《交互设计之路》

图书基本信息

书名：《交互设计之路》

13位ISBN编号：9787121021626

10位ISBN编号：7121021625

出版时间：2006-3

出版社：电子工业出版社

作者：库帕

页数：230

译者：Chris Ding

《交互设计之路》

内容概要

本书是基于众多商务案例，讲述如何创建更好的、高客户忠诚度的软件产品和基于软件的高科技产品的书。本书列举了很多真实可信的实际例子，说明目前在软件产品和基于软件的高科技产品中，普遍存在着“难用”的问题。作者认为，“难用”问题是由这些产品中存在着的高度“认知摩擦”引起的，而产生这个问题的根源在于现今软件开发过程中欠缺了一个为用户利益着想的前期“交互设计”阶段。“难用”的产品不仅损害了用户的利益，最终也将导致企业的失败。本书通过一些生动的实例，让人信服地讲述了由作者倡导的“目标导向”交互设计方法在解决“难用”问题方面的有效性，证实了只有改变现有观念，才能有效地在开发过程中引入交互设计，将产品的设计引向成功。

本书虽然是一本面向商务人员而编写的书，但也适合于所有参与软件产品和基于软件的高科技产品开发的专业人士，以及关心软件行业和高科技行业现状与发展的人士阅读。

《交互设计之路》

作者简介

Alan Cooper，作为20世纪70年代中叶的一名软件发明家，Alan Cooper坚信必然存在一种更好的方式创建软件。新的方法，应该通过应用“使用者第一、硅片第二”原则的设计和工程过程，使软件使用者从让人厌烦、困惑、不恰当的软件行为中解放出来。采用这种方法，技术团队能够在第一时间做对事情，因而，也就能更快地创建更好的产品。他的信念结出了硕果。1990年，他创建了Cooper公司，一家技术产品设计公司。今天，Cooper在软件设计方面的创新手法已经被认可为行业标准。
在Cooper打开面向商业的大门后的十年时间里，这家旧金山公司向诸如Abbott Laboratories、Align Technologies、Discover Financial Services、Dolby、爱立信、富士通、富士通Softek、惠普、Informatica、IBM、罗技、Merck-Medco、微软、Overture、SAP、SHS Healthcare、Sony、Sun Microsystems、Toro公司、 Varian，以及VISA公司提供了创新的、以使用者为焦点的解决方案。Cooper团队提供他们

《交互设计之路》

书籍目录

《交互设计之路》

章节摘录

第一版的前言　　研究交互设计商务案例的书　　我原本想写一本与本书截然不同的书：一本讲述如何进行交互设计的技术类书籍。但是，在1997年5月我去托斯卡纳 的探亲旅途中，我的好友Don McKinney和Dave Carlick曾谈及此事，他们说服我应该首先为商务人员编写一本书。　　他们知道我在计划编写一本关于如何进行交互设计的书。他们尽管表示了鼓励，但同时也怀疑交互设计有哪些需求。他们要我先写一本能够让他们确信交互设计有价值的书。他们的想法很有吸引力，但是当时我没有把握写出他们所期望的书。　　一个深夜，在可以眺望到佛罗伦萨的土黄色别墅的阳台上，我和Dare、Don进行着诚挚的交谈。桌子上是几个喝空的葡萄酒瓶，还有一些面包、奶酪和橄榄。天空中星光闪烁，萤火虫轻盈地飞过草坪，远处托斯卡纳区政府的圆顶古代建筑的灯光闪烁不定。Dave再一次向我建议，我应该推迟编写如何进行交互设计的书，而应先“列举出一些使用了交互设计的商业案例”，让大家对交互设计信服。　　我强烈地抗议道：“但是Dave，我不知道如何写你说的那本书。”