

软件测试—《软件开发的科学和艺术》节选

撰文/陈宏刚

《软件开发的科学与艺术》是电子工业出版社联袂微软公司华人专家于近期推出的一本优秀之作。书中凝聚了微软公司多位专家多年研究与工作的宝贵经验，并通过对许多成功或失败案例的中肯剖析，为读者展现了软件开发的思想和流程，值得软件人员好好阅读和领悟！

一、微软的测试人员

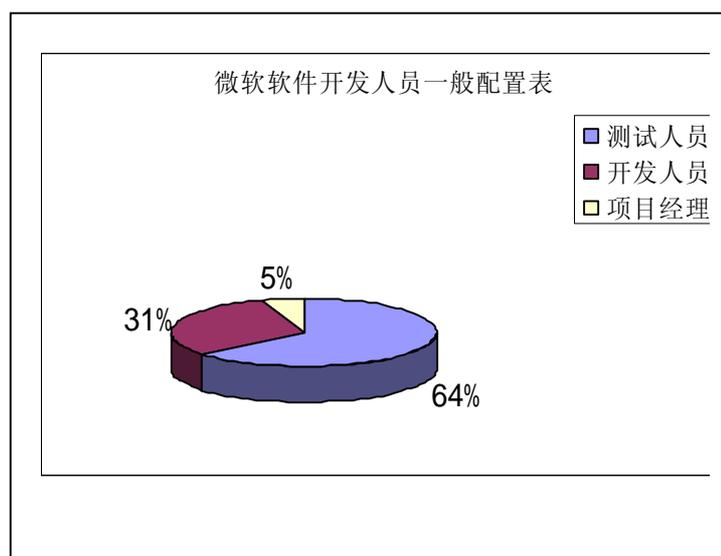
微软的软件测试人员分为两类：测试工具软件开发工程师（Software Development Engineer in Test, 简称 SDE/T）和软件测试工程师（Software Test Engineer, 简称 STE）。

测试工具软件开发工程师：负责写测试工具代码，并利用测试工具对软件进行测试；或者开发测试工具为软件测试工程师服务。产品开发后的性能测试（Performance Test）、提交测试（Check-in Test）等过程，都有可能要用到 SDE/T 开发的测试工具。由于 SDE/T 和 SDE 的工作都是写代码，具有相通的地方，所以两者之间互相转换的情况比较多。但需要注意的是，两者写出来的代码用途是不一样的，SDE 写的是产品的代码，而 SDE/T 写的代码只用于测试产品。

软件测试工程师：负责理解产品的功能要求，然后对其进行测试，检查软件有没有错误（Bug），决定软件是否具有稳定性（Robustness），并写出相应的测试规范和测试用例。除此之外，在一个软件产品的研发和销售过程中，还会需要负责给产品打补丁（Service Pack）的快速修正工程师（Quick Fix Engineer），通常由 SDE 来担任，通过电话方式向用户提供售后技术支持的支持工程师（Support Engineer），销售和市场营销（Sales and Marketing）人员，研究员和研究工程师（Researchers & Research SDE）。

在进行产品开发的时候，主要是由前面三类人员（项目经理、开发人员及测试人员）组成产品开发团队来进行的。

在微软内部，软件测试人员与软件开发人员的比率一般为 1.5-2.5 左右，这可能远远超出了大家对测试人员的理解，但微软软件开发的实践过程已经证明了这种人员结构的合理性。下图中显示了上述两个产品的微软软件开发人员的一般配置图。



下面以微软 Exchange2000 和 Windows2000 为例介绍一下微软产品团队的人员结构(这里只分析三类主要的人员,即项目经理、开发人员及测试人员),如下表所示。

	Exchange 2000	Windows 2000
项目经理	25 人	约 250 人
开发人员	140 人	约 1700 人
测试人员	350 人	约 3200 人
测试人员/开发人员	2.5	1.9

