Bug 管理的经验和实践（上）

孟岩：刘振飞，你好。我知道你以前是方正出版印刷系统的核心开发人员，后来来到微软的 Office 开发组。我认识你的时候你还在微软工作，状态似乎不错。为什么后来又离开微软了呢？

刘振飞：93 年到 96 年，我在北大计算机研究所读研。96 年毕业后，我留在所里继续从事方正核心产品世纪 RIP --- PSPNT 的研发、维护、升级（还有外围软件开发比如新女娲补字 NewNW、PDF 流程系统等）。PSPNT 是国内比较大的、成功的软件产品，我一直为曾参与研发过这个产品而自豪。

工作中，我模模糊糊地觉得应该有一个清晰的、可控的流程，而不是依靠几个开发 高手 来支撑一个项目。方正人的个人素质非常优秀，集中在一起应该做得更为出色。我在方正的时候最多也就带过 10 来人的队伍，但即使这么小的团队，已经让我精疲力竭、疲于应付了。我发现面临一个无法解决的难题：如何有效地控制软件研发流程以保证产品质量和进度。我意识到做好一个软件，只靠技术好是很不够的，必须要有一套好的研发流程和配套的支持工具。你也知道国内软件企业的项目经理都是 全才 ：需求、设计、编码、测试、维护乃至产品发布都要精通，事必躬亲，但实践中你又不可能样样都精通，所以结果只有一个：四处救火，累得半死但永远看不到尽头。

当时就觉得这么干有问题，但究竟问题出在哪里、如何有效改进都不知道。我最纳闷的是：这么 10 来号人的研发管理就这么费劲，人家微软上千人、分布全球的 Windows、Office 研发队伍是怎么有效管理的？我当时深入研究了一些软件工程方面的理论，比如花了一段时间仔细阅读了原版的 Rational Unified Process（RUP），觉得很兴奋，RUP 里面的研发理论很完备，和几个同事聊天觉得应该按照 RUP 的去做，但是理论归理论，具体到实际产品开发该如何做，还是一筹莫展。

恰好那时候读了微软（中国）公司前总经理吴士宏的畅销书《逆风飞飏》，提到了微软的企业管理、内部的数字神经系统及相关叙述，非常感兴趣，想去那里看看。刚好有个机会，得到了微软（中国）研发中心 Office 组的一个 PM（Program Manager）职位。在微软的 4 年中，我先后经历过 Office XP、Office 2003 的研发，中间还夹着做过 Project 2002。微软所有产品的研发都遵循同样的研发模式、使用同样的研发工具来进行管理，只不过产品大小不一、人员配置有点区别罢了。经历这几个大产品的研发流程，加上在方正的体验的对比总结，我觉得自己比较深入的理解了微软做研发的 套路 。