我掰着手指头一点一点地抛出了我的理由：“那意味着我不得不解释目前流行的开发过程是如何的混乱；我得解释软件公司是如何因低效率的开发而浪费金钱；我得解释客户们为什么这么容易‘移情别恋’；我得解释一个好的交互设计应如何解决这类问题。”　　Dave打断我的话，轻松地说：“这些都可以是书中的某些章节呀，Alan。”　　他的话让我无话可说。我意识到我在老调重弹，Dave是对的。一本“面向商务人员”的书比解释“如何做交互设计”的书有更迫切的市场需求，更合时宜。Dave和Don最终让我相信，我确实可以写这样的书。　　懂商务的技术专家/懂技术的商人　　21世纪，能够成功的人是：理解商务的技术专家或懂技术的商人。这本书是为他们而写的。　　懂技术的商人知道，他自己的成功依赖于获取高质量信息的能力和有效地处理这些信息的能力。另一方面，懂商务的技术专家是具备商业眼光的工程师或科学家。他们具有敏锐的商业眼光，懂得信息的威力。这两类新人将主宰当代商业。　　我们可以将所有的商人分为两类：一类是精通高科技的人；另一类是正在走向破产的人。作为商人的企业主管再也不能将处理信息的工作托付给专业人员。商务就是处理信息！企业主管需要在处理信息方面，而不是在制造系统方面，有别于人。如果一个企业制造某种产品，那么这种产品中是否含有微型芯片，就是成功与否的关键所在。如果提供某种服务，则胜算可能来自于这种服务是通过计算机化的工具提供的。因此试图识别商务是否依赖于高技术与识别商务是否依赖于电话的商务一样，没有任何意义。因为高科技已经渗透到了每一种商务活动，数字信息已成为我们每天工作中的脉搏。　　有一句这样的话，“人总会犯错，要振兴只有靠电脑。”低效率的机械系统可能会在加工的每个部件上浪费几个美分，而错误的信息处理则有可能毁掉一个企业。基于软件的产品——制作这些产品的工程师们——对一个企业的影响力是不可估量的。　　可悲的是，数字化工具非常难于学习、使用和理解，经常使我们达不到的目标。我们因此而浪费金钱、时间和机会。一位理解商务的技术专家或懂得技术的商人可能制作或使用软件产品，或者是两者兼而有之。因此，拥有更好、更容易掌握、更容易使用的高技术产品，会为你个人及企业带来更大的利益。其实，开发更好的产品并不需要更长的时间，也不需要更高的成本。具有讽刺意味的是，它们本来不必如此困难，事实上却如此困难。那是因为我们采用的软件产品开发过程是过时的，需要修正。长期根植于我们头脑中的传统的错误观念使我们得不到好的产品。本书将向我们展示如何要求并得到我们渴望以久的更好的产品。　　本书的观点不难理解：在开始编制软件之前做好交互设计，我们就可以创建强大而令人愉悦的软件产品。目前广泛流行的观点正好相反，我们已经不按流行的观点做了。设计有交互的基于软件的产品是一种专业，像构造它们一样需要很多技能，并要付出相当的努力。　　由于我已经选择了写一本基于商务案例的书，而不是写一本关于如何设计的技术类书籍，估计阅读此书的交互设计师们或许会失望，因此我请求你们能够予以谅解。本书中仅用了少量篇幅提及交互设计方法的基础知识及其核心（主要在本书的第4篇）。但这些篇幅足以证明：①这样的设计方法确实存在；②它对任何项目都适用；③任何人都能看到它的好处，不管这个人的技术背景如何。前言　　最近，我在一家世界级的技术公司见到一位高级主管，他的头衔是“易用性”副总裁。他负责相当数量的大小软件产品。他很精干，并在正统的“人机交互”领域有所建树。他和他的公司一样，崇尚“可用性”（usability）方法，即通过单向隔离窗观察并测试人们对产品的反应。但是他开始谈论设计而不是测试，谈论人物角色而不是用户。他说，他的公司已经完全停止了开发后进行可用性测试的方法，现在采用的是事先设计的方法。他进一步肯定，他们公司所有受过观察用户技能培训的人员正在接受人种学研究方面的培训。　　这位主管和他所在公司态度的转变，象征着本书首次出版后短短的5年时间内，软件业界发

