案例 1：老塔科马大桥——建筑工程史上失败的设计。老塔科马大桥是位于美国华盛顿州的塔科马海峡上的第一座大桥，绰号舞动的格蒂，于 1940 年 7 月 1 日通车，它被当时的媒体和桥梁行业美喻为“人类鉴定不移的独创净胜的结晶”，但四个月后戏剧性地被微风摧毁。当人们从倒塌事件中回过神后，工程设计人员开始分析大桥被风“吹”断的原因。然而，大桥的施工质量是无可挑剔的：由坚硬的碳钢和混凝土建成，整个施工过程被严格监督。桥梁界最后给出的正式结论是震撼而有深远借鉴意义的：“塔科马大桥使用了崭新而没有经过验证的桥梁设计结构，使大桥建成后毁于风力造成的风振。”从此，老塔科马大桥成为几乎每一位桥梁建筑设计人员都知道的最著名的工程设计失败案例。

案例 2：CONFIRM 项目——信息化经典失败案例。CONFIRM 是希尔顿饭店和其他两家大公司的一个雄心勃勃的软件开发项目，这个系统被认为是最先进的综合了旅游、住宿和出租车工业的预订系统。1987 年 3 月启动，开发了 3 年半，耗资 125 万美元，最终被取消了。

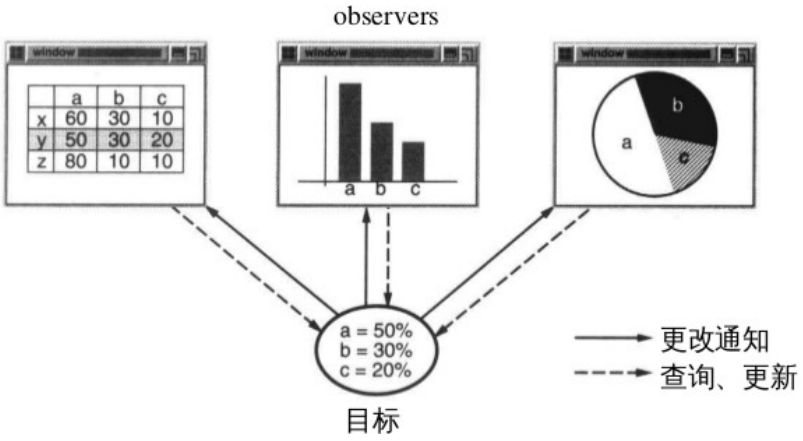
美国航空公司子公司 AMRIS 被授权管理和开发 CONFIRM 系统。协议使这

个公司负责设计与开发新系统的所有方面，协议规定了两个阶段：设计阶段和开发阶段。设计阶段需花费 7 个月时间，而开发阶段准备在协议签署后的 45 个月内完成。因此，项目的最后期限在 1992 年 6 月底。协约提供的开发 CONFIRM 的整个花费不超过 557 万美元。系统开发者保证说开发花费没有理由超过此数目 1988 年 12 月 30 日，系统开发者呈交了系统的基本设计书，但大部分合伙人都认为功能设计得不完全。1989 年 3 月，系统设计者声称功能和技术上的详细说明书完成了。该月底，公司公布了一个初级发展计划，这个计划没有被投资者接受接下来的 6 个月不得不致力于计划的修订。系统设计者于 1989 年 9 月完成了设计阶段并公布了一个可行性报告供投资者审阅；同时，系统设计者把项目的价格从 557 万美元提高到 726 万美元。1989 年 10 月 16 日，系统设计者向投资者保证项目能按时完成，并不会超过预算。但是，1990 年 1 月，系统设计者没有按时完成终端荧光屏设计。1990 年 2 月，又没有按时完成商业区域分区阶段的重大事件设计。明显地，开发者将把这阶段未完成的工作放到下个阶段去做。…..　3 年半后的 1992 年 7 月，在对该项目花费了 125 万美元后解体了。开发者专业上的主要难题是将 CONFIRM 预订系统与决策系统联系起来。但设计者承认：我们发

现它们是无法统一的， 而且，后来发现数据库在冲突事件中也无法补救。

案例 3：Atlas——设计要求先多样化再聚合。多样化是指要获取多种方案和设计的原始自理，聚合是指从多样化汇聚的信息中挑选满足需求的元素。Atlas 是由奇虎 360 公司 Web 平台部基础架构团队开发维护的一个基于 MySQL 协议的数据中间层项目。它在 MySQL 官方推出的 MySQL-Proxy 0.8.2 版本的基础上，修改了大量 bug，添加了很多功能特性。目前该项目在 360 公司内部得到了广泛应用，很多 MySQL 业务已经接入了 Atlas 平台，每天承载的读写请求数达几十亿条。其成功的要点有很多，如深入了解业务的应用场景、以及应用程序员与 DBA 面临的各种难题，提出有价值的需求；目标不是 100%模拟 MySQL，而是只针对那些通用且有用的功能，如果某个特性只有很少人使用且开发代价不菲，会果断舍弃，将开发资源投入到其他方面；开发软件尤其是基础服务时，眼界要开阔。在开发 Atlas 的过程中，有必要了解 LVS、Zookeeper 等其他开源软件的实现原理，对同类产品做广泛的调研，吸取其他产品的优点，避开其缺点，这样才能将自己的产品做好。

案例 4：观察者模式。我们将一个系统分割成一系列相互协作的类有一个常见的副作用：需要维护相关对象间的一致性。但是，我们不希望为了维持一致性而使各类紧密耦合，因为这样降低了它们的可重用性。例如，许多图形用户界面工具箱将用户应用的界面表示与底下的应用数据分离。定义应用数据的类和负责界面表示的类可以各自独立地复用。一个表格对象和一个柱状图对象可使用不同的表示形式描述同一个应用数据对象的信息。表格对象和柱状图对象相互并不知道对方的存在，这样可以根据需要单独复用表格或柱状图。然而，它们又似乎相互知道。也就是说当用户改变表格中的信息时，柱状图能立即反映这一变化，反过来也是如此。



这一行为意味着表格对象和柱状图对象都依赖于数据对象，因此数据对象的任

何状态改变都应立即通知它们。同时也没有理由将依赖于该数据对象的对象数目限定为两个，对相同的数据可以有任意数目的不同用户界面。

观察者模式正好描述了如何建立这种关系。该模式定义对象间的一种一对多的依赖关系，当一个对象（目标）的状态发生改变时，所有依赖于它的对象（观察者）都得到通知并被自动更新。

案例 5：MFC 应用框架。在现在的应用系统中，图形用户界面（GUI）的使用非常普遍，例如：下拉菜单、帮助屏幕、对象拖放以及文件打开、存储、打印、查找、排序和简单的文本编辑，是几乎所有系统都需要用到的，而且其基本用法已经延续了十几年，已经形成习惯。微软把涉及文件、数据库操作、网络 I/O、图形、通用数据结构、用户界面对象等应用功能组件，组合在一起，构架了一个应用框架这就是 MFC（Microsoft Foundation Classes）。在应用层，这些功能是常用的、应用特性是相似的。在操作系统的服务层，这些功能的实现是相同的。程序员只要理解这个结构，就可以重用大量的类库程序，只需要添加或修改少量的代码，就可以快速地构建自己的应用系统，而不需要自己去编写复杂的与底层（操作

系统、外部设备、网络等）打交道的代码。