<<梦断代码>>

图书基本信息

书名：<<梦断代码>>

13位ISBN编号：9787121066795

10位ISBN编号：7121066793

出版时间：2008.06

出版时间：电子工业出版社

作者：Scott Rosenberg

页数：336

字数：274000

译者：韩磊

前言

大多数技术书籍只是讲技术和理论，但我们不知道在真实的软件开发过程中，这些技术和理论究竟是被什么样的人如何去使用？
《梦断代码》向我们展示了硅谷一流软件开发者是如何进行产品开发的，把真实的人、事、技术以及产品的发展过程结合在一起，每个有志于开发畅销产品的程序员都值得耐心去品味这个故事。
CSDN总裁蒋涛2008年6月

<<梦断代码>>

内容概要

软件乃是人类自以为最有把握，实则最难掌控的技术。
本书作者罗森伯格对OSAF主持的Chandler项目进行田野调查，跟踪经年，试图借由Chandler的开发过程揭示软件开发中的一些根本性大问题。
　　本书是讲一事，也是讲百千事；是写一软件，也是写百千软件；是写一群人，也是写百千万人。任何一个在软件领域稍有经验的技术人员看完本书，必掩卷长叹：做软件难。

作者简介

Scott Rosenberg，作家，编辑，1981年毕业于哈佛大学，1995年与他人共同创办了Salon网站，此后担任其首席技术编辑达数年之久，并负责技术工作。
从1986到1995年，一直为San Francisco Examiner写作，最初写剧评，后来又写影评和“数字文化”专栏。
所写的剧评曾于1989年获George Jean Nathan奖。
在进入Examiner之前，一直为Boston Phoenix写剧评、影评和书评。
个人博客地址为www.wordyard.com。

<<梦断代码>>

书籍目录

第0章　软件时间第1章　死定了[2003年7月]第2章　Agenda之魂[1968～2001年]第3章　原型与Python[2001～2002年11月]第4章　乐高王国[2002年11月～2003年8月]第5章　管束奇客和狗[2003年4月～8月]第6章　完成设计方案[2003年7月～11月]第7章　细节视图[2004年1月～5月]第8章　白板上的即时贴[2004年6月～10月]第9章　方法第10章　工程师和艺术家第11章　通往狗食版之路[2004年11月～2005年11月]尾声　长赌[2005～2029年及以后]译后记附录A　专有名词对译表