生众多巨大的变化。本书既是一本革命的宣言，也是一本提供规范的指南。无数中层产品经理在阅读本书首版后给我发电子邮件，说明他们为自己部门的资深高级主管们购买本书的原因。另一方面，软件制作者和大学将本书第4篇的三个章节：“交互设计”作为实施基于人物角色的目标导向设计的主要参考指南。　　我深深感谢那些运用本书中描述的方法，帮助将可用性测试方法从实验室带到实际工作中，并将关注重心从测试转变到设计的所有主管、程序员、高级主管和可用性方面的实践者。正是因为他们的努力，整个可用性专业领域发生了变化。今天，我接触到的多数企业拥有一位或多位专业的交互设计人员。　　他们对软件产品、服务的质量和行为有着越来越大的影响力。得知这本书对他们的成功做出贡献，着实让我欣慰。　　记得本书刚刚出版后的1999年，我在一个程序员论坛上做过一次主题演讲。演讲的题目与本书的副题相同。我开门见山地提出我的见解，“精神病人在运营管理着精神病院，而你们就是精神病人。”当时，会场静得连一根针落地的声音也可以听得到。我可以感觉到由2500名工程师组成的听众群体有强烈的抵触情绪。在鸦雀无声的会场，我开始讲解本书的基本逻辑。演讲结束时，这群“逻辑人”已经被我彻底说服，并给予我热烈的掌声。令人惊讶的是，现在大多数程序员已经成为设计和设计者的忠实支持者。他们认识到他们需要在软件的人性化方面得到帮助。他们非常高兴终于得到有价值的指点。他们已经认识到，任何有助于改进质量和让他们的程序易于被人们接受的实践，都不会威胁到他们的职业地位。　　在过去，高层主管认为交互设计是编程问题，而将其职责授权给了程序员。程序员们也努力去解决交互设计方面的问题，虽然他们的技能、培训、想法、工作计划无法使他们成功。本着分析问题的精神，本书真实地描述了一些由于程序员的原因而导致项目失败的故事。有些程序员认为我在中伤，有些程序员则认为我谴责他们制造了劣质软件，因而他们攻击我的主张。劣质软件的确是由他们制造的，但是我认为应该受到谴责的不是他们。如果我给了他们那样的印象，那么我深感歉意。除了个别人，我知道多数程序员在为了让最终用户感觉良好而勤奋工作，在为改进他们的程序质量而不断努力。就像用户一样，程序员们只不过是混乱开发过程的另外一群受害者，时间太短，大量互相矛盾的命令，指导也严重不足。如果任何程序员有了我在挑剔他们的印象，那么我也同样深感歉意。　　软件制作过程的烦琐性——特别是编程工作的高昂成本和软件产品很低的交互性——根本不是技术问题。那是过时的商业实践强加于软件编程行业的结果。带着真情、好的愿望、上级管理层的祝福，程序员们更加努力地试图通过工程实践解决这个问题。但是无论如何，工程方法是不能解决这些问题的。程序员们已经感觉到他们的努力徒劳无为，工作中的挫折感也与日俱增。　　在我最近的一次旅行中，我注意到在程序员群体中，消沉的气氛在扩散。令人叹息的是，那些做得最好、最有经验的程序员们的遭遇却最差。他们对自己的努力感到疲倦和无奈，因为他们知道自己的技能被无情地浪费了。他们也许不能确切地知道事情是如何发生的，但是他们无法忽视事实。事实上，很多优秀的程序员已经停止了编程，因为他们发现编程工作令人沮丧。他们改行去做培训、促销、写作和咨询，因为这些工作让他们感觉到时间没有被浪费，工作效率也不低。这是一个悲剧，它本来是可以完全避免的。（虽然有些争议，不过开放源码运动成为这些沮丧的程序员们的天堂。他们根据自己的标准编程，仅由他们的伙伴评价其工作的好坏，而不需要市场或管理人员的建议或干预。）　　没有人给程序员们充裕的时间、明确的方向和恰当的设计，因而他们无法获得成功。企业主管们不愚笨也不是恶魔，然而这种情况的确是现实状况，因此，他们应对此负责。企业主管们仅仅是因为没有用恰当的工具武装自己，而无法解决信息时代所面临的复杂而独特的问题。现在可能又让人觉得我在树敌。是的，也只有在这样的时候，我的视野中出现的是商务人员的形象而不是程序员的形象。我想再次强调的是，解决问题的途径是有破才有立。我追寻的是解决问题之道，而不是替罪羊。　　管理大师Peter Drucker在他92年生涯中的多数时间里，观察并指导着企业主管们。他具有用独特的眼光看待问题的能力。在最近一期CIO（首席信息官）杂志的采访中，他对20世纪五六十年代出现的对数字计算机盲目乐观的企业主管们进行了评价。那些企业主管们曾想像计算机“会对如何运营企业产生巨大的影响”。但是Drucker解释道，“事实并非如此。没有几个主管提出‘我的工作需要何种信息？’这样的问题。”虽然数字计算机给予企业主管前所未有的大量数据，但是没有多少人问及这些数据是否能适当地引导企业走向成功。企业的运营模式发生了巨大变化，但是企业管理却没能跟着变化。他谴责生于商业社会的萌芽期，成熟于蒸汽机和钢铁时代，步履蹒跚地进入21世纪信息时代的会计系统。Drucker进一步肯定，“我们最需要的是与外部有关的信息，但是计算机并没有向我们提供这些信息。”　　在20世纪的最后几年里，随着.com泡沫的膨胀，因特网上大量的文章兜售了关于新经济的想法。学术界的人士宣称，在因特网上构建的商店里销售东西，与在那些用砖瓦水泥建筑的商店里销售东西是截然不同的，传统的“旧经济”已经名存实亡。当然，现在几乎