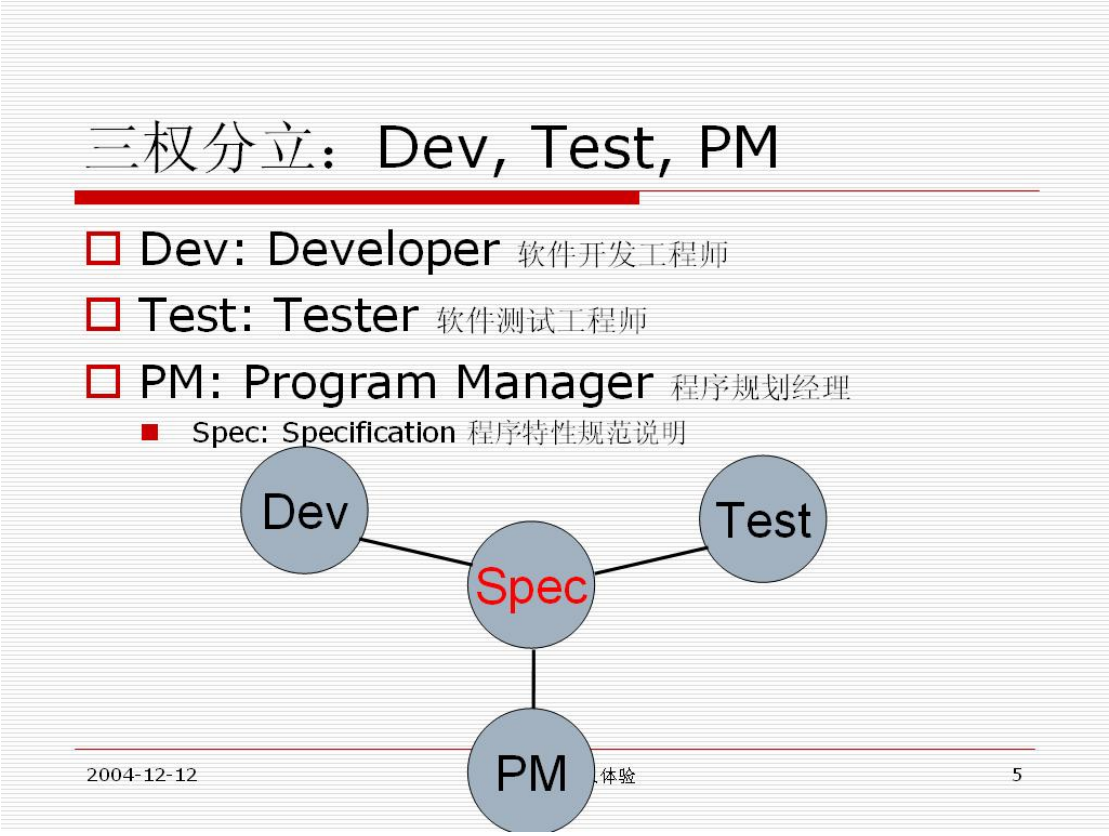
我是 C++程序员出身，当 PM 后就没怎么写过代码，总还想写写。刚好几个朋友开的公司要做网站、短信、声讯，还要对报纸做数据支持，他们需要一个懂研发管理的人去带技术部。我觉得已经熟悉了微软的研发流程，这刚好是一个检验自己所学所思的机会，所以就离开微软去做这家小公司的技术总监了（而且满足我另外的愿望：我对 Windows 之外的世界充满好奇，比如每天去新浪网看新闻，他们网站是如何做出来的、用到什么样的技术？Linux、开源软件到底怎么回事？）

