# Git

# 软件模式

<https://design-patterns.readthedocs.io/zh_CN/latest/>

这个中文网址写的很好。这里提到了三个层次，非常贴切。

a) 能在白纸上画出所有的模式结构和时序图。

b) 能用代码实现；如果模式的代码都没有实现过，是用不出来的；即所谓，看得懂，不会用。

c) 灵活应用到工作中的项目中。

## UML类图和时序图



* 车的类图结构为<abstract>，表示车是一个抽象类；
* 它有两个继承类：小汽车和自行车；它们之间的关系为实现关系（realize），使用带空心箭头的虚线表示；
* 小汽车为与SUV之间也是继承关系，它们之间的关系为泛化关系（generalize），使用带空心箭头的实线表示，泛化关系表现为继承非抽象类；
* 小汽车与发动机之间是组合关系，使用带实心箭头的实线表示；
* 学生与班级之间是聚合关系，使用带空心箭头的实线表示；
* 学生与身份证之间为关联关系，使用一根实线表示；
* 学生上学需要用到自行车，与自行车是一种依赖关系，使用带箭头的虚线表示；

笔记：在梳理这些关系的时候，通过假设是否存在来分析两者之间的关系。

## 时序图

<https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/?ref=lbp>

通过箭头来查看异步消息。





有时候时序图的箭头很让人困惑。把两个对象理解成两个人在交谈，Sender: Hi，Welcome. Receiver: Got it (也许收到了，但是不作回答)。

## 状态图



**Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

导致，是event导致了状态的变化。

## 活动图

也就是流程图，用于流程控制。用例图只能用来表示high level 的内容。

 An activity diagram can be used to illustrate a business process (high level implementation) to a stand alone algorithm (ground level implementation). However, Use cases have a low level of abstraction. They are used to show a **high level** of implementation only.



例如下图 – 当蒸牛奶和添加咖啡这两项活动都完成后，我们将它们合并为一项最终活动。这里面有Fork和Join。



# Linux

## CentOS

CentOS适用于服务器。

需要参加红帽RHCSA、RHCE、RHCA认证，肯定要安装RHEL或CentOS系统。

## Ubuntu

Ubuntu则适用于个人桌面。但其实Ubuntu在服务器方面也并不比CentOS弱，并且ubuntu 相比 centos 在更新策略上要积极一些。

### Xubuntu

Xubuntu（发音为ZOO-bun-too）是一个Ubuntu Linux的官方派生版，它基于桌面环境Xfce，主要运行基于GTK+的程序，面向旧式电脑的用户和寻求更快捷的桌面环境的用户。

它与Ubuntu不完全相同，但使用Ubuntu的高质量软件源。

Xubuntu是一个完整的基于Ubuntu的GNU/Linux发行版，但是更为轻量，比使用Gnome和KDE的Ubuntu系统更有效率，因为其使用了Xfce桌面环境。

### 安装问题

### Command