二、测试时应考虑的几个问题

(1) 测试最重要的一件事就是要考虑到所有的出错可能性。同时,还要做一些不是按常规做的、非常奇怪的事。

说起来可能不太好听,测试的过程就像黑客(Hacker)的攻击过程那样。可以这么说,像黑客这样的人是最好的软件安全测试员。他们专门找软件的漏洞,从而破坏这个软件,这样就可以修复这些漏洞来保证软件的性能。如果找不到这种漏洞,那就说明该软件质量已经很好了。

(2) 除了漏洞之外,测试还应该考虑性能(Performance)问题,也就是一定要保证软件运行得很好,非常快,没有内存泄漏,不会出现那种越来越慢的情况。

我们可以在不关机的情况下,与其他软件一起持续运行一个多月,看看是否会出现越来越慢的情况(当然必须保证其他软件是没有问题的)。我们在做 IE 的时候,就是让它 72 小时连续不停地打开不同的网页,处理几万个不同的网页,而且速度不能减慢。有许多软件,当只有一两个人用的时候,可能感觉不到什么问题,而当几百个用户一起用的时候,有的网站就出现各种各样的异常,这就是测试工作还比较欠缺的缘故。

(3) 另外,测试还要考虑软件的兼容性(Compatibility)。

一般来说,一个软件是由许多小软件构成的,如果其中一个小软件与它的前一版本不兼容,那么这个软件就会出现错误。这种错误需要通过测试来发现和解决。

许多人认为写代码的人一定能找出错误来。其实开发人员在写代码的时候,如果有错误,他可以意识到了,可是写出来的错误,他不一定能想得到。我自己也编过程序,在编程序的时候很自信,觉得不会有错,可事实上,即使是我编的小程序也有错误,但要自己找出来,却要费很大劲。因为我一直认为自己不应该出错,但常常错误就出现在我认为最有把握的地方。我是学数学的,是一个很细心的人,可是--样还是会出错,但要找出自己的错误却要花费很长的时间。后来我写的代码让我的师弟帮我看,结果他很快就找到许多问题,可是我自己花一个月时间可能都找不到。所以,开发人员和测试人员完全不一样,

开发人员确实可以找到一些 Bug，但是有更多的 Bug 是他意识不到的。

在一般的开发团队中，并不需要测试人员定位 Bug 的具体位置，所以，对测试人员的要求并不高。只要你愿意学，测试工作是非常容易做的。但是，我当年所在的 IE 团队(IE 4.0)就不同，因为当时正在与另一个公司的产品竞争，所以微软就要求尽量找到一流的开发人员和一流的测试人员，尽快开发出新产品，打败对手。所以，当时对我们测试人员的要求非常严格，不仅要找出 Bug，而且要定位引起此 Bug 的代码行。然后将这些信息交给开发人员，后者就可以很快更正，省去了他们找错误出处的时间。因此，当时 IE 的开发速度非常快，一年之内就发布了一个新版本，而且几乎没有任何大 Bug，大大超越了竞争对手。

二、关于 Bug

Bug 的定义很广泛，在软件使用过程中所出现的任何一个可疑问题，或者导致软件不能符合设计要求或满足消费者需要的问题都是 Bug，即使这个 Bug 在实践中是可行的。

有时候，Bug 并不是程序错误。例如，软件没有按照一般用户的使用习惯来运行，此时也可以把这个问题看成是该软件的一个 Bug。

通常，对 Bug 的跟踪过程如图所示。



首先，测试人员根据测试结果来报告他发现的所有 Bug。通常，这项工作还需要用户的参与。微软在正式发布一个软件之前，经常要依次发布 Alpha 测试版、Beta 测试版让用户试用，以使用户能够反馈出相关 Bug 的信息。但是，现在随着微软测试队伍的扩大及测试水平的提高，越来越多的 Bug 都是我们内部的测试人员自己发现的，很少出现外部用户所反馈的 Bug 没有被测试人员发现的情况。