不过我一直认为微软是一家伟大的公司，很喜欢其工作氛围。特别的，微软的软件研发流程我认为是最先进的，值得大家去学习。

孟岩：那请你介绍一下你所体会的微软研发管理的妙处。

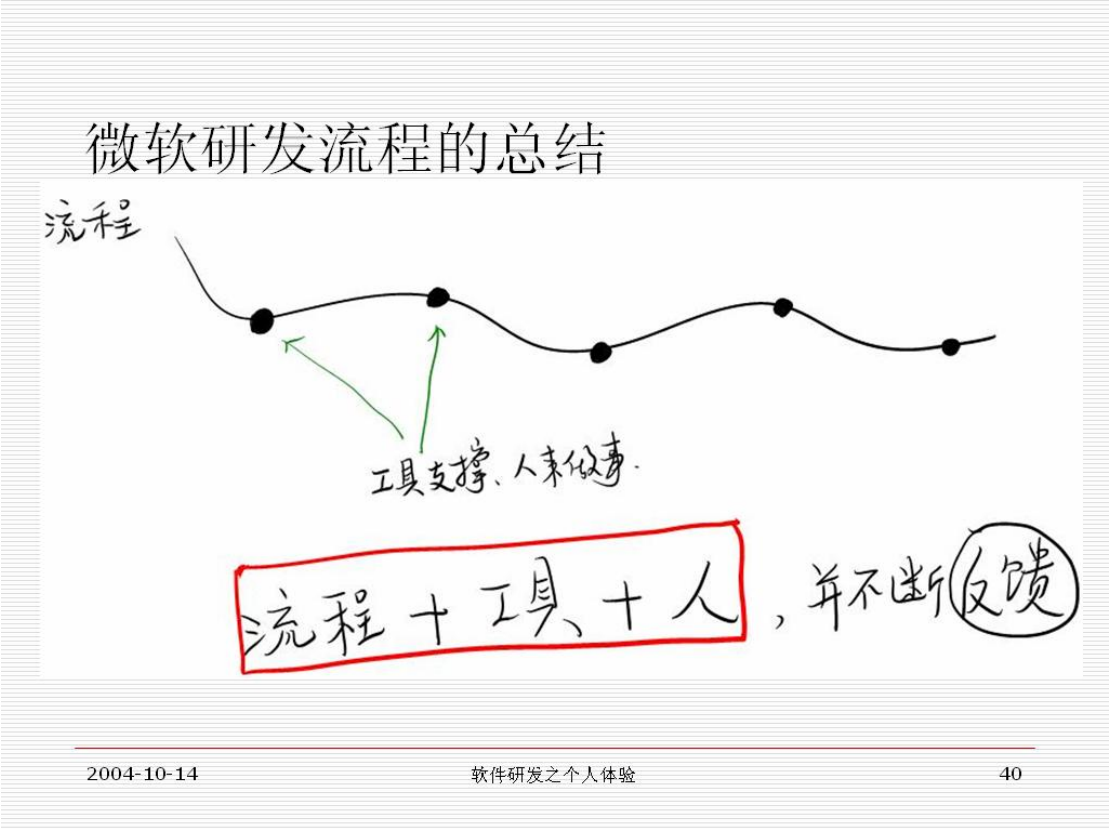
刘振飞：从我理解的角度，微软的研发管理可以从以下几个方面描述：

（1）研发人员分工明确。主要的三个角色：PM (Program Manager)、 Dev (Developer)、Tester 三者分工明确、接口清晰，PM 来定义需求、书写出来每个功能特性 (Feature)的设计文档 (Spec)，Dev 写代码来实现这个 Spec，Tester 来测试 Dev 做出来的东西是否符合 PM 定义的 Spec，三个角色之间并无必然的上下级关系，只是分工合作完成某个功能(Feature)。我将之形容为 三权分立 ，三者之间有效合作并制衡。国内企业好像还没有 PM 这个角色，而测试人员又往往成为开发人员的附庸，一个 Bug 是否要被解决全由开发人员说了算，这很糟糕，就像政治上一个权力没有被有效的制衡一样，一定会产生各种问题。



（2）研发工具很配套。PM 将写好的需求设计文档(Spec)保存到 SharePoint 文档库中，所有相关的人都可以随时查看；Dev 用 Source Depot （功能类似 CVS 的微软内部源代码管理工具）来保存源程序；Tester 把发现的 Bug 记录到 Raid 中以有效跟踪这个问题的处理流程。

（3）分阶段的研发流程。和任何软件公司一样，微软的研发无非也分为规划、开发、测试、发布等几个阶段。但是微软的研发流程不走形式，可以统一产品组所有员工的思想，并且能够有效地控制住进度。做完一个版本后，还会让所有员工匿名投票，找出这次研发过程中出现的各种问题以便在下个版本中解决 (此过程称为 Postmortem，挺吓人的一个词)。



可以这么比喻，微软这套研发模式让其中的每个人都成了一架高速运转的机器上的各种零件，少数零件坏了不要紧，可以随时更换。当然微软有许许多多技术高手，但我认为更重要是其研发模式保证了软件产品的高品质、可持续发展。

孟岩：提到微软的研发管理，你说过一句话，我印象很深。你说微软的研发管理中，它的 bug 管理系统是居于核心地位的。你这么说，有什么道理吗？

刘振飞：前面说过，微软所有产品的研发都遵循同样的研发模式、使用同样的研发工具来进行管理。在所有的工具中，我最佩服的就是其 Bug 管理系统 Raid（现在叫 Product Studio）。可以说，遍布全球的微软研发人员能够保持统一的思维模式、做事及语言习惯，与整个研发流程的配套工具密不可分，其中最重要的就是通过 Raid 把整个产品的研发有机的联系起来。阅读每个 Bug，你可以详细的看到大家讨论解决该问题的完整思路。

