第一图书网, tushu007.com

<<卓有成效的程序员>>

图书基本信息

书名：<<卓有成效的程序员>>

13位ISBN编号：9787111264064

10位ISBN编号：7111264061

出版时间：2009-3

出版时间：机械工业出版社

作者：Neal Ford

页数：216

译者：ThoughtWorks公司（中国）

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

前言

译者序消除浪费，始于细节在一次关于敏捷的讨论中，我说了一句令很多人不解的话：我不要敏捷。
和很多话一样，断章取义的理解很容易造成误会。
我当时说的整句话是：我不要敏捷，我要致力于消除软件开发中的一切浪费。
当“敏捷”渐渐变成一个人见人爱的“大词”，越来越多的人开始发现，其实自己要的不是“be agile”，而是切实地消除浪费、提高效率。
所以，作为ThoughtWorks员工的Neal Ford在他的这本书里闭口不谈“敏捷”。

内容概要

本书讲述如何在开发软件的过程中变得更加高效。
同时，本书的讲述将会跨语言和操作系统：很多技巧的讲述都会伴随多种程序语言的例子，并且会跨越三种主要的操作系统，Windows（多个版本），Mac OS X以及 *-nix（Unix或者Linux）。
　　本书讨论的是程序员个体的生产力，而不是团队的生产力问题，所以它不会涉及方法论（好吧，可能总会在这里或那里谈论到一些，但肯定不会深入讨论）。
同时，本书也不会讨论生产力对整个团队的影响。
我的使命，是让作为个体的程序员通过掌握恰当的工具和思想变得更加高效。

作者简介

Neal Ford是ThoughtWorks的软件架构师。
他曾在美国和其他国家进行现场授课，客户包括军方和很多《财富》500强的企业。

书籍目录

译者序序前言第1章 简介　为什么要写一本关于程序员生产力的书？
　本书包含哪些内容？
　如何读此书？
第一部分 机制　第2章 加速　启动面板　加速器　宏　小结　第3章 专注　排除干扰　搜索优于导航　找出难找的目标　使用有根视图　设好“粘性属性”　使用基于项目的快捷方式　使用多显示器　用虚拟桌面拆分工作空间　小结　第4章 自动化　不要重新发明轮子　建立本地缓存　自动访问网站　与RSS源交互　在构建之外使用Ant　用Rake执行常见任务　用Selenium浏览网页　用bash统计异常数　用Windows Power Shell替代批处理文件　用Mac OS X的Automator来删除过时的下载文件　驯服Subversion命令行　用Ruby编写SQL拆分工具　我应该把它自动化吗？
　别给牦牛剪毛　小结　第5章 规范性　DRY 版本控制　使用标准的构建服务器　间接机制　利用虚拟平台　DRY 阻抗失配　DRY 文档　小结 第二部分 实践　第6章 测试驱动设计　不断演化的测试　代码覆盖率　第7章 静态分析　字节码分析　源码分析　用Panopticode生成统计数据　动态语言的分析　第8章 当个好公民　破坏封装　构造函数　静态方法　犯罪行为　第9章 YAGNI　第10章 古代哲人　亚里斯多德的“事物的本质和附属性质”理论　奥卡姆剃刀原理　笛米特法则　“古老的”软件学说　第11章 质疑权威　愤怒的猴子　连贯接口　反目标（Anti-Objects）　第12章 元编程　Java和反射　用Groovy测试Java　编写连贯接口　元编程的归处　第13章 组合方法和SLAP　组合方法实践　SLAP　第14章 多语言编程　历史与现状　路在何方？
　Ola的金字塔　第15章 寻找完美工具　寻找完美编辑器　编辑器参考列表　为你的工作选择正确的工具　丢弃错误的工具　第16章 尾声：继续对话附录 Building Blocks

