《软件随想录 卷2》

图书基本信息

书名：《软件随想录 卷2》

13位ISBN编号：9787115387729

出版时间：2015-4

作者：[美] Joel Spolsky

页数：288

译者：阮一峰

《软件随想录 卷2》

内容概要

《软件随想录 卷2》是一部关于软件技术、人才、创业和企业管理的随想文集，作者Joel Spolsky以诙谐幽默的笔触将自己在软件行业的亲身感悟娓娓道来，观点新颖独特，内容简洁实用。全书分为36讲，每一讲都是一个独立的专题，分别介绍了作者在人员管理、程序员成长规划、软件设计细节、具体的项目管理、如何编程以及如何创办和经营软件公司等方面的独到见解。
《软件随想录 卷2》从不同侧面满足了软件开发人员、设计人员、管理人员及从事软件相关工作的人员的学习与工作需要。

《软件随想录 卷2》

作者简介

《软件随想录 卷2》

书籍目录

《软件随想录 卷2》

精彩短评

1、程序员的阅读感受可能会更深一些～
2、1
3、上下两集给人的干货吸收时间完全不同，我花在下集的时间要远大于上集
4、我觉得对我最有用的就是「简洁」的那一段——使用需要简单，但是功能该有的一个都不能少
5、Joel Spolsky 實在太會玩了！！！

P.S. 看完才發現，和《軟件隨想錄》一樣內容嘛
6、还是略冷饭了。
7、跟第一本差不多，值得看
8、对一个观点的印象更深了：最好的程序员，生产效率会比普通程序员高十倍以上。
9、应该是还没看就给汪qin然了，然后自己去图书馆借了本旧版
10、推荐第一本
11、里面关于程序员的工作和软件公司那两部分实在是太认同了.
12、并没有看完，只挑了感兴趣的文章阅读。从第二部分写给未来程序员的建议了解到软件工程与计算机科学的差异，也明白了自己距离合格的程序员有多远，学习底层原理与思想是最重要的，然后学习点微观经济学也是必需的，最后写作可以帮助自己更好梳理知识并掌握它们。多学习并多实践然后输出
13、内容新意不多，基本还是传统的观念，另外多次重复Excel的经验，难道没有更多样深入的例子吗！
14、给用户太多选择的话，他们会生气的。
15、就是以前那本的第 2 版啊，还弄成什么卷 2…回顾了两篇：「学校只教 Java 的危险性」、「要挣钱，就别怕脏」[doge]