所有的新经济公司都已经死亡并消失，这让在背后支持这些公司的风险投资家感到震惊。而那些学术界人士也已经适时地修正了他们的观点，声称所谓的新经济是空中楼阁。最最新锐的主张告诉我们，必须继续维持原来那种古老的经济形式。　　实际上，我相信我们真的是处于新经济时代了。但是我不认为.com参与了新经济。相反，.com是旧经济（制造业经济）的苟延残喘。　　在计算机软件出现以前的工业时代，产品是通过加工有形物资制造出来的。采矿、冶炼、采购、运输、加热、成型、焊接、喷涂等过程消耗的费用远大于其他开支。财务人员将这些费用叫做“变动成本”，因为那些费用随着每一产品的生产数量而变化。而“固定成本”（你可能猜得到）则基本上不随产品生产数量的变化而变化，如企业行政管理和建厂初期成本。　　传统的商业管理源于工业时代的制造传统。不幸的是，我们必须正视现在是信息时代这样一个现实。在信息时代，产品不是由构成物质的原子制成的，而是由构成软件的比特（bit）的特定排列制成的。而且，“比特经济”不遵循“原子经济”的规则。

　　有些基本经济原理同时适用于旧经济和新经济。所有商业的目的都是创造可持续的利润。达到这一点的合法途径只有一种：以高于制造成本或进货成本的价格销售一些商品或服务。进一步细分的话，有两种增加利润的方法：或者降低成本，或者增加收入。在旧经济时代，降低成本是最有成效的。而在新经济时代，增加收入则更有成效。　　当今世界最为关键和昂贵的产品基本上或完全由软件组成。软件不需要原材料。软件没有制造成本，也没有运输成本。不需焊接，不需锤打，不需喷涂。工业时代经济与信息时代经济的本质区别在于：在信息时代几乎没有变动成本，而在工业时代，变动成本是决定因素。正是由于不存在变动成本，造就了现今的新经济。　　企业支付给程序员的工资是固定成本还是变动成本？程序员的一个工时肯定没有与一套软件产品的销售量产生直接的联系。企业可以反复销售相同的代码。投资于编程可以影响不同品种的上百万个产品，就像投资于工厂可以影响在工厂中生产的所有产品一样。　　编写软件的成本不是变动成本，但也不是固定成本。编写软件不同于建造工厂，它是企业实现销售收入的持续过程。建造工厂的工匠虽然收费昂贵，但是一旦建筑完成，他们会去另外一个地方做同样的工作。程序员比工匠和铁匠更加昂贵，但是他们不会去别处，因为他们的工作似乎永无休止。有人认为编程是研究与开发，两者的确存在着相似之处。可是，研究与开发是建立一项产品的理论上的可行性的思考和实验过程，其环境与实际生产环境有很大的不同。用一个恰当的比喻来说，传统的会计原则将研究与开发的经费从产生销售收入的日常营运费用中分离出来。而编写软件的经费，却无法准确合理地在传统的会计方法中列支。　　现在，有人可能将此看做是文字游戏，但是事实上它对软件项目如何获得融资，如何管理，如何被资深主管看待（这是最重要的）具有巨大影响。　　程序员创建软件，商务主管创造收入和利润中心。程序员用产品质量衡量自己的成功，而商务主管用投资的收益性衡量自己的成功。商务主管通过可以识别的，如固定成本、变动成本、企业的日常开支、研发费用等商业数学语言衡量收益性。但是，不幸的是，不存在适合软件或编程业务的会计模式。会计是基本的商业语言，上述商业术语对所有商业核算和商业沟通是极其重要的，以至于当代商务主管将其完全教条化。他们将编程业务简单地看做可以归于现存某一会计类别的另一项企业费用。在实践中，大多数商务主管简单地将编程当做制造工作－变动成本（只有在计算纳税额时，大多数企业才将编程归类于研发）。这是最坏的选择，因为这个选择无情地导致了他们商业决策的失误。　　工业时代的最大益处是产品可以大量生产，使得产品变得让大多数消费者买得起。消费者的收益是可以买到从前买不起的东西。手工产品只面向富裕阶层。企业围绕着与变动成本（制造成本和流通成本）直接关联的价格进行竞争。在信息时代，软件产品可以以非常低廉的价格获得。任何人可以通过下载方式或其他媒体方便地获得软件，而几乎不花费任何代价，也无需多少人为参与。　　请记住，企业可以通过增加收入或减少成本的方式增加利润。也就是说，企业可以增加固定成本投资以提高产品质量和销售价格，或者降低变动成本，亦即降低制造成本。在传统的制造经济中，降低成本是简单而有效的手段，因而被广泛采用。当今天的商务主管们将编程与制造等同起来时，他们以为降低编程成本同样简单而有效。很不幸，这些规则不再适用了。　　相对而言，软件受变动成本的左右较小，努力减少其成本并不能带来很大的商业收益。从财务角度看，程序员的薪资看似变动成本，但他们更像长期投资——固定成本。降低编程成本并不像降低制造成本那样有效。这样的效果更接近于给工人便宜的工具而不是压低工资。那些为降低薪资而将编程工作委托给国外去做的企业根本不得要领。　　进一步讲，在信息时代提高经济效益的惟一途径是通过提高质量，制造出更有吸引力的产品。这是不可能通过削减花费在设计和编程上的资金投入来实现的。事实上，必须在研究、思考、策划和设计阶段投入更多的实践和资金，才能让产品更好地满足客户的需求。　　当然，这种方式是一种不为21世纪的商务人员所熟悉的思考方式。他们需要增加在制造所有产品上的整体费用，而

不是降低单个产品上的个体费用。这是新经济的本质，也正是Peter Drucker所强调的。　　发明新药特药的现代药业公司，具有与新软件经济类似的特征。一粒药丸的实际生产成本微不足道，而一种新药特药的研制可能需要10年甚至更长的时间，投入上亿美元的资金。如果能提供一种奇效的新药，那么收益将接近无限，但是如果将不成熟的药品推向市场，则风险无比巨大。药业公司的主管们都清楚地了解削减研究开发成本是不可行的商业战略。　　创建软件更像发明药物而不像建造工厂。工厂是企业可见的固定资产，工人可以相对自由地相互替换。软件体现着一种无形的，极其复杂的思想模式，只有与编写它的程序员在一起的时候才有价值。企业不可以像对待工厂工人那样对待程序员。程序员们需要远超过于任何工厂工人的持续的关注和支持。　　在创建软件的过程中，软件架构——程序的人性化设计部分——用户研究、用例定义、交互设计、窗体布置、行为描述——常常是在压缩成本时最先被考虑的部分。设计过度的情况可能会发生，但是简化设计不会带来任何益处。花费在软件架构上的每一美元和每个小时都会在编程阶段节省十余倍的金钱和时间。更进一步地讲，如果在有竞争力的设计上投资，产品就会满足用户的期望，那样产品会创造更多的销售收入。期望的满足会建立品牌，有利于提高价格，提高客户忠诚度，使产品具有更长、更强的生命周期。这样做不仅会降低成本，而且非常有助于质量改进。具有讽刺意味的是，在信息时代，增加利润的最好方法是增加支出。