然后，开发经理根据这些 Bug 的危害性对它们进行排序，确定 Bug 的优先级，并安排给相关的开发工程师。

接着，开发人员根据 Bug 的轻重缓急依次修复各个 Bug。

最后，测试人员再对开发人员已经修复的 Bug 进行验证，确认 Bug 是否已经被彻底更正。

微软开发一个产品经常会遇到几十万条 Bug。随着测试人员越来越多，测试工作也越来越完善。但是，如何快速有效地追踪并修复 Bug，仍然是摆在软件测试人员面前的一个主要困难。测试产品和追踪 Bug 时最重要的是问题的定义，要清楚需要解决什么样的问题，确定 Bug 的主次关系，挑选出最主要的问题，并先解决它们。例如，有些 Bug 可能会导致死机或程序崩溃，这时就要先修复它们，而另一些 Bug 可能对软件的运行影响不大，或者出现的机会很少，就可以考虑以后再修复。

可以说，没有任何一个产品没有 Bug，也永远不可能找出并修复所有的 Bug。在修复了旧的 Bug 的同时，往往又会生新的 Bug。

根据微软的经验，每修复三到四个 Bug 一般就会产生一个新的 Bug。

这个过程就类似于数学中的“极限”的定义，尽管我们知道某个极限值为 0，但是它永远也不可能达到 0。这也就产品的 Bug 永远也修复不完的原因。因此，我们在修复 Bug 时要注意，一定要记住用户最需要的是什么，然后优先尽力修复那些影响用户使用的 Bug；而对于大部分对用户影响很少、甚至根本不影响的 Bug，则可以推迟修复，甚至不修复。

在微软公司，Bug 经过开发人员解决后，再回到软件测试人员手中时，会被分为以下几种类型。

Fixed: 表示 Bug 已经被修复或更正了。

Duplicated: 表示测试人员所找到的某个 Bug 已经被别人找出来了。这种情况是很难避免的。一是因为同时有许多测试人员在进行测试，这样就很有可能多个测试人员先后发现同一个 Bug；二是因为同一个 Bug 在不同的进入境况下所表现的现象也不一样，尽管表面上看起来是不同的 Bug，但实际上可能是相同的。因此在报告一个 Bug 之前一般都要搜索该 Bug 是否已经存在。

Postponed: 表明这个 Bug 不是很重要，在当前阶段不用进行更正了；或者更正这个 Bug 风险太大，Bug 本身又不会造成大的影响。

By design: 测试人员认为是 Bug，不符合逻辑，也不符合用户的需求，但开发人员则认为是按照项目经理的设计做的。

在查找并修复 Bug 的过程中，测试人员和开发人员经常会发生争执。例如，当开发人员宣布某一个 Bug 已经被 Fixed 时，测试人员往往还不放心，又重新对该 Bug 进行检查；当开发人员宣布某一个 Bug 是 Duplicated 时，细心的测试人员也要验证该 Bug 是否和另外一个 Bug 相重复，如果另外一个 Bug 已经被修复了，但这个 Bug 还存在，就会让开发人员重新修复该 Bug；对于是否需要 Postponed 一个 Bug，测试人员和开发人员也常常争论不休，测试人员认为这个 Bug 需要修复，而开发人员则认为修复这个 Bug 不值得，这时候就需要项目经理来决定，因为测试人员要从用户的角度来看待 Bug，开发人员则要考虑到开发期限和付出的代价，而项目经理要同时考虑到这两种情况。

只有项目经理经过权衡之后才能确定是否要推迟修复该 Bug；在 By design 情况下，通常是争论最多的时候。这时候也需要三方都坐下来谈，其结果一般有三种：坚持原来的设计、修改原来的设计并增加特性、或者取消该设计。对 Not repro 和 Won't fix 情况也是这样，需要测试人员、开发人员和项目经理进行协商，直到三方达成共识。