我曾读过微软 Project 2002 产品的 Architect 写的一个备忘录，其中提到： 如果 Raid 是别家的产品，需要微软每年付出一笔巨大的费用，BillGates会支付这笔钱吗？ He wouldn t be happy, but you bet he would. Microsoft depends on Raid to get the job done. 当时我心有戚戚焉 ，立即给这哥儿们发 Email 表示赞同之意J 他回信说希望 Project 能够做的像 Raid 一样成功，但可惜他要离开微软自己开公司了。

在微软上班，我每天第一件事是打开 Outlook 来处理有关自己的重要邮件，第二件事就是打开 Raid 来看看有关自己的 Bug 情况，赶快处理。我一直纳闷，微软为什么不把这个 Bug 管理系统作为软件来出售，那可是任何一家软件企业都需要的啊！

表面上看 Raid 其实是一个简单的工具，那么它的重要性体现在什么地方呢？

- Raid 从一开始就支持多用户
- 一个团队中的所有人都可以创建、指派 Bug，或者改变 Bug 状态
- 用户可以非常自由的去定制 Raid

l 基于 SQL，很多有用的报告可以被生成出来
l 管理层需要所有 Bug 都在 Raid 中被有效的跟踪处理

Raid 的价值在于它密切跟踪当前产品的实际 Bug 状态，使项目组中的成员非常有效的协调他们的工作。大家都很聪明，如果一个工具是容易理解的、而且管理层提供其使用指南（比如 Bug 怎么被指派和解决），那么简单的工具也能提供巨大的价值。

孟岩：能否请你简要地介绍一下微软的 bug 管理体制？
刘振飞：在整个产品的研发过程中，特别是在测试产品、修复 Bug 的中后期，团队中所有人都生活在 Raid 中：

- 测试人员（Tester）只要发现问题就立即新建一个 Bug 予以跟踪并指派给相关的开发小组长（Dev Lead）
- 开发小组长会判断这个 Bug 属于某个特定的开发人员（Dev）并指派给他处理
- 开发人员会根据 Bug 的详细描述信息找到问题所在，修改程序解决这个 Bug 并把 Bug 返回给当初的测试人员；或者有争议的时候，把 Bug 指派给这个 Feature 的定义者 PM，要求一个澄清说明
- 测试人员在看到某个 Bug 被解决后，就去验证这个 Bug 是否真的不存在了，根据最初的发现步骤去证实问题真的解决了就关闭这个 Bug；若还能重现，或者不同意开发人员的解法，可以激活这个 Bug，返还给当初的开发人员做进一步调查处理
- 当测试人员和开发人员无法达成一致意见的时候，由对应的 PM 出面做协调，判断这个 Bug 的严重程度、对用户可能的影响，根据产品的进度和项目资源做出评估，是否真的需要修理掉这个问题
- 管理团队利用 Raid 来跟踪整个进度：单个人的工作、小组的进度，整个产品研发进度

研发队伍中的所有人都通过 Raid 来商议、沟通某个 Bug 是否符合当前解决 Bug 的 门槛 ，决定是否需要真正修理掉这个 Bug、如何修理、可能的副作用、如何测试其解决方案等等。每个人可以在 Raid 中看到某个 Bug 的全部历史档案，比如几年前发现的一个 Bug 一直推迟到这一版才解决，前几年大家是如何讨论的，可能的处理思路是什么，都被完整地记录下来了。

每月、每周甚至每天，参与这个产品研发的人都收到一封当前 Bug 状态的 Email：每个人都上有多少 Bug，Dev、PM、Tester 头上 Bug 数最多的前 5 名都是谁，哪个子产品、子模块中的问题还是处于上升阶段，整个 Bug 的趋势怎么样等等。这是最详尽的产品状况 内参 ，暴露在团队中每个成员的面前，一览无遗。只要你的名字被列在 Email 中，你就非常紧张，因为你脑袋上的 Bug 比较多、影响整个产品的质量。这些该死的 Bug 等待着你去快速处理，尽快把自己从排行榜上去掉。每解掉一个 Bug，或者把 Bug 转给另外的人去处理，就会舒一口气，因为头上又少了一个；某一天你头上的 Bug 数降为 0 了，心里就会非常高兴J