不幸的是，大多数主管具有无法克制的削减对编程时间和资金投资的愿望。他们运用的削减成本的方式已经过时。他们没有看到，削减在编程上的投资对产品的远期质量、期望性、利润具有负面作用。当然，只是简单地花更多的钱也不能保证产品质量有改观，而且如果没有智慧、分析、引导相伴，增加花费往往使事情变得更糟。我的启蒙老师Dan Joaquin常说，“有多大的投入，就有多大的收获”这句古老的谚语也许应该说成“没有投入就没有收获”。没有适当的计划就贸然前行的风险太大。这里的秘笈是花费适量的金钱，但是做到这一点要求管理人员具有很强的软件工程管理方面的专业知识，还要求有一种向管理人员提供做出正确决策所需的深度信息和处理工具。提供这样的工具是本书的目标。　　.com热是由那些以削减变动成本为主要商业模式的企业传播开的。虽然.com企业宣扬各种在线商务的好处，但它们的网站过于笨拙和无助，并不比驱车到购物中心让人舒适和方便。.com的创立者们（还有媒体）被兴奋冲昏了头脑，以为可以用非常低的变动成本重建流通企业。他们彻底而惊人壮观的失败无可争议地显示，信息时代的经济规则与工业时代的经济规则是不同的。　　在旧经济时代，更低的变动成本意味着更广泛的流通和更低的流通成本。这两项优点直接让消费者获益，它们是工业革命经济成功的基石。在新经济时代，商业上的成功依赖于向消费者提供更新、更好的附加价值高的东西。交易过程中每一部分的质量，从浏览页面到比较购物到多样性，必须让终端使用者明显感觉更好。如果在细心地阅读了11个画面的内容后，最终还得打电话的话，那就远不如传统的购物方式方便。反复三四次输入自己的名字、地址、信用卡信息，却发现不能从这个网站购买任何东西，还得去传统商店，只能使得整个在线销售变成不必要，令人失望。今天，简单地降低供应商的成本是不能保证成功的。　　当Pets.com公司在因特网上出售狗食物时，它并没有提供更好的东西，它没有提供比传统的宠物商店更好的用户购物体验；它没有提供更丰富的信息，没有显得更聪明，也没有显示更强的信心。Pets.com公司提供的仅仅是更便宜的配送、存储和销售——所有这些对Pets.com公司而言都是变动成本。这是忽视新经济基本原理的典型的工业时代的经营策略。Pets.com公司的商业模式不是新经济的开端，而是旧经济的残喘。　　我相信在因特网上可以销售任何东西并获利和成功。实现的秘笈就是在线商店必须比传统商店提供更令购物者满意的可衡量的服务，价格只是满意内容的很小一部分。实现的方法只有一种：必须整体地设计系统，才能尽可能地向终端用户提供最高的满意度。像对待制造工程那样对待软件设计和制作的任何方面都会导致失败。软件的设计和编程不应该成为传统的削减成本手段的目标。在创建软件的过程中过度花费时间和金钱的可能性不是没有，但是因过少花费而导致的风险要大得多。　　你可能熟知这样的风险，但是却很难让那些运营大企业的资深商业主管认同。这些主管依旧使用在蒸汽机时代就流行的会计系统，虽然其企业的运营、决策、沟通、财务等各个方面都已完全依赖于软件。他们使用的术语、概念根本不能反映现代商业独特的本质。在现代商业中，商业工具和产品是由看不见的比特排列而成的，而不是成吨的钢铁修筑的铁路。　　尽管企业聘用交互设计师并应用目标导向方法，但软件产品的质量并无太大改进。更为甚者，编程成本仍然高居不下，软件的创建过程仍难以驾驭。这是为什么？　　在资深商务主管认识到软件问题不是技术问题，而是值得注意的商务问题之前，改变是不可避免的。只有改变我们的创建过程和组织模式，问题才会得到解决。　　很多企业不仅遵循着过时的财务模式，而且还遵循着不恰当的组织模式。这样的模式是直接从学术界抄袭过来的。在学术界，创建软件的行为与软件的计划和工程联系紧密。这

是研究的本质。可悲的是，这样的典范被悄悄地完整无损地带到了并不属于它的商业世界。　　除了软件，所有现代制造规范源自前工业时代，软件这种独特的媒介出现在后工业化时代，只有编程直接来自学术界。在学术界，研究没有时间限制，学生劳力极其低廉，追求利润会遭到鄙视，失败的程序有可能被看做是非常成功的试验。像微软、IBM、Oracle和其他领先的软件企业驻扎在“校园”并非巧合。大学永远不需要挣钱，恪守交付时间，或创建满足期望，有用的产品。　　所有非软件企业以研究开始，结束于大量生产和产品或服务的流通。因为知道过早地生产不完美的产品会危害到收入和名声，他们会在研究和生产流通之间精心计划。他们知道在计划上投资时间、思考、金钱产生的结果是速度和顺畅，在最终产品的受欢迎成度和获利方面将取得巨大收获。　　在其他建造行业里，工程师做出建造规划，由工人来实施。工程师不修筑桥梁，那是建筑工人的工作。只有在软件行业，工程师的工作与创建产品的过程紧密相连；只有在软件行业，“建筑工人”参与如何生产产品的决策活动；只有在软件行业，这两项工作是同时而不是顺序地实施。但是，创建软件的企业似乎没有察觉到这个特殊性。在软件行业，工程和建筑如此密不可分地混杂在一起，以至于实践者和主管们也无法区分开来。各种计划或被省略，或被推迟，而此时已为时已晚。即使到了即将上市的代码进入了正常的开发周期，相当复杂的工程技术问题也经常性地被遗留下来。这时在经费上已经陷入窘境，通常需要额外的资金支持才能继续前行。　　系统架构的规划必须集成到工程计划的早期阶段。事实上，它应该对早期阶段的工程有所促进，但是因为早期工程经常被推迟到建造过程开始之后，并且会受到生产编码（production code）的破坏，架构设计缺乏进入建造工程的切入点。尽管事实上有些企业已经聘用了交互设计师，并且也重新培训他们的可用性测试人员来创建“人物角色”，但他们的工作对制作成本和最终产品质量的影响微乎其微。　　解决办法在企业总裁和首席执行官手中。如果这些企业主管们将解决办法下放到首席技术官或工程副总手中，就说明他们还不得要领。那些有价值的主管是技术人员，但是问题却不是技术问题。就像Drucker指出的那样，CEO们依赖的会计工具根本不代表其企业的真实状态。就像是说，因为速度表的显示是准确的，所以汽车行走的方向是正确的。在被数字技术主宰的商务世界里，这不再是真理。　　在创建软件的过程中，错误地应用会计和组织模式而引发的最大问题之一是主管们不能认识到有多少编程资金被浪费。一个精确的系统能够展示出每个美元都有一半是被花在了错误的地方，而且还需要额外的两三美元去解决由于不好的原始投资引起的问题。在任何其他行业中，这样的统计数据都会成为改革的动力，但是在软件业，我们却处在极度无知的状态。