四、软件测试方法和辅助工具

有了 Bug 类型的定义以后，如何去找出这些 Bug 呢？这就需要采用好的测试方法。以下介绍几种常用的软件测试方法。

有多种方式对软件测试方法进行分类。例如，从代码和用户使用的角度可以将软件测试方法分为如下几种。

(1) 覆盖性测试 (Coverage Testing)

覆盖性测试是从代码的特性角度(即内部)出发的测试方法，包括以下方式。

① **单元测试 (Unit Test)**: 按照代码的单元组成逐个进行测试。

② **功能测试 (Function Test)或特性测试 (Feature Test)**: 按照软件的功能或特性逐个进行测试。例如，在 Exchange 中，发送邮件和接收邮件就是两个不同的功能或特性，在测试时就分别对它们进行检查，看是否工作正常。

③ **提交测试 (Check-in Test)**: 在开发人员对代码做了任何修改，或者修复了某个 Bug 时，需要重新 Check-in 代码 (即将修改后的代码放大到整个大的系统中)。这时，开发人员往往也要进行测试，看代码是否工作正常。为了保险起见，开发人员往往要找测试人员帮着一起进行测试(我们把这种情况称做 Buddy Test)。测试人员和开发人员之间搞好关系是非常重要的，稍后我会专门讲述这一点。

④ **基本验证测试 (Build Verification Test, 简称 BVT)**: 对完成的代码进行编译和连接，产生一个构造，以检查程序的主要功能是否会像预期一样进行工作。这是最简单而又最省时的一种测试方法。每产生一个新的构造时都要进行测试。如果连 BVT 都通不过，表明问题很严重，开发人员需要尽快修复出现的问题，测试人员也就不需要浪费时间做其他

测试了。

⑤ **回归测试 (Regression Test)**: 过一段时间以后, 再回过头来对以前修复过的 Bug 重新进行测试, 看该 Bug 是否会重新出现。

(2) 使用测试(Usage Testing)

使用测试是从用户的角度(即外部)出发的测试方法。它也包括许多类型。

① **配置测试 (Configuration Test)**: 从用户的使用出发进行多方面的测试。例如, 保证软件不仅能够在 Windows 9X 下运行, 也能够在 Windows ME 下运行, 还能够在 Windows NT/2000/XP 下运行;或者软件不仅能够在配置高的计算机上运行, 也能够在配置很低的计算机上正确地运行。总之, 要考虑到用户的多种情况, 用多种配置对软件进行测试。

② **兼容性测试 (Compatibility Test)**: 主要考虑兼容性问题, 如同一个产品的不同版本(如 Office 2000 和 Office XP)之间的兼容问题, 不同厂家的同一个产品(如 IE 和 Netscape)之间的兼容问题, 不同类型软件(如 IE 和 Office)之间的兼容问题等。最难测试的往往就是软件的兼容性问题, 往往要投入巨大的人力和物力。一些厂商开发出来的产品在兼容性上做得很不好, 就是因为没有足够的人力和物力进行测试。

我在做 SQL Server 的 XML 测试的时候, 为了解决 XML 的兼容性问题, 用了 6 个测试人员和 100 台计算机进行测试。正因如此, 微软产品的兼容性都非常好。而不像市场上的一些产品, 安装以后就导致计算机上的许多其他软件无法使用, 或者出现各种各样的问题, 这样不仅伤害了其他软件, 也伤害了用户。

③ **压力测试 (StressTest)**: 在各种极限情况下对产品进行测试 (如很多人同时使用该软件, 或者反复运行该软件), 以检查产品的长期稳定性。例如, 我们在开发 IE 4.0 的时候, 由于当时有一个非常强的竞争对手, 因此我们必须保证 IE4.0 要做得非常好。当时, 为了测试 IE4.0 的长期稳定性, 我们专门设计了一套自动测试程序, 它一分钟可以下载上千个页面。我们使用这个测试程序对 IE4.0 进行了连续 72 小时的测试, 也没有出现任何问题, 如内存泄漏、程序崩溃等。

