uml概述

## **幻灯片 1：封面**

* ****标题****：UML 概述 —— 统一建模语言的深度解析
* ****演讲人****：[演讲人姓名]

## **幻灯片 2：目录**

* UML 简介（UML 的主要视图和图）
* UML 建模过程
* UML 工具支持
* 优缺点总结
* 结论与展望

## **幻灯片 3：UML 简介**

定义：统一建模语言（UML）是一种通用的可视化建模语言，用于对软件系统、业务流程等进行可视化描述、构造和文档化。它为软件开发团队提供了一种标准的交流方式，使不同角色的人员（如分析师、设计师、程序员、测试人员等）能够更好地理解系统的架构和行为。

发展历程：介绍 UML 的起源和发展，从最初的多种建模语言竞争，到统一成为事实上的行业标准。

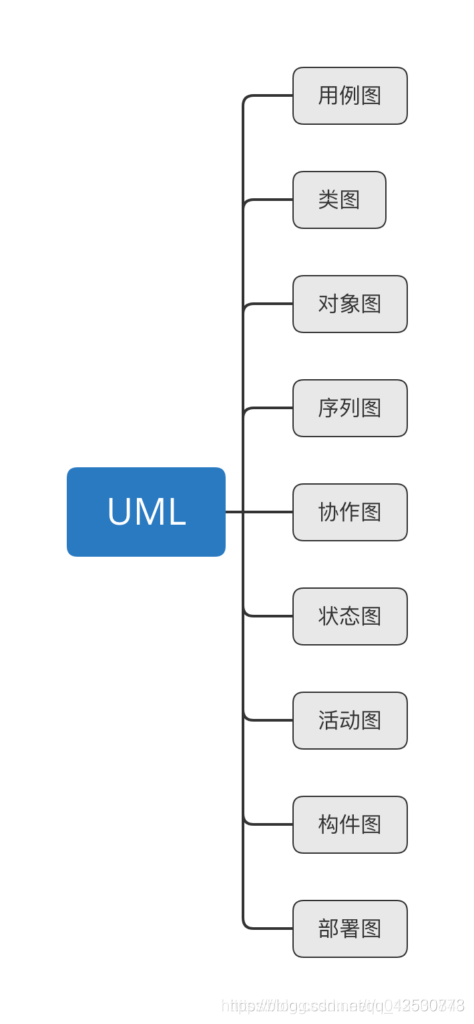
特点：

（1）UML统一了各种方法对不同类型的系统、不同开发阶段以及不同内部概念的不同观点，从而有效的消除了各种建模语言之间不必要的差异。它实际上是一种通用的建模语言，可以为许多面向对象建模方法的用户广泛使用。

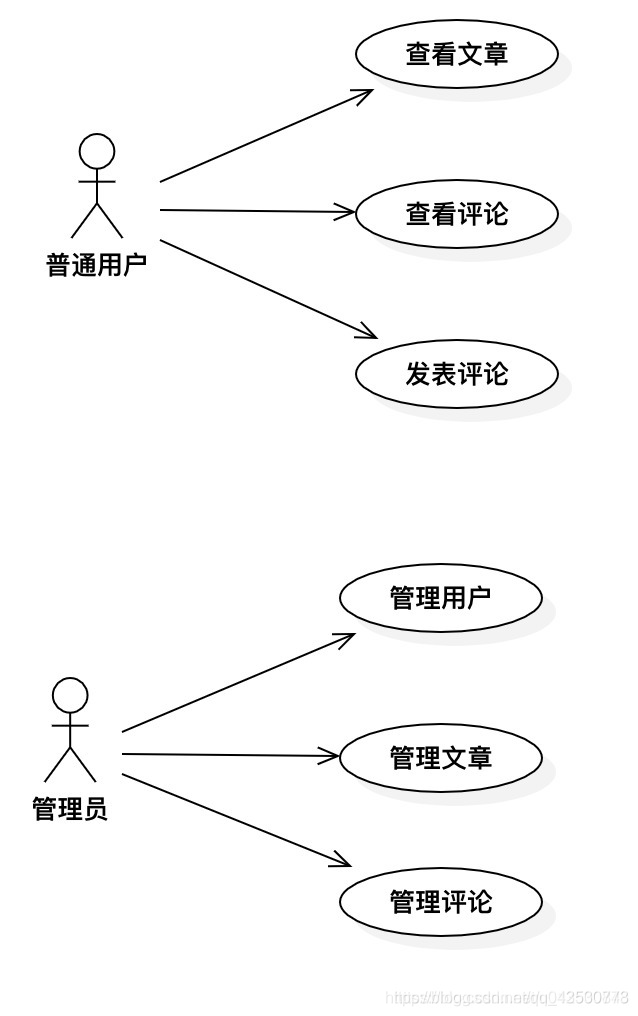
（2）UML建模能力比其它面向对象建模方法更强。它不仅适合于一般系统的开发，而且对并行、分布式系统的建模尤为适宜。