　　在过去的13年中，我的Cooper公司为几百家企业提供了咨询服务。我们富有才华的设计师们为多数企业提供了可以极大地帮助他们的产品蓝图，但是只有屈指可数的几家企业从中获益。多数企业仅将交互设计和软件架构当做建议，而最后决策仍是由他们的程序员和工程师们做出的。没有一家企业的CEO知晓，在工程师的工作间里发生着什么事情，因此他们毫无理由地压缩时间表。程序员们总是在资源不足的环境中工作，主要是缺乏足够的时间编写好的程序，同时还缺乏决定编写什么的时间。他们被迫拒绝建议，搪塞他们的经理以保护自己。　　我相信有两种主管：一种主管本人就是工程师，另一种主管是惧怕工程师的人。前者将交互设计视为建议，是因为他们的眼睛被利益所蒙蔽。后者轻率决断，是因为他们不会说程序员的语言。我指的不是Java或C#语言。我的意思是商务人员和程序员缺乏共同的工具和目标。“普通人”将非技术问题托付给“逻辑人”，并不知道如果由他们自己在主管层面应用适当的财务和组织模式，解决方法会更好。　　企业面临着巨大的机会去突破障碍，去建立围绕着客户的满意度而不是软件组织企业的，围绕着角色而不是技术，围绕着利润而不是程序员的业务模式。我殷切等待着受启发的主管出现，他们能够抓住这个机会，并通过提供软件业大胆而成功的例子而永远地改变创建软件的方式。　　Alan　　于加州-门洛帕克　　2003年10月

　　inmates@cooper.com　　Cooper在软件设计方面的创新手法已经被认可为行业标准。在Cooper打开面向商业的大门后的十年时间里，这家旧金山公司向诸如Abbott Laboratories、Align Technologies、Discover Financial Services、Dolby、爱立信、富士通、富士通Softek、惠普、Informatica、IBM、罗技、Merck-Medco、微软、Overture、SAP、SHS Healthcare、Sony、Sun Microsystems、Toro公司、Varian，以及VISA公司提供了创新的、以使用者为焦点的解决方案。Cooper团队提供他们自己发明和优化多年的目标导向交互设计工具。这些工具包括在本书第一版首次披露的，叫做“角色”的用户建模和模拟用户的革命性技巧。　　1994年，比尔?盖茨向因Alan Cooper发明了Visual Basic语言背后的可视化编程的概念而向他授予了视窗先锋奖（Windows Pioneer Award），1998年，Alan Cooper在软件开发者论坛上获得了声望很高的软件梦幻奖（Software Visionary Award）。在1995年，Alan Cooper在他的的首部畅销书“About Face：The Essentials of User Interface Design”一书中，将分类学引入了 软件设计。2003

年，Alan Cooper和合著者Robert Reimann出版了此书的大幅修订版“ About Face 2.0：The Essentials of Interactive Design ”。 20年以来，Alan Cooper设计和开发的消费类软件产品包括，SuperProject，MicroPhone II for Windows和微软公司的Visual Basic语言的视觉编程用户界面。早在1976年，Alan Cooper创办了Structured Systems Group公司，一家被“ Fire In the Valley ”报道说制作出了“ 也许是为微型电脑的第一套商务软件 ”。 Alan Cooper还是一位为在电子产品开发过程中被遗忘的人——客户——的利益大声疾呼的人。 Alan Cooper是企业设计基金和美国设计中心的会员。他曾任软件设计协会硅谷分会会长，是总部理事会成员。Alan Cooper是软件设计和软件论坛的理事，还是SEF的Windows SIG ——世界上最大的Windows开发者群体——的创始人。他是关于用户操作界面和概念软件设计领域中的一位经常性的、有独到见解的、专注的行业演讲者和作者。 Alan Cooper的妻子，Susan Cooper，是Cooper公司的总裁和CEO。他们有两个十多岁的男孩，Scott和Marty。除了软件设计，Alan Cooper非常热爱通用航空、城市规划、架构、机动两轮车、烹饪、火车模型、飞盘高尔夫等。请向inmates@cooper.com发邮件给他，或访问Cooper公司的网站www.cooper.com。

媒体关注与评论

Cooper的设计方法已经赢得了很多大牌公司的青睐，如Sun公司、可口可乐公司、康柏公司和Dow Jones。 ——Fast Company杂志 Alan Cooper再一次展示的他的魅力！他的书值得所有认为自己正在服务客户的技术公司反思。我们需要更多这样所书，更多像Alan Cooper样的人。 ——Don Norman《情感化设计》一书的作者 本书主题鲜明地叙述着，要创建能赢得市场的系统，领导者们应该知道些什么。……你将会发现，在你读过的书当中，这是一本最具有思想、最实用的书。 ——Larry Keeley Doblin Group的总裁 Cooper的书是一纸宣言，它告诉人们，在我们需要花如此之多的时间与技术进行交互的时代里，如何提升生活质量。 ——Peter Hirshberg Elemental Software的CEO

《交互设计之路》

编辑推荐

本书是基于众多商务案例，讲述如何创建更好的、高客户忠诚度的软件产品和基于软件的高科技产品的书。本书通过一些生动的实例，让人信服地讲述了由作者倡导的“ 目标导向 ”交互设计方法在解决“ 难用 ”问题方面的有效性，证实了只有改变现有观念，才能有效地在开发过程中引入交互设计，将产品的设计引向成功。本书面向商务人员，也适合于所有参与软件产品和基于软件的高科技产品开发的专业人士，以及关心软件行业和高科技行业现状与发展的人士阅读。