压力测试时间要求:根据微软的实践经验, 如果一个软件产品能通过 72 小时的压力测试, 则该产品超过 72 小时后出现问题的可能性微乎其微。所以, 72 小时就成为微软产品压力测试时间的标志。

本项测试可以帮助找到一些大型的问题, 如死机、崩损、内存泄漏等, 因为有些存在内存泄漏问题的程序, 在运行一两次时可能不会出现问题, 但是如果运行了成千上万次, 内存泄漏得越来越多, 就会导致系统崩滑。

④ **性能测试 (Performance Test)**: 本项测试是保证程序具有良好的性能。如果别人的产品只需 5 秒钟就能得出结果, 而你的产品需要 10 秒钟才能得出结果, 就说明你的产品

性能不好。如果在测试过程中发现性能问题，修复起来是非常艰难的，因为这常常意味着程序的算法不好，结构不好，或者设计有问题。因此在产品开发的开始阶段，就要考虑到软件的性能问题。

⑤ **文档和帮助文件测试 (Documentation and help file Test)**: 因为用户通常是通过文档和帮助文件来学习使用产品的，如果文档和帮助文件存在错误，就可能会导致用户无法正常使用产品。这项工作通常在产品即将 Ship(即准备包装发布)时进行，以避免在修复 Bug 的过程中需要反复修改文档，或者忘记修改文档，导致文档与产品的特性不相符。

⑥ **Alpha 和 Beta 测试 (Alpha and Beta Test)**: 在正式发布产品之前往往要先发布一些测试版，让用户能够反馈出相关信息，或者找到存在的 Bug，以便在正式版中得到解决。

还有一种分类方法将测试方法分为如下几种。

(1) **白盒测试 (White Box Testing)**

又叫做玻璃盒测试(Glass Box Testing)。在软件编码阶段，开发人员根据自己对代码的理解和接触所进行的软件测试叫做白盒测试。这一阶段测试以软件开发人员为主，有时候 SDE/T 也会辅助开发人员进行测试。

(2) **黑盒测试 (Black Box Testing)**

黑盒测试的内容主要有以下几个方面。

① **接受性测试 (Acceptance Testing)**: 类似于 BVT 测试。

② **Alpha/Beta 测试 (Alpha/Beta Testing)**: 在此过程中，产品特征不断地修改。当发现 Bug 后，在开发人员修改的同时，项目经理也会对产品计划做出相应的调整，产品计划不是一成不变的

③ **菜单/帮助测试 (Menu/Help Testing)**: 大家千万不要以为这一项测试不值得进行。其实，在软件产品开发的最后阶段，文档里发现的问题往往是最多的。因为在软件测试过程中，开发人员会修复测试人员发现的 Bug，而且可能会对软件的有些功能进行修改，同时项目经理也会根据情况调整软件的特性，因而在软件开发和测试的过程中，所有的功能都不是固定不变的，都会进行调整。所以，一般来说，直到软件 Ship 时才编写软件的帮助文档，这样才能保证帮助文件的内容与软件功能相符，我在做帮助文件测试的时候，总是假装什么都不懂，就按照帮助文件提供的步骤去做，看看该文件是否正确。在实际测试中，我经常能发现帮助文件中的 Bug。

④ **发行测试 (Release Testing)**: 在正式发行前，产品要经过非常仔细的测试。除了专门的测试人员外，还需要几千个甚至几十万其他用户与合作者通过亲自使用来对产品进行测试，然后将错误信息反馈给我们。到了发行测试这一步，如果出现非改不可的 Bug，就必须推迟软件的发行，有的时候一推就是几个月，期间需要重新对软件产品进行全面的测试，耗费大量的时间和人力物力。

⑤ **回归测试 (Regression Testing)**: 回归测试的目的就是保证以前已经修复的 Bug

在软件 Ship 以前不会再出现。实际上，许多 Bug 都是在回归测试时发现的，在此阶段，我们首先要检查以前找到的 Bug 是否已经更正了。值得注意的是，已经更正的 Bug 也可能又回来了，有的 Bug 经过修改之后可能又产生了新的 Bug。所以，回归测试可保证已更正的 Bug 不再重现，不产生新的 Bug。