（3）UML是一种建模语言，而不是一个开发过程。

## **幻灯片 4：UML 的主要视图**



## 用例图（UseCase Diagrams） 用例图是从用户（角色）的角度出发，描述角色和用例之间的关系。即：谁要使用系统，一级他们使用系统可以做什么。简单来说就是：谁，可以用此系统做什么。



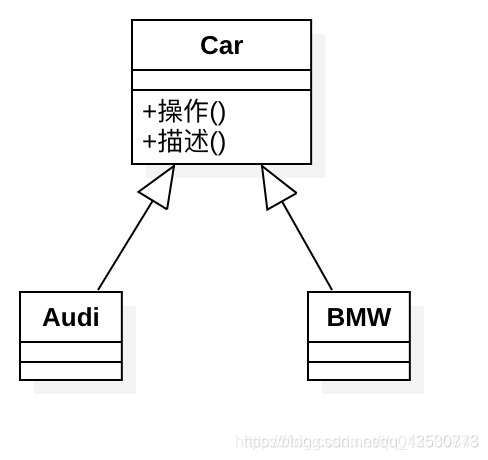
****2.类图（Class Diagram）****  
类图是描述系统中的类，以及各个类之间的关系的静态视图。是面向对象系统建模中最常用和最重要的图，是定义其他图的基础。

在UML类图中，常见的有以下几种关系: 泛化（Generalization）, 实现（Realization），关联（Association），聚合（Aggregation），组合（Composition），依赖（Dependency）。

各种关系的强弱顺序： 泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖

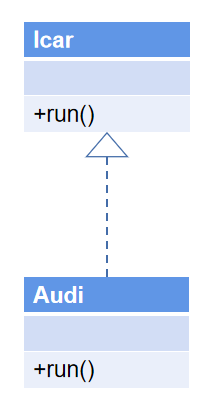
2.1.泛化（Generalization）

泛化是一种继承关系，表示一般与特殊的关系，它指定了子类如何继承父类的所有特征和行为。



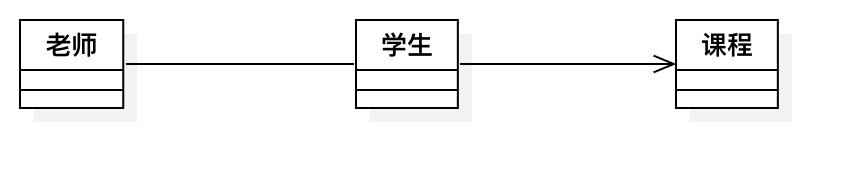
2.2.实现（Realization）

实现是一种类与接口的关系，表示类是接口所有特征和行为的实现。



2.3.关联（Association）

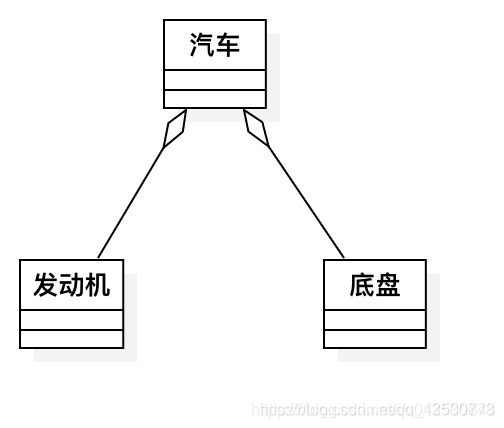
关联是一种拥有的关系，它使一个类知道另一个类的属性和方法；关联可以是双向的，也可以是单向的。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。