我觉得微软的 Bug 处理过程，非常类似于 击鼓传花 的游戏。鼓点响起，你的任务就是尽快把自己手中的 花 （Bug）传给下一个人，不要让它在自己手里耽误很长时间。从表面上看，在微软工作从不打卡、上班时间也很自由、上午很晚到办公室也没人管你，但是有 Email 跟着、有 Bug 催着，你永远不可能偷懒。没有人盯着你，只是事情如影随形，而且所有和你相关的事情都是公开的，相关的人都知道，就像处于非常开放的舆论监督之中，除了把事情办漂亮你还能有别的选择吗？

最后要强调两点：

（1） 上 Bug 不仅仅是测试人员的事情，团队中的每个人发现问题时都上个 Bug 来跟踪；
（2） Raid 中不仅仅是跟踪软件功能方面的 Bug，其他各种问题如需求文档（Spec）的改

动、界面上的错别字、帮助文档的遣词造句问题、某项任务指派等等都可以通过一个 Bug 来跟踪。

我至今对刚进微软时老板的一句话印象深刻：Everything should be tracked in Raid!

孟岩：就你了解到的情况，国内的公司在这方面怎么样？

刘振飞：在微软这几年，我也一直和国内软件公司的朋友们保持接触。很遗憾的是，国内的一些软件企业，特别是不少中小企业，其软件研发还是处于作坊式的状态，只不过作坊规模有大中小之分罢了。有意思的是，走在国内 IT 最前沿做各类网站的企业，根据我的了解，也在走软件企业最初几个 大虾 （牛人）搞定一切的阶段。我不是说个人技术好不重要，而是需要更进一步，把研发管理真正搞起来，做出规模效应，从而有效的保证质量、控制进度、把对某个人的依赖尽量降低，并使产品可持续发展。

你知道国内不少软件企业在做 ISO9001 或 CMM 认证，花费不菲。少数企业纯粹是为了认证而认证，对付着拿到证书就达到目的了；更多的企业确实是想利用这个认证的过程，把自己的研发流程规范化。但似乎能从这些认证中享受到真正的研发管理提升的并不是很多，甚至开发人员现在需要花费大量的时间去书写一些例行公事的、没有任何实际价值的格式化文档，苦不堪言。

我觉得软件研发管理必须结合自己企业的实际情况，不要生搬硬套书本上的理论，只要人员分工、配置合理，能够控制质量，什么方法都可以采用。黑猫白猫，能抓耗子的就是好猫。千万不要走形式、走过场。

孟岩：其实国内公司在研发方面与微软的差距非常大，也存在于很多方面。为什么你独独看中 bug 管理？为什么你认为中国中小型企业的软件开发管理规范化，应当从 bug 管理入手呢？

刘振飞：从微软的研发管理来看主要是需求、开发、测试这三大块，毫无疑问国内公司在开发这个环节一直都很重视，不过需求和测试较弱一点。大家现在都已经认识到充分理解业务需求的重要性，如果没有很好的对项目或产品用户需求的真正把握，后面所有的工作都将是白费工夫、事倍功半，这一块缺乏的是如何有效地将用户的需求转成一份份详细的、后面开放测试人员可以理解的文档。

但是测试这一块大家还是不够重视，对测试人员的素质要求也不是很高、人员比例也较低，测试人员往往依附于开发人员的直接管理，人微言轻。而在微软，测试人员和开发人员的比例很多时候是 1 比 1 的，有时候会更高。测试人员和开发、需求人员一样有自己单独的行政管理路线，比如我就注意到有非常资深的测试人员可以做 VP，专门管理某个产品的测试工作。这在国内企业来说，几乎就是不可能的。

