面向方面的对象和工程

面向切面编程(也叫面向方面)：Aspect Oriented Programming(AOP),是目前软件开发中的一个热点。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。　　  
AOP是OOP的延续，是（Aspect Oriented Programming）的缩写，意思是面向切面（方面）编程。

在大型工程中，需求和组件的要求往往很复杂，因此在多对多关系中，系统维护和重用组建往往很困难，ASOS主要就是为了使软件更加易于维护和管理。

AOSA基于一种称为方面的新型抽象。方面与其他抽象（例如对象和方法）一起使用。方面封装了可以交叉使用并与系统中其他功能共存的功能。它提供了具有其他体系结构类型的辅助体系结构。根据源代码中包含的规范，通过自动组合（编织）对象，方法和方面来创建面向可执行方面的程序。

AOP的现状：  
AOSD尚未成为主流SD的一部分。尽管它可能具有缺点，但它具有独特的优势并且正变得越来越流行。

横切关注点（concern）



关注点不是程序问题，而是反映系统要求和系统涉众的优先级。一个典型的软件系统包含一些核心的关注点和系统级的关注点。举个例子来说，一个信用卡处理系统的核心关注点是借贷/存入处理，而系统级的关注点则是日志、事务完整性、授权、安全及性能问题等，许多关注点——即横切关注点（crosscutting concerns）——会在多个模块中出现。如果使用现有的编程方法，横切关注点会横越多个模块，结果是使系统难以设计、理解、实现和演进。AOP能够比上述方法更好地分离系统关注点，从而提供模块化的横切关注点。

例如一个复杂的系统，它由许多关注点组合实现，如业务逻辑、性能，数据存储、日志和调度信息、授权、安全、线程、错误检查等，还有开发过程中的关注点，如易懂、易维护、易追查、易扩展等，图2.1演示了由不同模块实现的一批关注点组成一个系统。通过对系统需求和实现的识别，我们可以将模块中的这些关注点分为：**核心关注点和横切关注点**。对于核心关注点而言，通常来说，实现这些关注点的模块是相互独立的，他们分别完成了系统需要的商业逻辑，这些逻辑与具体的业务需求有关。而对于日志、安全、持久化等关注点而言，他们却是商业逻辑模块所共同需要的，这些逻辑分布于核心关注点的各处。在AOP中，诸如这些模块，都称为横切关注点。应用AOP的横切技术，关键就是要实现对关注点的识别。

动态横切

动态横切是通过切入点和连接点在一个方面中创建行为的过程，连接点可以在执行时横向地应用于现有对象。动态横切通常用于帮助向对象层次中的各种方法添加日志记录或身份认证。在很多应用场景中，动态横切技术基本上代表了AOP。

动态横切技术的核心主要包括**join point（连接点），point cut（切入点），advice（通知）和aspect（方面）**。在前面，我已经概要地介绍了这些术语分别代表的含义。接下来，我将以一个具体的实例来进一步阐述它们在AOP动态横切中实现的意义。

考虑一个电子商务系统，需要对订单进行添加、删除等管理操作。毫无疑问，在实际的应用场景中，这些行为应与权限管理结合，只有获得授权的用户方能够实施这些行为。

Viewpoints and Concerns



**AOP技术的优势**

AOP技术的优势是显而易见的。在面向对象的世界里，人们提出了各种方法和设计原则来保障系统的可复用性与可扩展性，以期建立一个松散耦合、便于扩展的软件系统。例如GOF提出的“设计模式”，为我们提供了设计的典范与准则。设计模式通过最大程度的利用面向对象的特性，诸如利用继承、多态，对责任进行分离、对依赖进行倒置，面向抽象，面向接口，最终设计出灵活、可扩展、可重用的类库、组件，乃至于整个系统的架构。在设计的过程中，通过各种模式体现对象的行为、暴露的接口、对象间关系、以及对象分别在不同层次中表现出来的形态。然而鉴于对象封装的特殊性，“设计模式”的触角始终在接口与抽象中大做文章，而对于对象内部则无能为力。

通过“横切”技术，**AOP技术就能深入到对象内部翻云覆雨，截取方法之间传递的消息为我所用。**由于将核心关注点与横切关注点完全隔离，使得我们能够独立的对“方面”编程。它允许开发者动态地修改静态的OO模型，构造出一个能够不断增长以满足新增需求的系统，就象现实世界中的对象会在其生命周期中不断改变自身，应用程序也可以在发展中拥有新的功能。

设计软件系统时应用AOP技术，其优势在于：

* 在定义应用程序对某种服务（例如日志）的所有需求的时候。通过识别关注点，使得该服务能够被更好的定义，更好的被编写代码，并获得更多的功能。这种方式还能够处理在代码涉及到多个功能的时候所出现的问题，例如改变某一个功能可能会影响到其它的功能，在AOP中把这样的麻烦称之为“纠结（tangling）”。
* 利用AOP技术对离散的方面进行的分析将有助于为开发团队指定一位精于该项工作的专家。负责这项工作的最佳人选将可以有效利用自己的相关技能和经验。
* 持久性。标准的面向对象的项目开发中，不同的开发人员通常会为某项服务编写相同的代码，例如日志记录。随后他们会在自己的实施中分别对日志进行处理以满足不同单个对象的需求。而通过创建一段单独的代码片段，AOP提供了解决这一问题的持久简单的方案，这一方案强调了未来功能的重用性和易维护性：不需要在整个应用程序中一遍遍重新编写日志代码，AOP使得仅仅编写日志方面（logging aspect）成为可能，并且可以在这之上为整个应用程序提供新的功能。

总而言之，AOP技术的优势使得需要编写的代码量大大缩减，节省了时间，控制了开发成本。同时也使得开发人员可以集中关注于系统的核心商业逻辑。此外，它更利于创建松散耦合、可复用与可扩展的大型软件系统。

**apo技术的缺点**

1.针对面向方面的程序的检查和派生测试的问题是采用AOSA和AOSD的重大障碍。

2.程序检查问题：无法从源代码中得知方面切面将在何处进行编制和执行，因此程序检查会变得非常困难

3.流程图问题

不可能导出具有方面的程序的程序流程图。因此，很难系统地设计测试以确保执行基本代码和方面的所有组合。