2.4.聚合（Aggregation）

聚合是整体与部分的关系，且部分可以离开整体而单独存在。如车和轮胎是整体和部分的关系，轮胎离开车仍然可以存在。

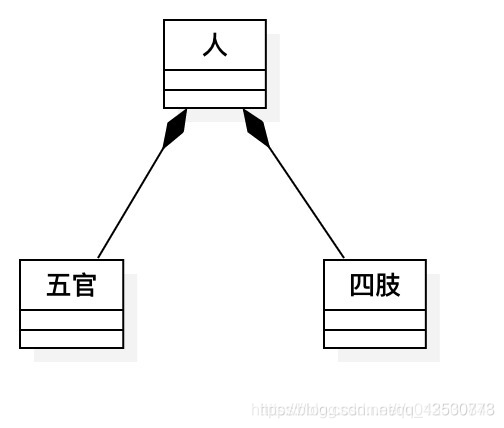
聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。



2.5.组合（Composition）

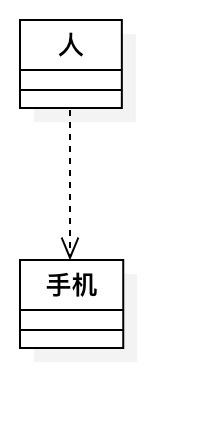
组合是整体与部分的关系，但部分不能离开整体而单独存在。如公司和部门是整体和部分的关系，没有公司就不存在部门。

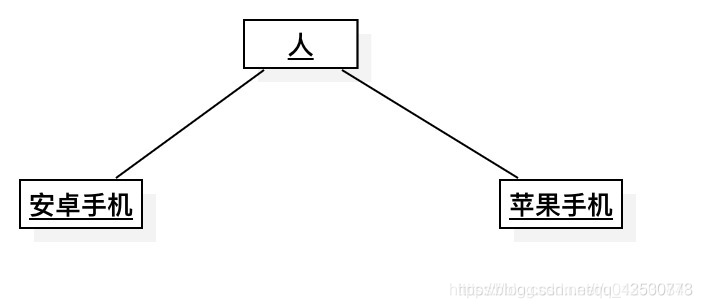
组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