通过前面的介绍，无论人员的配置和工具的提供，你可以看出微软是非常重视测试的。所以我觉得如果我们学习微软先进的研发理念，可以从测试入手，打造必要的测试管理系统，通过这样的系统，把需求、开发、测试三个环节融合在一起，让团队中所有的人都遵循同样的研发思路：需求人员真正理解用户的业务需要，认真研究后形成需求文档作为产品一个个功能的 合同 ；开发人员写出设计文档，然后动手写代码；测试人员理解需求文档，然后测试做出来的功能是否符合最初的设想。整个过程碰到的任何问题都要通过 Bug 系统来记录、跟踪，相关人员通过 Bug 管理来商谈、沟通相应的解决方案。

这样通过 Bug 管理，逐步统一研发人员的思维、做事模式，让整个团队可以有效地运转起来，并不断优化自己项目的软件研发流程，提高产品质量，从而使产品可持续发展。

孟岩：看来你对于 bug 管理可真是重视。听说你在离开微软后，开发了一个开源项目 BugFree，

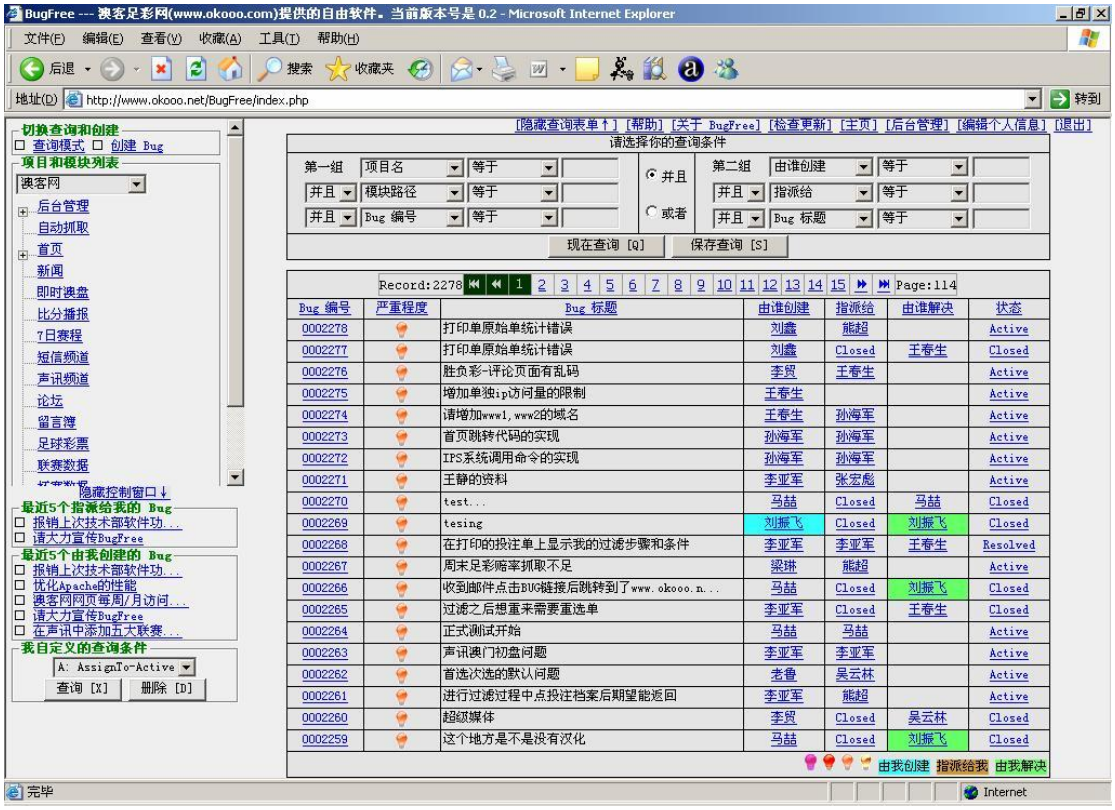
号称是要全面模拟微软的 bug 管理机制。能介绍一下大致的情况吗？

刘振飞：今年四月我加盟朋友的公司开始做网站，发觉自己已经习惯了微软的研发模式，于是建议这几个朋友先做一个　数字神经系统 ，其目的是让一切可以数字化、文档化的信息被记录下来，为公司的进一步发展和决策提供基础信息支持。