精彩书评

1、第一部分 人员管理- 怎么寻找优秀的程序员提供优良的工作环境，我觉得作者还蛮激进的，他创办的公司（在纽约）人均面积 40 平，每一个程序员都有自己的独立办公室（必须有门），而且电源插座不少于 20 个，必须有窗，给程序员休息眼睛用，为此特地请了厉害的建筑师来帮忙设计。相比较之下，工资不是第一考虑的要素，不是说不重要，类似于不患寡而患不均，这样。以上是吸引优秀程序员的方法，但是也要主动出击。广撒招聘贴效率很低下的，因为优秀的程序员很少出现在人才市场，常年在找工作的人往往是因为干不好，因此收到的很大概率是这些因为才能不够而往所有公司投简历的人。一个改进办法是，在技术圈子内投招聘贴，类似于精准投放。更加有效但是很难做到的做法是，围绕自己公司创办一个社区，在上面讨论相关技术问题，吸引有才能的人，因为被吸引来的是对公司感兴趣的人，因此如果他们看到招聘启事，会有更大可能应聘。通过奖励员工推荐不是一个很好的办法，如果不能保证一定录用，员工是不会推荐朋友来面试的，要不然多尴尬，但是没有面试就录用风险太大，除了技术以外，是否可以融入公司环境也是需要考察的。如果再加上奖金的诱惑，员工推荐的人会更加良莠不齐。因此员工推荐只可以用于减少复杂的面试环节，但是不能作为招聘常态。还有一个办法是，招收实习生，在实习的过程中发现优秀的人才并且留住他们，相较于只面试一两次的人，实习生有长达一年的熟悉阶段，留下来的人会更符合期望。于是问题变成如何发现并且招收优秀的实习生。因为程序员的特殊性，真正对编程有兴趣并且有才能的人，可能高中甚至初中就已经开始接触并且逐渐掌握相关技能以及思想。怎么把他们区分出来。和各个高校的教授或者学生保持联系，拜托他们帮忙推荐。筛选实习生的 GPA，因为是综合表现，如果不能很好的处理不感兴趣／枯燥的课程，那有很大可能遇到的枯燥的工作也没有办法很好的应对。提前招收实习生，从大一、大二就开始邀请他们到公司实习，并给他们留下深刻的好印象，即使最终没有留下来，也会在同学中口口相传，是无形的广告。- 怎么管理程序员三种管理方法，军事化管理、经济外驱管理、内驱管理。毫无疑问最后一种办法是最理想的。军事化管理最终很容易变成抽风式（hit-and-run）管理，因为只有真正写代码的人才了解全盘情况，可以做出正确的决策，被管理人员下达强制命令后，很容易产生抵制，而管理人员又不可能长期盯着某一个程序员。经济外驱管理就是常见的绩效，因为没有很好的办法来衡量一个程序员的输出，不管制定什么样的规则，都会有相应的糊弄办法，本来程序员发自内心的改进，因为金钱／奖励的介入，就会变质。内驱管理最好的达成办法，就是让程序员知情，不是简单的下达命令，而是和他商量要达到这个目标该如何做，往往程序员会给出更好的办法。第二部分 写给未来程序员的建议很可惜我已经毕业（很久）了，回想大学时候的无所事事和若有所失，如果当时读到这本书该多好的。作者吐槽现在的大学只教授 Java 过于简单，虽然这正是 Java 设计的一个特点，它不需要程序员操心太多事情，但是这样做的后果是，没有办法筛选无法胜任程序员工作的人，他回想了自己当时试听了一节动态逻辑的课，随后便断了读研的念想，并且为此庆幸。推荐了《SICP》，巧合的是我现在正在看这门课的参考书，感觉确实讲得很基础，不是 101 那种，而是基石的那种基础，相对应的，确实好难，有啃不下去的危险。哦，GPA 似乎是这部分谈到的，要保证自己的 GPA 足够高。练习写作，那些流行起来的技术，很大一个原因是有很棒的布道人，扩大影响最起码的步骤是开始交流。因此如果有相关技术写作的课程，比如每两周就要交一篇论文，一定要选。学好 C 语言，因为它是程序员的共同语言，只有明白它的指针，才有可能看懂 Linux 内核，而编译器和操作系统是基础知识，不懂的话，就像不了解解剖学的医生。学好微观经济学，程序员做的很多决定会直接影响到面向市场的软件，有经济头脑是一件好事。选修有大量编程实践的课程。不解释。（啊，作者还是用了蛮大篇幅讲得，就是我单纯不想解释）不要因为担心找不到工作而不选，优秀的人永远不需要担心找不到工作。找一份好的暑期实习，真正的工作和学校中的作业是不一样的，另外作者在这部分主要是为了给自己公司打广告。第三部分 设计的作用以微软和苹果的字体设计为例，有时候用户只是选择自己熟悉的产品而已，没有什么特别的理由。但是也不能因此得出设计无用，真正打磨、寸土必争的软件，才有可能是优秀的软件。但是又以《梦断代码》为例，警告到，不要陷入过大的设计当中，只注重所谓的全景，并且高估自己的能力，而细节落实没有跟上，导致功亏一篑。然后呢，从代码实现再回到用户呈现，不要过于琐碎，以 vista 为例，当用户要离开电脑时，从锁屏到关机，给了 9 种选择，这种时候用户就会懵，除了较真的程序员，一般的用户是不开心的。关于软件的易用性，当然很重要，但是分情况说，如果这个软件非常有用，那么它是否易用就无所谓了，因为反正你都要用，但是假如在同等功能下的软件，易用性就会变成决定性因素。不过作者觉得这没什么好讨论的，他关心的是下一