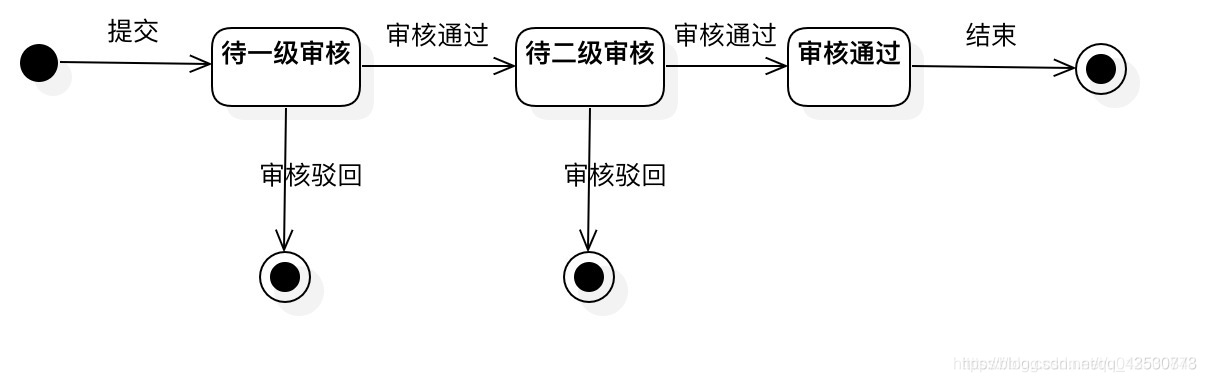


2.6.依赖（Dependency）

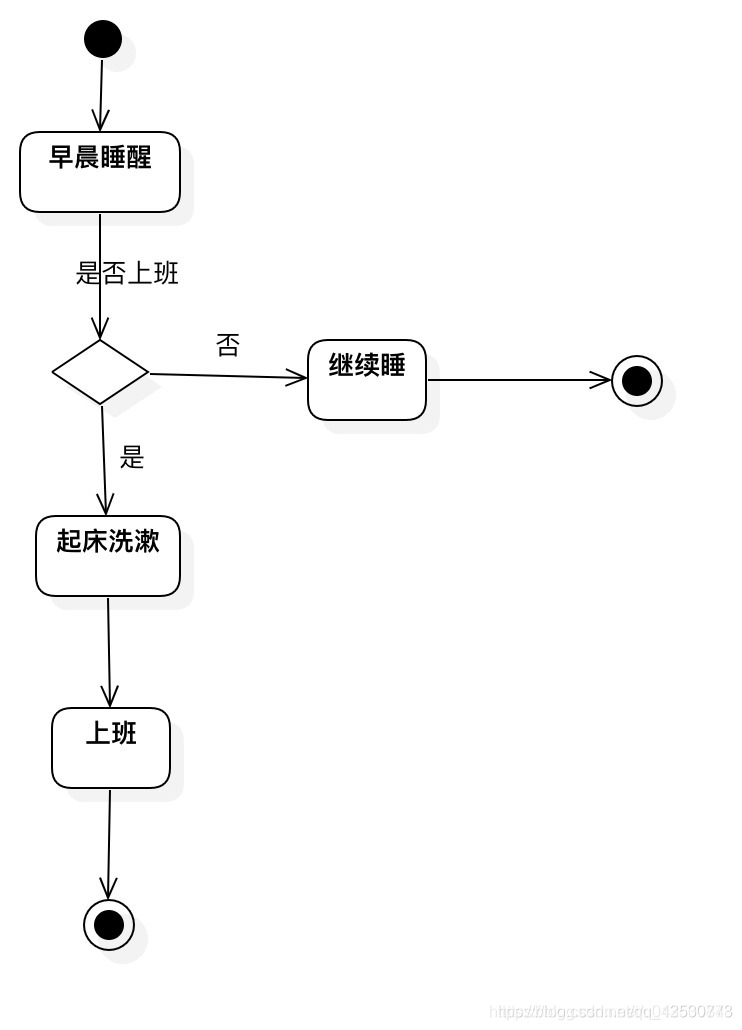
依赖是一种使用的关系，即一个类的实现需要另一个类的协助，所以要尽量不使用双向的互相依赖.

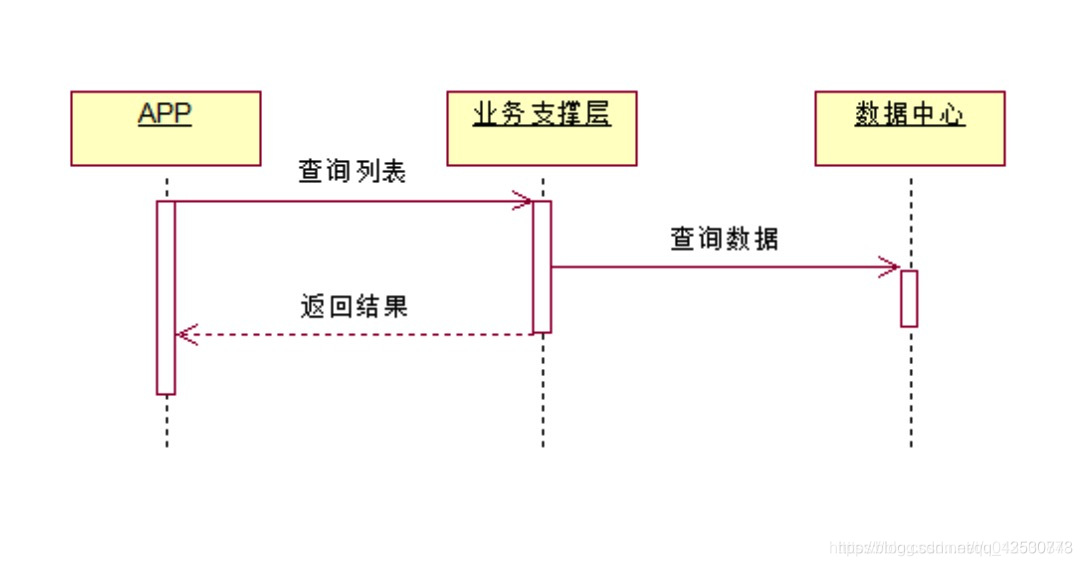
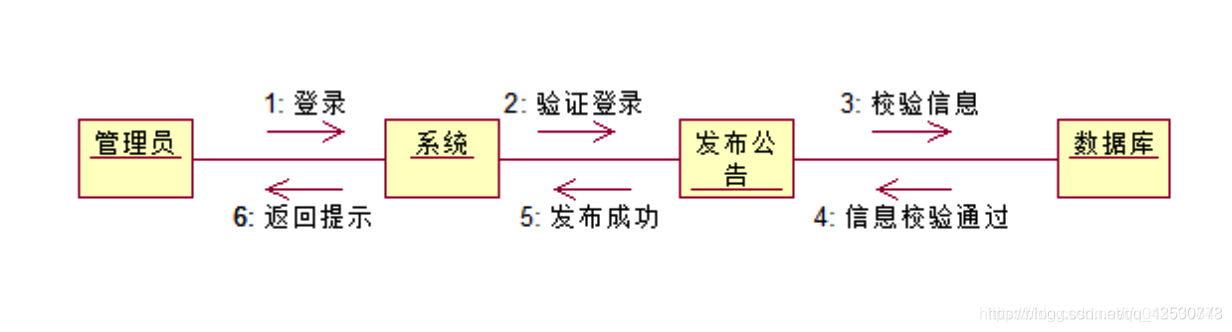