<<梦断代码>>

章节摘录

第0章 软件时间那是1975年的冬天。
我在终端机房中俯身敲击一台电传打字机，每打完一行，那笨重的机头就会摇头晃脑猛然撞回最左边，开始新的一行。
我从几个小时前开始输入一行行黑代码，忘记了时间流逝，全然不知已是午夜时分。
看门人已经关闭廊灯。
我并没有得到许可在纽约大学物理系大楼中流连忘返、使用向高中学生免费发放的计算机账号。
不过，倒也无人责难。
那时我年方十五，正迷恋于一个叫做Sumer的游戏，在游戏中，我管理着新月沃土上一座古代城邦。
今天的电脑游戏玩家也许会嘲笑其稚嫩：它在一卷纸上逐行打印出大写字母，报告游戏进程。
玩家运筹帷幄，分配食用和留种的谷物，然后程序就会告知城邦每年的发展情况。
“汉谟拉比陛下，”程序像一个诚惶诚恐的宰相般报告说，“微臣伏启圣鉴……”没过几天，我就已经把游戏玩了个遍。
但是，和现在令青少年着迷的大多数游戏不同，Sumer可以让玩家打补丁。
谁都能够窥探其内部运行机制：该游戏只是向计算机发出的一系列简单指令，这些指令存储于一卷多行八孔纸带上。
（电传打字机旁的塑料盘中堆积的纸带，几乎带来和游戏一样多的乐趣。
）纸带像地下出版物一般在朋友间流传共享。
只要花几个钟头学点简单的Basic语言，改游戏就会和玩游戏一样容易：将纸带上的指令装载到计算机，然后开始往程序里加代码。
Sumer是个空白画板——历史只是个轮廓，随时准备着让少年的梦想来浇筑。
我和朋友们掌握了它简单的构造，开始往里加东西。
让玩家可以选择不同的宗教信仰吧！
偶尔来一次腺鼠疫，会发生什么事？
蛮族入侵者应该很酷。
嘿，搞几具弹石机如何？
那天晚上，我倾力于改造游戏中民众造反模式的设计。
Sumer只提供粗糙的起义模式；如果你干得太差，人民就会起来推翻你。
（Sumer的原作者是个乐天派。
）我认为，游戏中的起义模式应该多种多样，所以就创建了一些子程序补丁——让Sumer陷入内战，或是引入一个想要争取合法地位的反对派政府。
我不在乎干到多晚。
F线地铁整夜运行，可以把我带回位于皇后区的家。
一定得重新设计游戏中的革命模式！
四分之一世纪后，2000年五月，我坐在旧金山的一间办公室里，眼睁睁紧盯着一台现代计算机屏幕（高解析度、数百万种颜色）。
喝剩的咖啡纸杯胡乱摆在键盘边上。
正是凌晨5点。
那时我四十岁，是在线杂志Salon的创始人和总编，此外还负责一个软件开发项目。
我们花了几个月精心规划，希望给网站增加动态特性，使之彻底改观。
然而，现在我却眼看着项目濒临绝境。
在没日没夜苦干了几个星期之后，主力程序员终于宣告工作完成，自己要飞往夏威夷，度一个全家盼望已久的假期。
剩下他的老板，技术副总裁查德?迪克森（Chad Dickerson），独自琢磨为什么存储网站文章的数据库就是不肯与负责显示页面的程序对接。
查德两个通宵没合眼，努力修复问题。

若是不然，到周一早上，我们的两百万读者就只能看到网站上没更新过的旧闻了。
难道我们以前没做过软件吗？
做过。
是没有全面测试吗？
显然不太充分。
怎么会搞得一塌糊涂？
鬼晓得。
我吃完了自动售货机里最后一袋饼干，徘徊又等待，却仍是毫无指望。
时间还多。
还有时间去读那位以新项目的名义准备香槟加蛋糕聚会的倒霉同事的邮件，回复他说：“或者咱们再等等吧。
”还有时间去体会身陷困境孤立无援的感受，然后琢磨将系统的中心服务器命名为“卡夫卡”是不是个好主意。
大约早晨9点，我们终于发布了站点“改进版”的第一个版本。
又是周一清晨，其他同事相继出现在办公室，他们过了好一会儿才知道，原来我们六个人昨夜压根儿就没回家睡觉。
又过了几星期，程序员们修复了最严重的问题，软件运行趋于平稳。
但后来每每听说某公司打算“升级其软件平台”、重新搭建一套大型系统时，我总不免暗自担心。
20世纪90年代科技行业的兴盛，给我们带来了“互联网时间”的概念。
对该短语含义的理解见仁见智，但多指“快速”之意。
数字时代的新时间机制下，一切皆有可能发生——技术产生、公司创立、创造财富——而且速度惊人。
这意味着你没时间做到尽善尽美——无须担心，因为别人也一样。
随着投资潮退，“互联网时间”这个短语也风头尽失，很快被其他时髦词所代替。
但新词掮客们的确一语中的。
在做软件的过程当中，时间似乎确实时快时慢。
如果一切顺利，你会沉浸在心理学家称之为“流逝”的状态中，全然忘记了时间。
如果事有不谐，你又会陷入困境，四顾茫然、举步维艰。
无论是哪种情况，时钟都被抛诸脑后。
你用的是软件时间。
在使用一门新编程语言时，程序员的第一个程序通常是“Hello World”——输入一系列代码，召唤计算机，命令它打印出这两个词，向主人致敬。
在Sumer游戏所用的Basic语言中，这个程序像这样：10 PRINT“HELLO WORLD!”20 STOP“Hello World”程序一无所用，但足可蛊惑人心；它鼓励新手，唤起每个程序员心中乐观的一面。
“既然能叫它说话，就能让它做任何事！
”计算机协会（The Association for Computing Machinery），计算机领域中的ABA或AMA，维护了一张网页，上面列出将近两百种编程语言版本的“Hello World”程序。
简直就是程序代码的罗塞塔石碑。
在Java这种商业世界中流行的重量级编程语言里面，“Hello World”看起来高不可攀：class HelloWorld {public static void main (String args[]){System.out.println(”Hello World!”);}}Public static void：无数个Java程序代码块中，都有这串密语存在。
这串词有特别的技术含义。
不过我常常把它看作一阙机器诗篇，在它召唤出的冷宫里面，多少软件项目一开始雄心勃勃，最终却未结善果。
如果你和计算机编程打过交道，就很难不对它又爱又恨。
作为少年游戏玩家，我品味过编写代码的巨大快乐。
作为媒体工作者，我见证了软件世界中无数个悲惨故事——无论是跨国公司、政府机构，还是军工大