《交互设计之路》

精彩短评

1、本书前面花费了很长的篇幅讲交互设计的重要性，软件不能交由程序员设计，否则会非常的难用。在90年代，这个思想应该是很前沿，突出编写出易用的符合大家需求的好软件。但是在现在，产品经理的大部分工作，都是调研找用户的真正需求、写PRD详细文档，最后交与程序员实现，程序员已经不处在主导软件设计的位置，只是负责实现而已。所有的PM都强调用户体验的重要性，大家都意识到用户体验的好处，可是真正做到的却很少。这本书后半部分，还是传授了很多有用的经验，比如设计时分角色、目标、场景，比如不是功能越多越好，而需要做用户最需要的，且把它们做精，比如，交互设计师需要对软件负全权的责任等等。虽然现在，程序员不处在主导的位置，但是书中讲到的程序员思维确实还存在大部分程序员脑中，他们实现软件时是基于电脑而不是用户。。。。。。
2、工作中如何与同事协同作战
3、就是设计在黑程序的段子
4、太繁琐，两句说完的用十句。总之不好。
5、这本书的方法论写得不是很详尽，但它用大量篇幅直指程序员和交互设计师思维上的代沟，有趣。确实在从事IT方面的产品开发，处理这代沟很重要，感同身受！
6、阅读于2015年3月
7、交互神器
8、程序员的生活过多。。。寻找共鸣的书，传达的是一种模糊的思想，启发性质的。针对实际解决方案少量一点。
9、作为一个小白，读完以后觉得还是有很多需要反复咀嚼的东西，也学到了很多东西，感谢笔者和译者，还会再读得。p.s.看到最热的书评我很气啊
10、还有两本
11、大师就是大师，对于程序员的理解真心恰到好处啊，我们遇到的就是同样的问题，但从没想过这就是高技术程序员的通用思维方式。设计产品：构建人物角色、建立明确的产品目标，为具体场景/人而设计……
12、没什么交互设计相关的，通篇都在吐槽
13、作者从这个时代的大背景出发，在分析了各类“智能”产品的不好用后，探讨了把设计放入编程之前的想法。软件的出现，带来认知摩擦的加大，使得程序员和交互设计师，用户和产品之间都有或多或少的隔离。
1、软件需精炼，少即是多，不添加过多的功能和元素，增加用户的使用负担。
2、实际需求和个人目标，不能混为一谈，好的产品在提供解决需求的方式后，还能满足用户的个人期望。期望值越高，用户粘性越大。
3、程序员的思考方式和大众迥异，要找到合适的方式去与其沟通。
4、目标导向型，任务不等于目标，该产品要解决什么问题，最后达到什么目标。
5、软件和人思考方式的不同，思考，如果是真实的人，面对这样的情况会有什么反应？不要把用户当傻子。
6、建立人物角色、场景、角色目标，具体化精确化
7、编写设计文档的重要性
14、虽然我不同意程序员就做不好交互设计，但是我不得不承认，程序员考虑的角度的确不太容易达到最终目标。
15、给管理层看的书，强调认识的重要性和一些基础概念的介绍，细节不多。
16、有点过时了
17、内容有点啰嗦，对交互设计我还是很感兴趣的，然而看完这本书，我感觉没有什么太大的收获。也可能是这本书比较老了的缘故了，里面的一些建议已经很广泛的使用了
18、1.程序员都是外星人2.角色建模3.认知摩擦
19、交互爸爸的书还是值得一看的,讲大道理比较多,想看干货的话直接跳到第8,9,10章。【全书微软被黑成狗,好熏疼
20、通读两遍 仅仅是了解了皮毛而已 虽说有年头了 但是味儿还挺正的
21、要好用

22、超赞的一本书，less is more，一口气读完，对交互设计也有了更深的认识，alan 的书都是经典
23、这本书好老了，稍微读过一点，还是很不错的
24、用词比较枯燥，要嚼很久，但是值得看
25、是翻译的问题吗？感觉讲了一整本书的业余废话
26、过年时候看了第一遍。对于任何产品开始以PM眼光去审视 一个闹钟 一个排插 一个煎饼果子。
27、2/3是废话或者早已经不符合当今时代，1/3是有用的、发人深省的。
28、有点老了 方法论1是让测试和设计先于编程2是在提高技术层面的假设下找到真正目标3基于可信度编造一个真正的人为他一个人设计4用设计文档 具体清楚
29、很有趣易懂，前半本啰啰嗦嗦一堆，后面很有用，把程序员黑的体无完肤哈哈哈哈
30、……看出版的年代和字里行间的语气，可能是在中国把程序员推下神坛，交互设计开始受到关注的引导性书籍吧……怒黑程序员的居多，干货不是特别多……
31、这本书比前两本读起来都累，因为全是字没有图，更重要的是作者花了太多篇幅介绍设计的重要性，读起来干货太少。我这种初学者了解得多一点没有坏处，要是急着读完看第四章就行了。
32、第一本让我读起来有自信 并不觉得自己愚蠢的书
33、看到96页，干货并不是很多
34、作者可是不遗余力的在黑我码农一族啊
35、Alan cooper
36、程序员的高级黑啊呵呵
37、在一本书里找到这么多共鸣是令人十分激动的事。
38、交互设计入门书，浅显易懂
39、1
40、看的很轻松也有收获的书！
41、毫无疑问此书已过时。不过不妨碍它是一本好书
42、对交互设计的重要性有了更深刻的体会
43、一大半都在讲程序员如何如何，不建议阅读，不值当。
44、算是交互设计领域的入门读物吧,其中阐释了交互设计中最基本的几种模型
45、讲了为何要做交互设计以及交互设计应该在什么时候做。这不是一本专门讲交互设计应该怎样做的书，但是其中的一些案例对初学者来说还是有很好的指导性！
46、又是入门读物，苦口婆心说交互。前面废话太多， 主要看的第4、5篇，分别是说用户角色、目标、场景等方法工具，以及设计在开发过程中的事情。设计的工具讲得比《交互设计精髓》要细一些，还有简单的案例。第5篇干货就比较少了，更像是为交互设计正名，“ 祈求 ”程序员和PM正视设计师。
47、还没看，不知道怎么样
48、且先找找感觉
49、很大的篇幅是在强调交互设计的重要性，真正讲如何做的内容不多。翻译不是很好，不太符合语言习惯。
50、飞速翻完。早7、8年这本书应该是不错的，现在再看都是already knew的老观点了。再算上作者对程序猿夹带太多私怨，只能给3星。