****3.对象图（Object Diagrams）****  
对象图和类图一样反映系统的静态过程，但它是从实际的或原型化的情景来表达的。对象图显示某时刻对象和对象之间的关系。一个UML对象图可看成一个类图的特殊用例，实例和类可在其中显示。  
  
****4.状态图（Statechart Diagrams）****  
状态图描述类的对象所有可能的状态，以及事件发生时状态的转移条件。他们可以告知一个对象可以拥有的状态，并且事件会怎么随着时间的推移来影响这些状态。

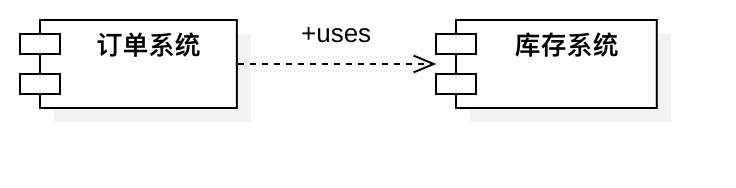
状态图是对类图的补充。  
  
****5.活动图（Activity Diagrams）****  
活动图描述用例要求所要进行的活动，以及活动间的约束关系，有利于识别并行活动。能够演示出系统中哪些地方存在功能，以及这些功能和系统中其他组件的功能如何共同满足前面使用用例图的业务需求。

活动图是状态图的一种特殊情况，这些状态大都处于活动状态。本质是一种流程图，它描述了活动到活动的控制流。

  
****6.序列图-时序图（Sequence Diagrams）****  
序列图是用来显示你的参与者如何以一系列顺序的步骤与系统的对象交互的模型。顺序图可以用来展示对象之间是如何进行交互的。序列图将显示的重点放在消息序列上，即强调消息是如何在对象之间被发送和接收的。

序列图展示的是多个系统或者对象之间的交互的顺序，强调时间顺序。  
  
****7.协作图（Collaboration Diagrams）****  
协作图和序列图相似，显示对象间的动态合作关系。可以看成是类图和顺序图的交集，协作图建模对象或者角色，以及它们彼此之间是如何通信的。  
  
8.构件图（Component Diagrams）  
构件图是用来表示系统中构件与构件之间，类或接口与构件之间的关系图。其中，构建图之间的关系表现为依赖关系，定义的类或接口与类之间的关系表现为依赖关系或实现关系。