个层次的内容，社会化界面。关于这个概念，有一种软件，是人与人打交道的中介，比如 email，论坛这些，那如何设计这些软件，就涉及到社会化界面，因为这些软件即使易用性再棒，那都只是单机的，没有用，更重要的是，在和别人交流的过程中，你的软件如何发挥更好的作用。作者举了一些例子，比如过滤垃圾邮件、屏蔽论坛广告（为了反驳那些“自由言论”说，作者说要维护高质量的社区必须这样做，要不然就是对提供价值的用户的不尊重，并且他们很可能因此不再光顾你的论坛），关于这个领域，作者认为是很新的一片天地，需要人类学家、社会学家的理论来做支撑。以此为基础，作者聊了一下搭建社区的 FAQ，比如回复和发帖的按钮放在最下方就是为了引导用户先看完别人的言论，防止重复发言；不去发社区警告，因为会犯规的人不会去看，而你在浪费有价值的人的时间和心情如此等等。第四部分 管理大型项目哈，聊了臭名昭著的“ Web 标准”，IE 是如何作茧自缚的，科普了为什么不可能有一个标准，只要网页通过了这个标准，就可以在所有浏览器上正常显示。不同的标准、不同的实现的多对多困境。由此讲到微软的 office 软件，为了适配早期硬件性能，它是二进制的文件格式，内部就是一个目录系统，再加上后来加新功能，向后兼容，以及横向的兼容其他非自家格式，导致 office 文件就是无比复杂的巨无霸。关于如何应对，一种是使用可交换的文本格式，比如 HTML，用更简单的 office 支持的格式，比如 csv，或者，如果你还要更进一步，那就调用 office 提供的库，如果你想自己编写，就只能祝你好运了！讲了这么多为了兼容而慢慢变成巨型项目的例子，但其实大家（似乎是写给创业者的）日常情况是，实现某一个功能的时候，可以选择调用某一些库，拼装一下，搞定。这样做是另外一个极端，但真正让你挣钱的在于解决“麻烦事”，比如提供不同平台的安装包。第五部分 编程建议第一大问题，你如何为自己的项目预估时间，安排日程。以他们公司提供的一个软件为例，提出了寻证式日程规划（evidence-based scheduling），大概就是记下你完成每一项工作的时间，画一个图表出来，总结你的预估时间和实际时间的关系，最终稳定下来以后，你就知道下一个项目大概要花多少时间了。他们的软件可以帮你记录时间、生成图表。关于战略问题的通信之六，是个系列文章，前面的在卷一。大概是畅谈系列。因为摩尔定律的存在，可能花大块时间优化软件性能是不必要的，并且进一步的，可以开发现在情况下硬件无法很好支持但是界面一流的软件。关注跨平台的编程语言，这样可以节省程序员的时间，但是不要打沙箱的主意，它不能利用系统的长处，过去没有成功，以后也行不通。完善的互动性和用户界面标准，这个小节我没有搞懂，似乎这里的用户是指程序员，给出的例子也是 SDK，大概就是提供 SDK 的人提供了平台最终大一统。编程语言部分，大概就是 SICP 课上讲到的选择语言时要问的三个问题的第二问，你的语言如何做抽象（另外两问是，你的语言有哪些基本元素（primitive element），如何做组合），关于如何做抽象，又分为两问，如何对不同的数据做抽象，得到函数，如何对不同的过程做抽象，得到高阶函数，另外一个小问题，是否支持匿名函数，函数是否是第一等公民。并且夹杂了一些例子，比如 map 和 reduce 以及 mapreduce。到最后作者稍微站了一个队：“最有生产效率的编程环境是那些允许你在不同层次上进行抽象的编程环境”。而真正开始下手写代码以后，作者谈的问题是，要把相关的代码放在一起，尽量达到看功能如何实现时不用跳来跳去的效果，因此，宏是糟糕的功能，异常处理也是，你没有办法一眼就知道哪里会运行哪里不会运行。然后涉及到在一屏放不下的状况下，如何知道代码是相关的，最直接的办法就是变量命名了，力推匈牙利命名法，为他正名道，因为文档中的笔误和他人的误读，才会变成现在的样子，它原来的样子是给变量名加上种类（kind）而不是类型（type），比如 xl 和 xw 分别代表“相对于页面的横坐标”（x relative to layout）和“相对于窗口的横坐标”（x relative to window），而 cb 表示“字节个数”（count of bytes）这样，你在操作的时候，就知道 xl = cb 这种没有语法错误的句子肯定也是错误的。第六部分 开办软件公司有两篇给别人的书写的前言收录进来，一个是讲你要对做生意有兴趣，一个是讲你要对经营公司有兴趣。然后一篇叫做飚高音，其实还是讲招优秀的程序员，然后放手让他们去干，用飚高音来做标题，是因为你唱不上去就是唱不上去，一些歌剧你就是没有办法演，10 个普通程序员也不能完成 1 个优秀程序员所能完成的事情。第七部分 经营软件公司炫耀了一下自己公司的办公室。用第三方库的时候，要保证如果出错，有办法应对，这个办法往往是，有这个第三方库的源码，并且自己有能力修改，否则，还是乖乖自己实现。不要被当下的简约风（这股风刮了好久哦，这本书是零几年的，感觉现在进化成性冷淡风了）所迷惑，以为简约就是只有一两个功能，那样是在糊弄客户，最终会被客户抛弃。对，代码是赶时间写出来的，总是一团糟，但是不要老是幻想推倒重来。更好的解决办法是，不添加新功能，保证 100% 可运行的基础上，用心整理一下。这样可以随时交付代码，而且不用去踩推倒重来会遇到的坑。beta 测试需要很长时间、很多用户。优质客服很重要，不要外包，而是把技术支持做成像对待空难一样，出现错误以后，从本质上解决

问题，保证不会以同样原因再次出现，对待投诉客户要上心，当做培养忠实用户的一个机会。同时，给客服一条好的晋升通道，比如作为管理层的第一年，因此可以招到优秀人员。第八部分 发布软件发布日期：根据软件规模、面向受众决定发布频率，如果是公司内部或者小型软件，可以持续发布，听取使用者的意见，持续改进；但是上架销售的软件就要在 beta 测试完成以后，确定软件功能完善，可以给人留下良好印象以后发布，否则别人再也不会光顾你的软件，更新频率大概在 12～18 个月，太久了会缺钱，太频繁会让客户认为没有什么改进而不断收钱。超级庞大的系统则是 3 到 5 年的频率，因为要保证兼容性，要做大量测试。软件定价，呵呵。好麻烦，不想讲。（作者讲了太多，我不想总结了）第九部分 修订软件遇到 bug 要刨根问底，不要流于表面。开发软件之前，要先确定优先顺序，这样如果时间来不及，可以砍掉优先级低的功能，而且被砍掉的功能往往会被证明是多余的，再也不需要开发。至于如何确定优先级，开会啊。召集各个岗位的人，开发、设计、测试、客服、销售甚至客户，每个人列出自己想要的功能，集中以后，开发给所有功能一一评判出开发难度，作为成本价，然后再公开，每人手上发 50 元，好了，可以去购买功能了，最终，各个功能的售出总额除以成本价，就是优先级。done。

《软件随想录 卷2》

版权说明