鳄，都曾一头撞上过代码的冰山。
而作为一个经理人，我也得对付自己桌面上的泰坦尼克号。
这25年令人气馁的软件历史，也许不具代表性，但却是我的个人经验。
依照硅谷的数字乌托邦理想，事情理应朝好的方向发展。
在Salon网站发布失败后的几个月里，理想与现实之间的差异开始对我露出了利齿獠牙。
编程已不再处于萌芽期。
我们的世界依赖于无穷复杂的软件。
在长达半个世纪的研究和实践之后，为什么还是很难做到按时限、按预算做出计算机软件？
为什么还是很难让软件可靠而安全？
为什么还是很难把软件做得易于学习使用，且具备按需修改的灵活性？
这只跟时间和经验有关吗？
是否有出现某种根本性突破的可能？
在软件的本质特性（抽象性、复杂性及延展性）上，是否存在某种总能击倒我们的无常之物，将开发者咒入充满不可挽回的延误和根深蒂固的缺陷的世界？
“ 软件难做， ”编程界经典教科书的作者高德纳（Donald Knuth）这样写道。
但原因何在？
你可能已经注意到，我把本章标为“ 第0章 ”。
我无意搞笑，只是想指出计算机程序员和其他人的一处小小不同：程序员从0开始计数，而不是从1开始。
要解释这种习惯的来源，得从计算机中央处理单元里的寄存器，以及数据队列的结构等等奥义秘辛说起。
不过，我发现最直截了当的解释来自于一个网页，该网页试图向大众解释黑客的行为——“ 黑客 ”一词的本义是“ 痴迷的编程匠人 ”，而非后来衍化出的贬义“ 数码入侵艺术家 ”。
为什么程序员要从0开始计数？
因为计算机从0开始计数！
所以，程序员也训练自己这样计数，以免让他们要指示操作的计算机产生误解。
这本也无伤大雅，只是使用计算机的大多数人是从1开始计数，未免令人烦恼。
往下到系统层面，在这个层面上，数据被存储和操作——意味着我们的金钱、工作和设想被转换为机器可读的符号——计算机程序及编程语言经常会做小小的偏移操作，即“ +1 ”或“ -1 ”，使得计算机从0开始计数的列表与人类从1开始计数的列表保持同步。
在计算机的二进制数字世界里，所有的信息都被简化为0和1的序列。
但是，在0和1 之间有空间存在，在机器计数和思考的方式和人类计数和思考的方式之间也有空间存在。
当你寻找软件缺陷、延误和不按设计思路运行的原因时，那原因就藏身于这空间之中。
在构思本书的那段时间里，我每天要驾车从旧金山海湾大桥（Bay Bridge）上通过。
一天早晨，当我的车努力爬上连接奥克兰（Oakland）岸边和桥东段中心较高地带的长长引桥时，我发现，右边有个新物体挡住了海湾碧水和远山绿树：那是一台高耸的红色起重机的顶端，正好超出桥面。
它在那儿日复一日地矗立着，突然有一天，又多了12台起重机，在桥北一线齐齐排列，如同挤在食槽旁的机械怪兽，等着倒霉的上班人士送进嘴来。
这工程是要替换双层大桥的北半部分。
在1989年的Loma Prieta大地震发生时，该部分上层一段五十英尺长的桥面坍塌到下层的车行道上。
现在，将在旧桥旁边搭建一座更安全、更现代的新桥。
随后几个月，这些240英尺高的起重机，开始将一根根直径达8英尺、长达300英尺的锈钢管打进海湾水底。
在清晨时分，从我远在伯克利（Berkeley）山的家中都可以听到敲击声。
总共将会有160根这种大管子被打入海底，填上混凝土，支撑新桥的水上部分。

