**制品的分类**

 部署制品：这类制品是构成一个可执行系统必要而充分的制品，例如：DLL、EXE文件。

工作产品制品：这类制品本质上是开发过程的产物，由源代码文件、数据文件等用来创建部署制品的事物构成。

 执行制品：这类制品是作为一个正在运行的系统的结果而被创建的。一般存在于内存之中。

**标准元素**

 可执行程序：说明一个可在结点上执行的制品

  库：说明一个动态或静态对象库

 文件：说明一个表示文档的制品，其中包含源代码或数据

  文档：说明一个表示文档的制品

**制品图**

制品图（artifact diagram）展示了一组制品及它们之间的关系。

关系包括：依赖、泛化、关联和实现关系

**制品图的一般用法**

 对源代码建模

 对可执行程序建模

 对物理数据库建模

 对可适应的系统建模