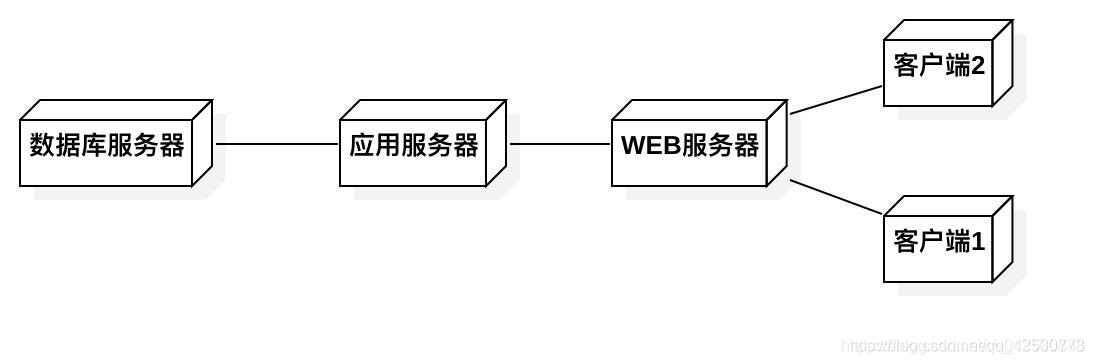
构件图也叫组件图，由组件、接口和组件之间联系构成，描述的是在软件系统中遵从并实现一组接口的物理的、可替换的软件模块。

组件图 = 构件（Component）+接口（Interface）+关系（Relationship）+端口（Port）+连接器（Connector）  
  
9.部署图（Deployment Diagrams）  
部署图又叫配置图，是用来建模系统的物理部署。

部署图由节点以及节点之间的关系组成。

部署图描述的是系统运行时的结构，展示了硬件的配置及其软件如何部署到网络结构中。

部署图通常用来帮助理解分布式系统，一个系统模型只有一个部署图。

部署图用于可视化的软件组件部署的系统中的物理组件的拓扑结构。  


## **幻灯片 7：UML 建模过程**

* ****需求分析阶段****：使用用例图捕获系统的功能需求，与客户和利益相关者沟通，明确系统的边界和功能。
* ****设计阶段****：根据需求分析的结果，创建类图、对象图等逻辑视图，设计系统的架构和对象模型。同时，使用序列图、状态图等描述系统的动态行为。
* ****实现阶段****：开发人员根据设计阶段的模型进行代码实现。UML 模型可以作为代码生成的基础，提高开发效率和代码质量。
* ****测试阶段****：根据 UML 模型编写测试用例，对系统进行功能测试、性能测试等。通过模型可以更好地理解系统的行为，发现潜在的问题。
* ****维护阶段****：当系统需要进行修改和扩展时，UML 模型可以帮助开发人员快速理解系统的结构和行为，降低维护成本。

## **幻灯片 8：UML 工具支持**

* ****主流 UML 工具介绍****：如 Rational Rose等。
* ****工具选择建议****：根据项目的规模、团队的技术水平、预算等因素，提供选择 UML 工具的建议和参考。

## **幻灯片 9：优缺点总结**

* ****优点****：
  + 可视化：UML 提供了丰富的图形符号，使系统的结构和行为一目了然，便于团队成员之间的沟通和理解。
  + 标准化：作为一种统一的建模语言，UML 具有标准的语法和语义，有助于不同项目和团队之间的交流和协作。
  + 可扩展性：UML 支持扩展机制，如版型、约束等，可以根据项目的特定需求进行定制。
  + 支持软件开发全生命周期：从需求分析到维护阶段，UML 都能提供有效的支持，有助于提高软件开发的效率和质量。
* ****缺点****：
  + 学习曲线较陡：UML 的概念和图形较多，对于初学者来说，需要花费一定的时间和精力去学习和掌握。
  + 模型与代码的同步问题：在软件开发过程中，模型和代码可能会出现不一致的情况，需要花费额外的精力去维护两者的同步。
  + 过度建模风险：在一些小型项目中，过度使用 UML 可能会增加项目的复杂度和成本，降低开发效率。

## **幻灯片 10：结论与展望**

* ****结论****：UML 作为一种强大的建模工具，在软件开发中发挥着重要的作用。它为团队提供了统一的交流方式，有助于提高软件的质量和开发效率。虽然 UML 存在一些不足之处，但通过合理的使用和结合其他技术，可以充分发挥其优势。
* ****展望****：随着软件技术的不断发展，UML 也在不断演进和完善。未来，UML 可能会更加注重与新兴技术的结合，如人工智能、大数据等，以满足日益复杂的软件系统的建模需求。同时，UML 工具也将更加智能化和便捷化，进一步提高开发人员的工作效率。