**多年的**[**测试**](javascript:;)**经验中，经常发现有这么一种现象：总有些提了的bug不能顺利的被修复。这些bug往往有4个走向：**

　　1.在被发现的版本中最终被解决，但中途花费较多周折。

　　2.有计划的在后续的版本中被解决。

　　3.决定永远不修复，却变成潜在的炸弹，在后续版本中被迫修复。

　　4.决定永远不修复，至今为止也一直没有被修复。

　　之前对我们做过的项目做过一次较大的统计，统计严重程度中等及以上的缺陷，这四种走向第一种占到了50%左右，第二、三种各占20%，最后一种约占了10%。

**这些没有被修改的bug带来的负面影响有：**

　　1.大部分时候最终还得改了，是被迫改，项目组疲惫，在领导和客户那里都落不了好。

　　2.这些bug积累到一定数量，发现系统快不能要了，得大规模重构，重构的过程不要太痛苦，最后没准就推倒重来了（见过n个这样这样的案例了）。

　　3.拖得越久改起来越难，之前出现过的一个案例是：某项目为了赶进度，使用了一个较低版本的底层组件，当时识别出低版本的底层组件特性有缺失，测试人员提出了功能bug，项目组决定忍了。一拖就是2年。结果项目很成功，越来越重要，与之交互的其它系统越来越多，但这个底层组件缺失特性的短板就越来越痛。最后不得不进行修复工作（高版本组件替换），但发现由于代码耦合太紧，已经不是一个月两个月能搞定的事情了。大规模重构还是推到重来现在成了一个难题。

　　4.每天跟带着太多毛病的系统朝夕相对，是杀死所有干系人士气的慢性毒药。当你的潜意识认为你在做的东西是一团shit，还有毛激情？想一想破窗效应马上能够反应过来。

**怎样降低大量bug长期遗留的现象呢？我有如下的一些建议：**

　　1.提升内建质量。这句话高大上，内涵也很丰富，从软件架构，开发过程，各种技术应用等各方面都能够找到无数的提升点避免系统存在太多遗留bug，展开说真的要一本书了。从里边抽取出最重要的一条精神：bug被发现的越早，修改遇到的阻力越小。

　　2.定期bug扫除，这其实是测试应该主动提出来的事情，并且应该让这件事儿变成项目组的例行活动。其实如果做好了，乐趣还是很多的，效果也非常好。

　　3.如果是大型系统，或者项目群，很多bug是跨项目组的，这时候组织级的机制就要建立起来了，必要的时候需要跟考核制度挂钩。这样有一些三不管的重要bug才能被最终解决。

　　4.有些bug还真得睁一只眼闭一只眼了，约有10%的顽疾会这样。难改，影响范围有限。对这类bug最有效的办法是：挖雷难，我给它上边插个旗子让使用者离他远点儿好不好？有时候处理这些bug挺艺术的，运维，客服，售前，售后，都得长点儿心眼。