什么是版本控制

版本控制（Revision control）是维护工程[蓝图](http://zh.wikipedia.org/wiki/%E8%97%8D%E5%9C%96)的标准作法，能追踪工程蓝图从诞生一直到定案的过程。此外，版本控制也是一种[软件工程](http://zh.wikipedia.org/wiki/%E8%BB%9F%E9%AB%94%E5%B7%A5%E7%A8%8B)技巧，借此能在软件开发的过程中，确保由不同人所编辑的同一程式档案都得到同步。

版本控制能提供项目的设计者，将设计回复到之前任一状态的选择权，这种选择权在设计过程进入[死胡同](http://zh.wikipedia.org/wiki/%E6%AD%BB%E8%83%A1%E5%90%8C)时特别重要。

理论上所有的资讯记录都可以加上版本控制，在过去的实务中，除了软件开发的流程，其它的领域中很少有使用较复杂的版本控制技巧与工具（虽然可能为其带来许多好处）。目前已有人开始用版本控制软件来管理[CAD](http://zh.wikipedia.org/wiki/CAD)电子档案，电路板设计，来补足本来由人手工执行的传统版本控制。

[软件设计师](http://zh.wikipedia.org/wiki/%E8%BB%9F%E4%BB%B6%E8%A8%AD%E8%A8%88%E5%B8%AB)常会利用版本控制来追踪、维护[源码](http://zh.wikipedia.org/wiki/%E6%BA%90%E7%A2%BC)、[文件](http://zh.wikipedia.org/wiki/%E6%96%87%E4%BB%B6)以及[设定档](http://zh.wikipedia.org/w/index.php?title=%E8%A8%AD%E5%AE%9A%E6%AA%94&action=edit&redlink=1)等等的改动，并且提供控制这些改动控制权的[程序](http://zh.wikipedia.org/wiki/%E7%A8%8B%E5%BA%8F)。

在最简单的情况下，[软件设计师](http://zh.wikipedia.org/wiki/%E8%BB%9F%E4%BB%B6%E8%A8%AD%E8%A8%88%E5%B8%AB)可以自己保留一个程式的许多不同版本，并且为它们做适当的编号。这种简单的方法已被用在很多大型的软件项目中。该方法虽然可行，但不够有效率。除了必须同时维护很多几乎一样的[源码](http://zh.wikipedia.org/wiki/%E6%BA%90%E7%A2%BC)备份外；而且极度依赖[软件设计师](http://zh.wikipedia.org/wiki/%E8%BB%9F%E4%BB%B6%E8%A8%AD%E8%A8%88%E5%B8%AB)的自我修养与开发纪律，但这却常是导致错误发生的原因。

有时候，一个程式同时存有两个以上的版本也有其必要性，例如：在一个为了部署的版本中[程式错误](http://zh.wikipedia.org/wiki/%E7%A8%8B%E5%BC%8F%E9%8C%AF%E8%AA%A4)已经被修正、但没有加入新功能；在另一个开发版本则有新的功能正在开发、也有新的错误待解决，这使得同时间需要不同的版本并修改。

此外，为了找出只存在于某一特定版本中（为了修正了某些问题、或新加功能所导致）的[程式错误](http://zh.wikipedia.org/wiki/%E7%A8%8B%E5%BC%8F%E9%8C%AF%E8%AA%A4" \o "程式错误)、或找出[程式错误](http://zh.wikipedia.org/wiki/%E7%A8%8B%E5%BC%8F%E9%8C%AF%E8%AA%A4)出现的版本，软件除错者也必须借由比对不同版本的程式码以找出问题的位置。

最简单的版本控制就是保留软件不同版本的数份copy，并且适当编号。许多大型开发案都是使用这种简单技巧。虽然这种方法能用，但是很没效率。一是因为保存的数份copy几乎完全一样，也因为这种方法要高度依靠开发者的自我纪律，而常导致错误。因此，有人开发出了将部份或全部版本控制工作自动化的版本控制系统。