⑥ RTM 测试 (Release To Manufacture Testing)：为产品真正的 Ship 做好准备所进行的测试。事实上，在这一测试阶段，对每一个 Bug 都需要经过很高职务的人同意才能更正。因为这时候修改软件非常容易产生其他的错误，所以只有那种非修复不可的 Bug 才会被允许进行修改。如果在发行阶段软件还有许多严重的 Bug 的话，恐怕就不能按时发布了。记得有一次一个微软核心产品刚刚完成，准备 Ship 时，我对其进行 RTM 测试时就发现一个 Bug：只要用该产品打印中文就会导致程序错误。这是一个很严重的 Bug，于是开发人员马上修复了该 Bug，重新 Ship 该产品。

功能及系统测试 (Function & System Testing)：这一点是最重要的，他包括了非常多的内容。

- Ø 规范验证 (Specification Verification)
- Ø 正确性 (Correctness)
- Ø 可用性 (Usability)
- Ø 边界条件 (Boundary Condition)
- Ø 性能 (Performance)
- Ø 压力测试 (Stress)
- Ø 错误恢复 (Error Recovery)
- Ø 安全性 (Security)
- Ø 兼容性 (Compatibility)
- Ø 软件配置 (Configuration)
- Ø 软件安装 (Installation)

另外一种分类方法就比较好理解了，主要将软件测试方法分为如下几种。

(1) 手工测试 (Manual Testing)：即依靠人力来查找 Bug。

(2) 自动测试 (Automation Testing)

即编写一些测试工具，让他们自动运行来查找 Bug。

自动测试的优点是能够很快、很广泛地查找 Bug，缺点是它们只能检查一些最主要的问题，如崩溃、死机，但是却无法发现一些一般的日常错误，这些错误通过人眼很容易找到，但机器却往往找不到。另外，在自动测试中编写测试工作量也很大，因此在实际测试中通常是手工测试和自动测试相结合，而且手工测试往往是主要的，占了 1/2—2/3，而自动测试只占 1/3—1/2。在不同的开发队伍中，这个比例会有所不同，但总体趋势是这样的。

自动测试案例一：我在 Exchange Server 团队的时候，常常要做一种 Stress Test，需要几万个甚至几十万用户同时把 E-mail 发送到服务器 (Server)，以保证 Server 不会出现死机或崩溃的现象。可是，需要几万人同时发送 E-mail，这在现实生活中很难人为实现。但是，利用测试工具就可以非常容易地做到。测试工具可以自动产生几万个账号，并且让它们在同一时间从不同机器上(一个机器上可以有多个不同的账号)同时发送 E-mail 信息。

自动测试案例二：举一个浏览 Web 页面的例子，要求 50000 个用户同时浏览一个 Web 页面，以保证网站的服务器 (Server)不会死机。一般来说，找到 50000 个用户同时打开一个网页是不现实的，就算能够找到 50000 个测试者，成本也非常高。但是通过测试工具则很容易做到，并且工具还可以自动判断浏览结果是否正确。

有了好的测试方法，还需要有高效好用的辅助工具，做软件测试通常需要以下一些基本工具。

- Ø 计算机
- Ø 优秀的办公处理软件（如字处理软件和表单软件，用于编写测试计划和规范）
- Ø 视频设备
- Ø 秒表（角色程序运算时间，测试产品性能）
- Ø 错误跟踪系统（微软内部使用的是 RAID，用来自动跟踪 Bug）
- Ø 自动测试工具（产生 Automation 脚本）
- Ø 软件分析工具
- Ø 好的操作系统（如 Windows NT/2000，其中有很多有用的工具，如文件比较器、文件查看器、文件转换器、内存监视器等）
- Ø 多样化平台

（本文节选自《软件开发的科学与艺术》第 9 章）