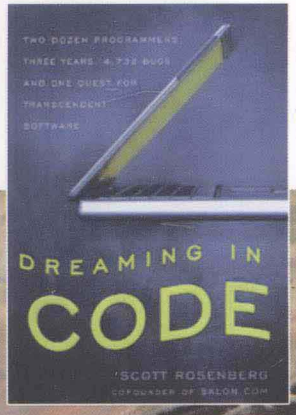
Broadview
www.broadview.com.cn

传世经典书丛
Eternal Classics

梦断代码

[美] Scott Rosenberg 著
韩磊 译



两打程序员，3年时间，4732个bug，只为打造超卓软件

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

梦断代码

DREAMING IN CODE

[美] Scott Rosenberg 著

韩磊 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

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 项目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。

版权贸易合同登记号 图字：01-2007-2963

图书在版编目（CIP）数据

梦断代码 /（美）罗森伯格（Rosenberg,S.）著；韩磊译. —北京：电子工业出版社，2011.6
（传世经典书丛）
书名原文：Dreaming in Code：Two Dozen Programmers，Three Years，4,732 Bugs，and One Quest for Transcendent Software
ISBN 978-7-121-13569-9

Ⅰ. ①梦… Ⅱ. ①罗… ②韩… Ⅲ. ①软件开发 Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字（2011）第 089575 号

责任编辑：徐津平
文字编辑：江 立
印 刷：
装 订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×980 1/16 印张：21.75 字数：274 千字
印 次：2011 年 6 月第 1 次印刷
印 数：4000 册 定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

悦读上品 得乎益友

孔子云："取乎其上，得乎其中；取乎其中，得乎其下；取乎其下，则无所得矣"。

对于读书求知而言，这句古训教我们去读好书，最好是好书中的上品——经典书。其中，科技人员要读的技术书，因为直接关乎客观是非与生产效率，阅读选材本更应慎重。然而，随着技术图书品种的日益丰富，发现经典书越来越难，尤其对于涉世尚浅的新读者，更为不易，而他们又往往是最需要阅读、提升的重要群体。

所谓经典书，或说上品，是指选材精良、内容精练、讲述生动、外延丰盈、表现手法体贴入微的读品，它们会成为读者的知识和经验库中的重要组成部分，并且拥有从不断重读中汲取养分的空间。因此，选择阅读上品的问题便成了有效阅读的首要问题。当然，这不只是效率问题，上品促成的既是对某一种技术、思想的真正理解和掌握，同时又是一种感悟或享受，是一种愉悦。

与技术本身类似，经典 IT 技术书多来自国外。深厚的积累、良好的写作氛围，使一批大师为全球技术学习者留下了璀璨的智慧瑰宝。就在那个年代即将远去之时，无须回眸，也能感受到这一部部厚重而深邃的经典著作，在造福无数读者后从未蒙尘的熠熠光辉。而这些凝结众多当今国内技术中坚美妙记忆与绝佳体验的技术图书，虽然尚在国外图书市场上大放异彩，却已逐渐淡出国人的视线。最为遗憾的是，迟迟未有可以填补空缺的新书问世。而无可替代，不正是经典书被奉为圭臬的原因？

为了不让国内读者，尤其是即将步入技术生涯的新一代读者，就此错失这些滋养过先行者们的好书，以出版 IT 精品图书，满足技术人群需求为己任的我们，愿意承担这一使命。本次机遇惠顾了我们，让我们有机会精心推出"传世经典书丛"。

在我们眼中，"传世经典"的价值首先在于——既适合喜爱科技图书的读者，也符合专家们挑剔的标准。幸运的是，我们的确找到了这些堪称上品的佳作。丛书带给我们的幸运颇多，细数一下吧。

得以引荐大师著作

有恐思虑不周，我们大量参考了国外权威机构和网站的评选结果，并得到了外方出版社的专业支持，又

进一步对符合标准之图书的国内外口碑与销售情况进行细致分析，也听取了国内技术专家的宝贵建议，才有幸选出对国内读者最富有技术养分的大师上品。

向深邃的技术内涵致敬

中外技术环境存在差异，很多享誉国外的好书未必适用于国内读者；且技术与应用瞬息万变，很容易让人心生迷惘或疲于奔命。本丛书的图书遴选，注重打好思考方法与技术理念的根基，旨在帮助读者修炼内功，提升境界，将技术真正融入个人知识体系，从而可以一通百通，从容面对随时涌现的技术变化。

翻译与评注的双项选择

引进优秀外版著作，将其翻译为中文供国内读者阅读，较为有效与常见。但另有一些外语水平较高、喜好阅读原版的读者，苦于对技术理解不足，不能充分体会原文表述的精妙，需要有人指导与点拨。而一批本土技术精英经过长期经典熏陶及实践锤炼，已足以胜任这一工作。有鉴于此，本丛书在翻译版的同时推出融合英文原著与中文点评、注释的评注版，供不同志趣的读者自由选择。