#### 中央式系统与分散式系统

大部分的软件开发案，会有好几个开发人员同时工作。如果两个人员同时要改变同一个档案，而没有管理存取权限，很可能会[覆写](http://zh.wikipedia.org/w/index.php?title=%E8%A6%86%E5%AF%AB&action=edit&redlink=1)彼此的工作。

所以权限管理控制系统会在两种方法中择一解决：采用中央式系统，由中央权威管理存取权限；或是像分散式系统容许多个单位同时进行，包括同时更动同一档案。

传统上版本控制系统都是采用中央式系统：所有版本控制的工作在一个服务器进行，由中央权威管理存取权限“锁上”档案库中的档案，一次只让一个开发者工作。

2000年后，TeamWare、BitKeeper、和[GNU](http://zh.wikipedia.org/wiki/GNU)开始用分散式系统：开发者直接在各自的本地档案库工作，并容许多个开发者同时更动同一档案，而各个档案库有另一个[合并](http://zh.wikipedia.org/wiki/%E5%90%88%E4%BD%B5)各个改变的功能。这个方式让开发者能不靠网络也能继续工作，也让开发者有充分的版本控制能力，而不需经中央权威许可。分散式系统仍然可以有档案上锁功能。

分散式系统[Linux内核](http://zh.wikipedia.org/wiki/Linux%E5%85%A7%E6%A0%B8)的发明人[林纳斯·托瓦兹](http://zh.wikipedia.org/wiki/%E6%9E%97%E7%B4%8D%E6%96%AF%C2%B7%E6%89%98%E7%93%A6%E8%8C%B2)就是分散式版本控制系统的支持者，他开发了目前被开源社群广泛使用的分散式版本控制系统[Git](http://zh.wikipedia.org/wiki/Git)。

什么是Git(分布式版本控制系统)

Git是一个分布式的版本控制系统，最初由[Linus Torvalds](http://baike.baidu.com/view/518124.htm" \t "_blank)编写，用作[Linux内核](http://baike.baidu.com/view/573460.htm" \t "_blank)代码的管理。在推出后，Git在其它项目中也取得了很大成功，尤其是在Ruby社区中。目前，包括[Rubinius](http://baike.baidu.com/view/1772201.htm" \t "_blank)和Merb在内的很多知名项目都使用了Git。Git同样可以被诸如[Capistrano](http://baike.baidu.com/view/7283319.htm" \t "_blank)和Vlad the Deployer这样的部署工具所使用。

特点

分布式和集中式的最大区别在于开发者可以本地提交。每个开发者机器上都有一个服务器的数据库。

Git的功能特性：

从一般开发者的角度来看，git有以下功能：

1：从服务器上克隆数据库（包括代码和版本信息）到单机上。

2：在自己的机器上创建分支，修改代码。

3：在单机上自己创建的分支上提交代码。

4：在单机上合并分支。

5：新建一个分支，把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。

6：生成[补丁](http://baike.baidu.com/view/228.htm" \t "_blank)（patch），把补丁发送给主开发者。

7：看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。

8：一般开发者之间解决冲突的方法，开发者之间可以使用pull 命令解决冲突，解决完冲突之后再向主开发者提交[补丁](http://baike.baidu.com/view/228.htm" \t "_blank)。

什么是Github

作为开源代码库以及版本控制系统，Github目前拥有140多万开发者用户。随着越来越多的应用程序转移到了云上，Github已经成为了管理软件开发以及发现已有代码的首选方法。

GitHub可以托管各种git库，并提供一个web界面。

GitHub项目本身自然而然的也在GitHub上进行托管，只不过在一个私有的，公共视图不可见的库中。[开源项目](http://baike.baidu.com/view/188327.htm)可以免费托管，但私有库则并不如此。Chris Wanstrath，GitHub的开发者之一，肯定了通过付费的私有库来在财务上支持免费库的托管这一计划。

任何希望节省时间并希望和团队其它成员一样远离页面频繁转换之苦的人士都会从GitHub中获得他们真正想要的价值。

在GitHub，用户可以十分轻易地找到海量的开源代码。因此GitHub被誉为程序员或IT人士的三宝之一。