每日构造和点烟测试（“Daily Build and Smoke Test”）

李帜 译

如果你只想创建一个简单得只包含一个文件的计算机程序，你所要做的工作只不过编译链接一个文件。在一个典型的团队项目中，包含了几十个、几百个、甚至几千个文件，创建一个可执行程序的过程可能变得非常复杂和耗时。你必须从程序的不同部件来进行构造。

一个在微软公司和其他一些业界领先的软件公司的一个普遍的实践经验就是建立："每日构造和点烟测试（“Daily Build and Smoke Test”）"过程。这个过程就是，每天把所有文件编译、链接、组成一个可执行程序， 然后对这个程序进行一个点烟测试（"smoke test"）也就是一个相对简单的检查，看一看这个程序是否能够在运行时冒烟，也就是是否能够体现出程序正常运行所能体现的一些正常表象。

优点

这个简单的过程有很多明显的优点。

它最小化了集成工作的风险。一个项目团队面对的一个最大的风险就是，当不同项目成员把他们的分别开发的代码组合或集成在一起，得到的结果代码常常不能很好的工作。并且依赖于何时才能够发现代码中的不兼容的情况，调试这些程序比集成工作做得早的情况更长，程序的接口可能会被改编，或者系统的重要部分不得不进行重新设计和重新实现。在一个极端的案例中，集成工作中引起发现的错误导致项目的取消。每日构造和点烟测试（“Daily Build and Smoke Test”）过程保证了集成的错误比较小并且是可管理的，防止遇到失控的集成问题。

它降低了产品质量低的风险。与不成功的和有问题的集成相关的风险是低质量。 通过每天对所有代码进行一次的最小的点烟测试 ，防止在项目中发生难以控制的质量问题。你可以使系统处于一个可知的良好状态，并且保持这个状态，当消耗时间的问题发生时，你可以简单的不允许不接受这个状态的恶化。

它是故障诊断变得容易。当每天对产品进行构造和测试，可以容易得查清在任何给定的日期产品失效的原因。 如果在第17天产品可以工作，但是第18天产品却失效，可以明确确认在两个构造之间发生的一些事情使产品失效。

它可以提高士气。 看到一个产品在工作可以使士气得到一个难以置信的推动。几乎不管是做什么产品，开发者仅仅看到产品显示了一个界面就会感到兴奋。当采用每日构造“Daily Build”，每天都能使产品的功能比前一天多一些，这样可以保持开发团队的高昂士气。

使用每日构造和点烟测试（“DAILY BUILD AND SMOKE TEST”）

这个过程中的含义是简单的每天对产品构造和测试。在这个简单的方法中有一些特殊的具有特点细节。

每日构造(“Build daily”)。每日构造中最基本的部分是构造是每日进行的，正如“Jim McCarthy”(*Dynamics of Software Development*, Microsoft Press, 1995，国内翻译微软成功的奥秘)说的，把每日构造看做一个项目的心跳，如果没有心跳，项目就死了，一个不太像的比喻，Michael Cusumano 和 Richard W. Selby(*Microsoft Secrets*, The Free Press, 1995)把每日构造形容为一个项目的同步脉搏。在这些脉搏之间不同的开发者的代码，允许有一点不同步，但是在每个同步点，这些代码必须返回到同步的状态。当你坚持保证所有的同步脉搏，你就能保证开发团队不会失去同步。

一些组织每隔几个星期才进行产品构造而不是每天进行。这样做的问题在于如果某一个构造失效的话，你可能要等几个星期才能得到下一个好的构造版本。当这样的事情发生，你会实际上失去所有经常性构造过程能够带来的好处。

对实效的构造进行检查。为了每日构造过程能够工作，每日构造过程中构件的软件应该能够工作。如果这个软件不可用，那么这个构造被认为是失效的，修复这个构造的任务被认为是具有最高优先级的。

每一个项目都会设置它自己的标准，包括什么是构造失效，这个标准也要设置一个质量标准，这个质量标准要足够的严格使重要关键的缺陷被排除，也要有足够的宽容度是让琐碎的不重要的缺陷能过通过测试，不要让不适当的过分关注不重要的缺陷使进度瘫痪。

一个"良好的" 构造至少应该做到：

- 成功的编译所有的文件、库和其他部件；
- 成功的链接所有的文件、库和其他部件；
- 不包含任何人和关键错误，这些缺陷可能使程序不能启动，或者可能对运行有困难
- 通过点烟测试“ smoke test ”