整个过程设计精密、执行无误；它分毫不差，完全满足了我们对工程一词的信心。
关于软件缺陷的话题，只要谈上几分钟，必会有人拍案叹道，“为什么就是不能像造桥那样造软件？”和摩天大楼、水坝等永久性建筑一样，桥梁体现了人类对物理世界的技术把握。
在过去半个世纪里，软件成为构建这个世界的虽不可见但却深入渗透的人造物。
“人类文明运行于软件之上，”广为应用的计算机语言C++发明人比昂纳?斯卓思柯普（Bjarne Stroustrup）这样说道。
初听起来，这像是奇谈怪论或是自卖自夸。
即便没有Microsoft Windows，人类文明也会同样延续，对吧？
然而，软件并不只是用来发电子邮件或写报告的程序那么简单；它已经不声不响地渗透到生活的每个角落。
它存在于厨具里、汽车中、玩具里，建筑中。
商业和银行、选举和新闻媒体、电影和交通网、医疗和国防、科研和基础公共服务——人类生存之所需都系于计算机代码这根易断的细线上。
而且我们要为其脆弱埋单。
根据国家标准和技术学会（National Institute of Standards and Technology）2002年的研究，软件错误每年造成美国595亿美元的经济损失，三分之二的项目明显延误或超出预算，甚至干脆无疾而终。
人类文明运行于软件之上。
但是，软件创建艺术却隐于暗处，即便对于专家们也是如此。
在历史上，我们从未如此地完全依赖于这样一种人类自己不知道怎么才做得好的产品。
在对软件系统的加速依赖和踱着方步学习怎么做好软件之间，有一条巨大且有时叫人恐惧的壕沟。
对软件的依赖以指数级增长，而做软件的技能——和应用技能的愿望——却进展缓慢。
你要和程序员说这些，就等着挨批吧。
这边厢，有人也许会说，世界从未如此光明：我们拥有了比以往更好的工具、更好的测试、更好的语言，还有更好的方法！
那边厢，你又听人家说，自计算机时代的黎明以来，其实并未取得多少进展。
计算机先驱莫瑞斯?威尔克斯（Maurice Wilkes）[7]回忆起1949年他在英国剑桥工作的情形，在拖着打孔纸带上楼给雏形计算机EDASC装载程序时，他看到了未来：“我强烈地意识到，生命中剩下的好日子，都将耗费在给自己的程序找错误上头。
”从威尔克斯的时代直到现在，尽管有那许多创新，程序员却一直陷于调试除错之苦境。
工作中只有百分之一的灵感迸发，剩下的是艰难寻找、汗湿重衣；他们的作品永远尚未完成或未臻至善，区别仅仅是“问题更少”的程度不同罢了。
软件就是麻烦一堆。
而且我们不能够也不愿意把电脑一关走为上计。
给我们带来挫败和束缚的软件，也用更多功能、更快更好的工作与生活方式来引诱我们。
无路可回。
我们对软件的需要，远甚于对它的仇恨。
所以我们梦想着得到更新更好的东西。
在现代软件研究领域多有建树的专家弗里德里克?布鲁克斯（Frederick Brooks）在1987年写了一篇题为《没有银弹（No Silver Bullet）》的著名论文。
布鲁克斯在论文中称，无论编写计算机程序是如何地令我们倍感挫败，也永远无法找到一种魔法般的突破——我们只能期待渐次前行。
布鲁克斯的观点难以辩驳，但也难以接受；参加计算机业界会议或是浏览程序员网站时，你总会遇到一些坚称其错的人。
有些人梦想炸毁今天的整座软件大厦，替以某种全新之物。
有些人则只一味盼望找到不太顽固、更能响应人类愿望和行为流程的程序员，盼望能得到召之即来、挥之即去的软件，盼望得到足堪依赖的代码。
梦之所寄，行之所为——地狱之门就此洞开。