BugFree 就是其中有关软件研发的 Bug 管理系统部分。我太了解 Bug 管理对软件研发的重要性、也对微软的 Bug 系统有了深入掌握，所以需要有一个类似微软的 Bug 系统才好开展工作。虽然网上有免费的 Bug 管理系统（如 Mantis、Bugzilla），但是我看后觉得都不好使，和我在微软用的差别太大，上海科泰世纪公司的 Bug 管理系统倒也很像微软的，但是要花钱买。琢磨着反正我也这一块非常熟悉了，何不做一个出来自己用？于是决定借鉴微软的研发流程和 Bug 管理工具自己开发一个，以便对我们开发新网站、声讯软件、客户端软件和公司事务管理中出现的问题进行有效的跟踪处理。

数字神经系统　中的 BugFree 是用开放源代码的 PHP+MySQL 写成、基于浏览器方式运行的。我以前没有任何 Linux+Apache+MySQL+PHP 的开发经验，但我很幸运的招聘到几名优秀的 Web 程序员，可以在短短的两个月时间内搭建起这样的系统。其中 BugFree 是由我的同事王春生开发的，他用了不到一个月的时间就把代码写完，让我很是惊讶，从而认识到基于 Linux 的 Web 开发魅力。之后我们测试一个多月，就可以在实际工作中使用并不断完善。现在 BugFree 已经成了我们日常工作最重要的工具，每个员工也都习惯用 Bug 来记录跟踪事情，不仅仅是代码中的缺陷可以上 Bug，新的需求、设计变化等都可以用这个 Bug 管理系统有效的管理起来。其实 Bug 不仅仅可以用来记录软件中的缺陷，也可以用来跟踪公司的日常事务。比如在公司的网上报销系统还没有建立之前，我们就用 BugFree 来处理报销的事情。甚至，一个同事给我上了这样的 Bug：你的桌面太乱了，请整理一下J

命名 BugFree 有两层意思：一是希望软件中的缺陷越来越少直到没有，Free 嘛；二是表示它是免费且开放源代码的，大家可以自由使用传播。也算为中国的软件业做点小小的贡献，特别的，希望能对国内中小企业的研发流程改进、Bug 的有效管理提供参考和帮助。