每日点烟测试（“ Smoke test ”）

点烟测试(smoke test)应该对整个系统进行端到端检验。他不应该是详尽周密的，但是它应有能力暴露出主要问题。如果构造通过点烟测试(smoke test)，应是足够彻底的测试，你可以认为 他是足够稳定的可以用来进行彻底的测试。

没有点烟测试(smoke test)每日构造没有一点价值。点烟测试(smoke test)是一个岗哨，用来防止产品质量的恶化以及 令人厌恶的集成问题。没有点烟测试，每日构造就会变成是每天得到一个干净的编译结果的浪费时间的活动。

点烟测试(smoke test)必须和系统的演进而演进。在最开始，点烟测试(smoke test)只是进行一些简单测试的，例如：系统是否可以表示"Hello, World"，随着系统的开发，点烟测试(smoke test)会变得更周详，第一个测试可能只需要几秒钟的运行，随着系统增长，点烟测试(smoke test)的时间可能增长到30分钟，一个小时甚至更多。

建立构造组。

在很多项目中，想要保持每日构造和使点烟测试(smoke test) 是最新状态是一个很大的任务，足够是某一位成员明确的部分工作职责。在一个大的项目中，这项任务可以是一个或更多成员的全职

工作。例如，在Windows NT 3.0 项目中，在构造组中就有4个全职项目成员(Pascal Zachary, *Showstopper!*, The Free Press, 1994)。

当感觉上有需要时就给构造增加修订。

单独的开发者通常在一天的基础上不能很快的编写代码，可以给系统增加有意义的增量。他们应该写出大段代码再把他们集成起来，这样就可以把代码集合保持在一个一致的状态—通常是每隔几天就做一次。

对构造的失效追究责任。

在多数采用每日构造过程的开发组中都要对破坏这一过程的原因追究责任。非常明确的从项目开始就确认保证每日构造过程的健康运转是项目最高优先级的任务。任何打破这一过程的都是一个意外，不是规则。一定要坚持让那个破坏构造的开发者停止任何其他工作直到修复好这个构造。如果这个构造过程过于频繁地被破坏，那么很难认真展开工作来确保构造不是失效的。

一个轻微的惩罚措施可以帮助强调这个优先级。一些组会给任何一个破坏构造过程的成员出示一个圆牌直到他修复了这个问题。另一些组会让犯错误的开发者戴上羊角标识或让其捐献出$5给活动基金。

一些项目建立了更进一步处罚措施。在微软，像的“Windows NT”，“Windows 95”,和“Excel”这样重要的项目中，在开发的最后阶段开发者要配置上报警器。如果他们破坏了构造过程，甚至如果缺陷在早上3点被发现，他们也会被叫来处理，修正这个缺陷。

在一定的压力下进行构建和点烟测试

当进度计划表的压力变大，维护每日构造的需要的工作变得好像是奢侈和过分的。相反的意见是对的。在压力下，开发者会失去一些他们自己的规矩，他们会采用一些在没有压力情况下不会采用的一些设计和实现的捷径。他们代码审查和单元测试会比通常的情况下更不细心。代码的会比没有压力的情况下更快趋向于熵的自然的趋向于混乱状态。

为了对付这些情况，每日构造过程强迫时间一些规则在这些有压力的项目在一定轨道上进行。代码仍然有趋向熵的趋势,但是每天的构造过程把这个趋势控制相应的范围在。

结论

谁能够从这个过程中得到好处？一些开发者断言说，因为项目太大以至每天于进行构造过程是不切实际的。但是近来的可能是最复杂的软件项目中都成功的采用每日构造过程。到产品发布的时候，“Microsoft Windows NT 3.0”包含有4万多个文件，共有五百六十万行代码。一个完整地构造过程会花费多台机器设备和19小时的时间，但是NT开发队伍仍然设法进行每日构造过程(Zachary,1994)。在里这每日构造过程远远没有被看成是对项目一个损坏，相反“NT”开发团队把他们在庞大项目上取得的成功部分归功于他们进行的每日构造过程。以至于那些在的项目工作中遇到较少错误的成员已经很难解释为什么这个过程带来没有给他带来的优点了。

译者后注：适合盒装软件／基础软件的开发，对于运营系统，MIS、BSS、OSS系统也许不是很适合。需要投入的资源，需要有一些特殊的技能和全新的概念和理念来支持。