承蒙国内一流译(注)者的扶持

优秀的英文原著最终转化为真正的上品，尚需跨越翻译鸿沟，外版图书的翻译质量一直屡遭国内读者诟病。评注版的增值与含金量，同样依赖于评注者的高卓才具。好在，本丛书得到了久经考验的权威译(注)者的认可和支持，首肯我们选用其佳作，或亲自参与评注工作。正是他们的参与保证了经典的品质，既再次为我们的选材把关，更提供了一流的中文表述。

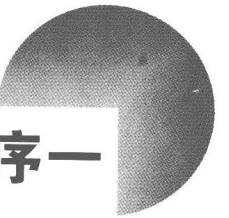
期望带给读者良好的阅读体验

一本好书带给人的愉悦不止于知识收获，良好的阅读感受同样不可缺少，且对学业不无助益。为让读者收获与上品相称的体验，我们在图书装帧设计与选材用料上同样不敢轻率，惟愿送到读者手中的除了珠玑章句，还有舒适与熨帖的视觉感受。

所有参与丛书出版的人员，尽管能力有限，却无不心怀严谨之心与完美愿望。如果读者朋友能从潜心阅读这些上品中偶有获益，不啻为对我们工作的最佳褒奖。若有阅读感悟，敬请拨冗告知，以鼓励我们继续在这一道路上贡献绵薄之力。如有不周之处，也请不吝指教。

电子工业出版社博文视点

二〇一〇年十二月



推荐序一

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

推荐序二

软件技术日新月异，软件项目领导的艺术也是日新月异。如同开发者在不断把自己的代码当作艺术品来雕琢一样，组织好一群天才开发者，则更是艺术中的艺术。

软件开发者不是堆砌代码的工人，也无法安于命令式的任务布置。这些天才开发者们，低调、寡言、有独立的自我意识，他们并不迷恋于成为焦点的感觉，但却十分沉迷于自己认为伟大的创造。天才们在一起的合作，貌似创意无穷，实则合力有限；貌似独当一面，实则整合艰难。等等诸多的问题，似乎成了很多国内外软件企业的通病。

真正的领导者，实际上早已被要求有着化不可控为可控、化不同愿景为共同愿景、激发天才们最大潜能的能力。这对于整个项目而言，这样的图景绝不应该止于一个美好愿望，尽管难上加难，但在所有开发者和经理们的努力下，我坚信一定可以实现。

Discuz!创始人 康盛创想（北京）科技有限公司总裁 戴志康

作者的话

这世上有太多给软件开发者看的指导书。这本书并不是其中之一。我自己只是个入门级程序员，怎能在专家们的面前班门弄斧呢。而且，如果我的研究在构造更好软件方面发掘出了前所未知的创新或洞见，何不径自去找投资商，何苦还要呈示给读者。

我期望程序员喜欢这本书，但本书对其他人也不无益处。它提出问题，讲述故事。为什么好软件如此难做？因为看来无人能有确定答案，即便是现在，21 世纪伊始，计算机时代开始 50 年之后，我以亲涉踏勘的方式写出这个做软件的故事——关于一队人马并肩托起代码大石、欲将其推上山顶，虽历经磨难，但仍奋力创造某种有用、丰富且持久之物的故事。

司各特·罗森伯格（Scott Rosenberg）

目录

第 0 章

软件时间

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，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[①]，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[②]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

[①] Black code，指原来程序逻辑中没有的部分，采用类似侵入的手段加进去的代码。

[②] 新月沃土指中东两河流域及附近一连串肥沃的土地。包括今日的以色列、西岸、黎巴嫩、约旦部分地区、叙利亚，以及伊拉克和土耳其的东南部。由于在地图上形似一弯新月，所以美国芝加哥大学的考古学家詹姆士·布雷斯特德（James Henry Breasted）把这一大片肥美的土地称为“新月沃土”。

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！

25 年后，2000 年 5 月，我坐在旧金山的一间办公室里，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！**计算机械协会（The Association for Computing Machinery），计算机领域中的 ABA[①]或 AMA[②]，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[③]。

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 {
```

① American Bankers Association，美国银行家协会

② American Medical Association，美国医药协会

③ Rosetta Stone，1799 年法国士兵在埃及发掘出来的象形字黑石碑。也是美国著名的外语教程品牌。

```
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。

图 0-1　Salon 网站

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）这样感慨。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与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。

图 0-2　建设中的海湾大桥

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。即便没有微软的Windows，人类文明也会同样延续，对吧？然而，软件并不像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