章节摘录

奥卡姆剃刀原理奥卡姆 的威廉爵士是一个厌恶华美装饰以及复杂解释的修士。
他对哲学和科学的贡献是奥卡姆剃刀原理：如果对于一个现象有好几种解释，那么最简单的解释往往是最正确的。
显然，这跟我们讨论的事物本质和附属性质理论紧密关联。
这个原理对于软件的影响度也是出乎我们意料的。
作为软件工业中的一员，过去十年我们一直在进行着某项实验。
这个实验始于上世纪90年代中期，主要是由于开发人员发现其开发进度远远跟不上软件需求的增长而引发的（其实在那时这已经不是一个新问题，这个问题自商业软件的想法出现之后就一直存在）。
实验的目的是：创造一些工具和环境来提高那些普通开发人员的生产率，即使一些人比如Fred Brooks（去看他的《人月神话》）已经告诉我们软件开发中的一些混乱事实。
此实验试图验证：我们是否可以创造一种能限制程序员破坏力的语言而使人摆脱麻烦；我们是否可以无需支付荒唐的大量金钱给那些令人生厌的软件技工（即使在那时候你可能还为找不到足够的软件技工而发愁），而同样生产出软件呢？
这些思考让我们创造出了如dBase, PowerBuilder, Clipper和Access这样的工具，并促成了工具和语言相结合的4GL(第四代语言)的崛起，比如FoxPro和Access。
但问题是，即使有这样的工具和环境你也不能完成所有的工作。
我同事Terry Dietzler为Access创建了一个叫做"80-10-10"的准则(而我喜欢把它称之为Dietzler定律)。
这个定律说的是：80%的客户需求可以很快完成；下一个10%需要花很大的努力才能完成；而最后的10%却几乎是不可能完成的，因为你不能把所有的工具和框架都"招致麾下"。
而用户却希望能满足一切需求，所以作为通用目的语言的4GL（Visual BASIC、Java、Delphi以及C#）应运而生。
Java和C#的出现主要是由于C++的复杂性和易错性，语言开发者们为了让一般程序员摆脱这些麻烦而在其内建了一些相当严格的限制。
在此之后"80-10-10准则"才发生了改变，无法完成的工作已经微乎其微。
这些语言都是通用目的语言，只要付出足够的努力，大多数工作都可以完成。
但Java虽然比较易用却常常需要大量编码，所以框架出现了，Aspects出现了，大量其它框架蜂拥而至。
下面有一个例子。
这段Java代码是从一个广泛使用的开源框架中提取出来的，试着找出它的用途吧（关于它的名字我只会提示你一点点）：public static boolean xxXxxxx(String str) {int strLen;if (str == null || (strLen = str.length()) == 0) {return true;}for (int i = 0; i < strLen; i++) {if ((Character.isWhitespace(str.charAt(i)) == false)) {return false;}}return true;} 花了多少时间？
这实际上是一个从Jakarta Commons框架（它提供了一些或许本该内置于Java的帮助类和方法）中提取出来的isBlank方法。
一个字符串是否为"空白"由两个条件决定：这个字符串是空字符串，或者它只由空格组成。
这段代码的计算公式非常复杂，因为要考虑参数是null的情况，而且还要迭代所有的字符。
当然，你还要把字符包装成Character类型以确定它是否空白字符（空格、制表符、换行符等）。
总之，太麻烦了！

媒体关注与评论

对于程序员，过去我们一直习惯于用单纯的技术水平，也就是实现程序功能的能力来衡量。
然而这个时代其实已经过去了。
虽然技术仍然很重要，但企业越来越多地认识到，对于程序员更全面的衡量标准，应当是生产率。
只有能够以较高的效率完成对项目、对企业有价值的工作，才是团队和组织所真正需要的人才。
反之，技术好，但不能真正促进整体价值，甚至其反作用，这样的“技术牛人”已经没有生存空间了。
—— 孟岩 《程序员》杂志总编&ldq

编辑推荐

《卓有成效的程序员》是一本揭示高效程序员的思考模式，一本告诉你如何缩短你与优秀程序员的差距。
以下媒体、专家、社区联合推荐：媒体：《程序员》杂志、《电脑编程技巧与维护》杂志专家：韩磊、孟岩、冯大辉、李剑、黄晶、温昱、周爱民

名人推荐

《卓有成效的程序员》这本书，个人觉得单独针对“程序员”可能还有点窄，其实《卓有成效的程序员》的大部分内容对所有技术人员也是适用的。
但愿看了《卓有成效的程序员》之后，能有更多的技术人员真正的行动起来，利用这本书提升自己，也去积极影响他人，形成更良性的互动，不要让“持续改进”成为一句空话。
另外，必须要补充的是，如果技术人员持续从事低效率的工作，极有可能逐渐厌烦技术，疏远技术，乃至对技术绝望，而一个高效的技术人才能从技术中获得真正的快乐。

第一图书网, tushu007.com

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com