和微软内部的 Raid 比较起来，BugFree 有如下特点：

（1）Raid 是 Windows 客户端软件，BugFree 是基于浏览器的。基于此，Raid 有很强大的编辑展示功能，而 BugFree 简单、方便、易用；

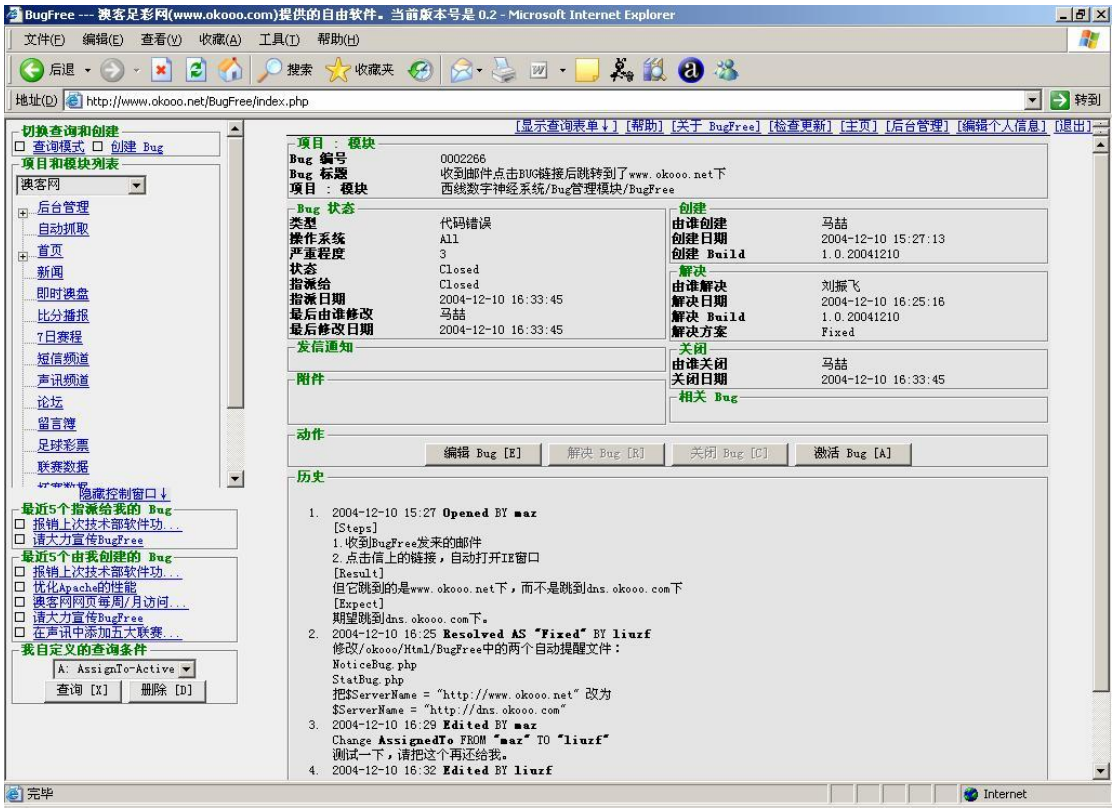
（2）Raid 可以进行极其复杂的组合查询，BugFree 的查询功能相对弱一些，但我觉得对中小企业已经够用了；

（3）一个 Bug 从创建到关闭这个 生命周期 的处理过程，BugFree 全面借鉴 Raid 的处理流程，处理方法甚至一些词汇都和 Raid 一样 (所以我现在用 BugFree 处理 Bug 的感觉和在微软时候基本一样)；

（4）BugFree 还有一个独创的功能：当一个 Bug 被指派给你的时候，系统会自动给你发一封邮件，告诉你有个 Bug 需要你处理，这样结合 Email，BugFree 被完美使用起来，成为我们现在网站开发、运行、维护必备的工具。我们还增加了两个 Bug 统计功能：一是每天早上 8 点钟每个同事都会收到一封 Email，告诉他/她头上还有多少 Bug 等待处理；二是每周一中午给所有人发一封邮件，告知上周 Bug 的处理情况和到目前为止所有 Bug 的统计数据；

（5）BugFree 程序规模很小，一个中等水平的 PHP 程序员就可以在 1~2 周内看懂所有的代码，然后就可以根据自己的需要做相应的定制了；

（6）最最重要是，BugFree 是免费并且开发源代码的。你可以体验到微软的 Bug 管理精髓，按自己需要自由地增加功能、修改代码而不用担心版权问题J



不过坦率的讲，BugFree 仅仅是个工具而已，重要的是掌握其中蕴含的软件研发的流程思想，才能用好这个工具。如果你以前没有用过 Bug 管理系统，那么一开始的时候也许你会觉得这个工具是在浪费时间，因为一个测试人员需要费神把发现 Bug 的详细步骤记录下来，有时还要贴一张示意图，这一切都不如当面说来得直接。

但是使用一段时间，你会发现 BugFree 很有用，它忠实的记录着每个问题的处理过程，不断提醒你存在的问题，永远不会丢失和忘记。如果你参与过较大软件项目或产品的研发，

就会理解它对软件可持续发展是至关重要的。而且研发的规模越大，BugFree 的作用就会越大。

感兴趣的朋友，可以到 http://www.okooo.com/OpenSource 来体验、下载最新版的 BugFree。

孟岩：好的，我们下期结合 BugFree 来具体看看一个软件开发项目的 bug 管理应该怎么做。