第1章　死定了[2003年7月]迈克尔?托伊（Michael Toy）双手托腮，将下巴埋入手腕之间，瞥向他的PowerBook电脑，嘴里嘟嘟哝哝：“约翰死定了。
在下个版本出来前，他得连干五百个钟头……凯蒂死定了。
她得干到天荒地老。
布莱恩越发要完蛋。
而且他只有一半时间了。
至于安迪嘛，只有安迪没事。
他的工作列表上可没有成百上千的任务。
”其实他们看起来还好。
这是一个夏日，程序员们围坐在加州贝尔蒙特（Belmont）的一张普通会议桌旁，倾听着经理的发言。托伊是位高个子男士，有着一副铁石心肠和一条马尾辫，不过看起来他正陷入沮丧之中，历数程序员们是多么地落后于进度。
这是2003年7月17日，他感到自己也死定了，只有不到两个月的时间，不可能在下一版本预定完成时间到达之前搞掂一切。
“谁的工作列表不够时间完成，就和我一起重新过一下列表。
”

媒体关注与评论

“每个有志于开发畅销产品的程序员都值得耐心去品味这个故事。
”——CSDN总裁 蒋涛 “《梦断代码》是一本代码史。
”——DoNews制作人，千橡集团副总裁 刘韧 “《梦断代码》乃承Tracy Kidder《新机器灵魂》血脉之初见者，融技术眼界与叙事功夫于一炉，实多年未见之奇书。
读毕，当可了解软件术士辈之所为。
”——詹姆士·菲罗斯（James Fallows），《大西洋月刊》 “技术人员爱把复杂的问题形容为非凡。
司各特·罗森伯格选取了极度非凡的主题，并使之浅显易懂。
他盛赞写代码的人，但也坦承他们如常人般复杂而有缺陷。
《梦断代码》实在是一流的报道和著作。
”——丹·基尔默（Dan Gillmor），公民媒体中心主席及《自媒体（We the Media）》作者
“《梦断代码》探索了编程如何鼓舞和破坏人类对新工具的创造，既令人迷醉，又教人冷静。
本书文笔优美，专为对创造与革新之根源感兴趣者——无论是开发者还是其他人——所写。
”——史蒂芬·强森（Steven Johnson），《开机（Everything Bad Is Good for You）》及《幽灵地图（the Ghost Map）》作者 “司各特·罗森伯格放胆踏入非程序员未敢涉足之地：那人类想象苦苦变身为代码的旋风中央。
在《梦断代码中》，他华丽地将一家初创软件公司的故事与我们理顺编程过程的（无穷尽）努力结合起来讲述。
”——艾伦·乌曼（Ellen Ullman），《走近机器（Close to the Machine）》作者

编辑推荐

当一本书与墓志铭联系在一起的时候，你可以想象得到《梦断代码》的奇异诡秘。
当一本书从翻译到即将出版，期间就已备受无数人的关注和期待，你应该可以感受得到《梦断代码》的震撼与强劲。
而当你有机会能够阅读这本书的部分章节时，你会在手不释卷之余长叹：的确是IT书丛中难得一遇的奇书妙谈？
。
《梦断代码》之奇首先于它的原作者Scott Rosenberg，1981年从哈佛大学毕业，1995年和人共同创办了Salon网站，此后担任其首席技术编辑达数年之久，还负责技术工作。
2003年，开始写一本有关软件开发及其问题的书。
为了写这本书，他2005年离开了Salon，对OSAF的Chandler项目进行田野调查，跟踪经年，试图借由Chandler的开发过程揭示软件开发中的一些根本性大问题。
这一系列工作的成果就是这本《梦断代码（Dreaming in Code）》。
书中极尽引经据典、推敲字句之能事，夹叙夹议，奇闻轶事和以精论妙谈，300来页一本书，触及软件开发中多个根本大题目，叫人看得心惊肉跳！
而它的译者韩磊先生同样也是一位中国网络媒体的奇人，“云南昭通人氏。
广州呆十年，北京飘三载。
学越南语，犯校对病，玩计算机，搞互联网，好长短句，嗜摄影术。
年过卅载有多，体重百斤不足。
”寥寥数语的自我勾勒，奇人气息扑面而来。
据博文视点总经理郭立说：“韩磊希望把翻译费全部换成图书，赠送给国内的CTO层面的人物。
”奇人做奇事似乎不足为奇了，作为IT媒体人，韩磊应该更多的是期待这本书能给中国IT界的人士带来些什么吧。
奇人著、译奇书，都是源于对开源奇梦的追求和热爱，“这里躺着一个野心勃勃的开源项目。
它曾立志超越Outlook，最后却无疾而终。
慷慨的Mitch Kapor带给它生命，又把命脉从它身上取走。
许多程序员以心血养育它，惜乎全不见成效。
它是温室中的花儿，有过绚烂的梦想，还未绽放即已枯萎。
那软件的花园中，还有多少会渐次凋零呢？
”《Dreaming in Code》在“叙”的部分就是写这个项目的。
《梦断代码》的副标题很长，也很吸引人：Two dozen programmers, three years, 4,732 bugs, and one quest for transcendent software（两打程序员，三年，4,732个缺陷，只为打造卓越软件）。
Mitch Kapor和他的程序员们曾经打造过最伟大的软件，现在他们有了近乎奢侈的资源，为了“用代码改变世界”的梦想，共襄盛举。
可惜他们遇到了“软件时间”的黑洞，从2001年开始，泥足深陷、举步维艰。
Scott Rosenberg从项目一开始就贴身跟踪，记录下Chandler项目开发过程中的点点滴滴。
如果《梦断代码》仅及于此，就不过是一篇尚可一观的“报告文学”。
幸好Scott Rosenberg的主要目标是揭示软件开发中的普遍问题：为什么做软件这么难。
在描写Chandler项目进展的段落之间，温伯格、布鲁克斯等大师级研究者，Linus Torvalds和Joel等实践者纷纷“出来”现身说法，还有来自各机构、媒体、作者的大量资料，从“软件时间”入手，兼及各种方法论、积木式编程、软件设计、项目管理，所有这些都是最后一个大话题的铺垫。
在《结语》部分，Scott Rosenberg提出了这个大话题：人工智能有没有可能超越人类？
在这个问题上，Mitch Kapor打了个20,000美金的大赌。
他打赌，到2029年为止，都不可能造出能通过图灵测试的机器。
博文视点正是感应到了这本奇书的奇特魅力，特将它作为“博文五年，以书为证”的特别献礼，隆重呈献于读者面前，呈献的更是一道绚丽多姿的风景线，期待读者将书中还未绽放即已枯萎的绚烂梦想

或野心勃勃却无疾而终的开源项目与自己在中国IT领域多年打拼的经验很好的融合，得到一些经验和启迪。
毕竟人人都是“我有一个梦”，程序员的梦想则更多绮丽和追求，虽历经磨难，但仍奋力创造，这正是《梦断代码》的意义所在，也何尝不是人生的意义所在呢？

版权说明