《交互设计之路》

精彩书评

1、【以下内容算是读书时候一些随想，想到哪写到哪，未必很有逻辑】在做用户访谈或者可用性测试的时候经常会遇到一种情况，就是用户常常为自己在某件产品的使用不便感到羞愧乃至歉意，认为自己太笨，或者没有很好地领会产品设计者的意图，有时因为这种羞耻感甚至避而不谈自身遇到的困难。这也就是所谓“认知摩擦”带来的最直接后果，我不知道应该把背后原因归结为“技术崇拜”还是“耻感文化”，总之，很多人在用不来一件产品时，第一反应是“我用不好”而非“它不好用”。有些产品（尤其是科技产品）因为自身使用逻辑无法让人充分理解，也就给使用带来了所谓“学习成本”。不过一些用户在投入了“学习成本”之后，并没有认为这是一个浪费。会产生游戏通关一般快感。这是由于，对于某种技能的掌握，让熟练操作的用户产生了某种智力优越感，从而会有：“你用不好不是东西不好，是你不会用”这类的辩护。这本书里，将这一类人群称为“辩护者”，且主要是指技术人员，其实不然。尽管作为设计者，进行调查的本意就是发觉这些任何产品使用中可能的问题，然而事实上，设计师有时候甚至也有意无意地为一些使用不舒服的设计进行极力辩护（有时甚至比技术人员更甚。好吧我可没说是itunes和android的粉丝哦）。本书认为这种思想本身就是反人类的，应该被警惕和清除。但是，实际生活中并不能这么简单粗暴一刀切。我们以图像处理软件为例，看看是否“辩护者”文化真的是完全错误：-Photoshop Lightroom：包括部分专业人士，对于使用方法和功能都需要学习和适应-Photoshop：使用方法和功能一目了然，如果掌握了基本专业背景知识，将能相对上手容易-美图秀秀：使用方法和功能一目了然，此外功能效果也直接呈现，即使不掌握专业背景知识，同样也能上手容易，不过操作熟练稍微需要些熟悉-Instagram：完全所见即所得，甚至无需任何练习就能使用自上而下这几个产品，学习成本递减，而同时功能的丰富性也递减。上述产品却没有优劣之分。这是一个关于将多少控制权和自由度交给用户的问题：一些用户有能力，也需要对操作有一个深度而全面的控制；而另一些用户对于使用结果要求简单，不想关心超过自己需要的功能。因此这里才回到本书真正地核心，认清你产品的对象，为具体的用户而设计。在一个产品的易用性和开放性之间，针对不同的对象，将有不同的平衡度为把握。“You press the button, we do the rest”，并非任何产品都需要。

2、不知道对真正的交互设计师有什么指导意义，不过，对我来说，作为一个程序员，书里的很多理念之前虽然知道，但在实际工作中，的确和书里的程序员一样，没有太多的把交互设计当一回事。这本书至少多少让我从认识上，把交互设计看得更加重要一些，以后和UI工程师打交道的时候，心态也许也会更平和些。；）

3、逻辑人Cooper说程序员是“逻辑人”，也许说的更为刻薄，之前总把其当作是跟程序员男朋友吵架后的消遣。因为专业交叉的问题，我们之间的谈话必定会有关于编程这样的专业问题，但是往往很痛苦，但凡我的疑问让他不能理解的时候，基本就是开始吵架的时候，因为我感到他的疑惑，疑惑我为什么这么“愚蠢”。知道来到俱乐部，三位程序员男生开始摆弄一台十年前就不用的服务器，企图让他成功的跑起来。这激发了他们的兴趣，当时无聊无事，我就着我渺小的计算机系统的知识艰难的理解着他们的讨论，时不时的还要google一下名词。让我费解的是他们居然在服务器没办法启动的情况下很兴奋的把一个U盘插了近去。他们竟然在基本无可能摆弄好这台股东服务器的情况下还讨论了那么久，而完全忽略了早就嚷着要吃饭的我。至此，我是真正的信服了Cooper的话了，他们是一群聪慧但是奇怪的物种。我居然有着软件和计算机的背景，但是我不是程序员物种，因为饿哦不会像他们那样的思考和兴趣。所以这注定了我们只能坐着不同的工作。先说服自己原本没打算写这本书的读后感的，因为读完后我发现其实没什么要写的，不是说这本书给我的印象不深，相反有些理念从一开始就灌输给我，直到读完正本的书我自己已经完全的吸收了，像自己的东西，浑然一体。就像是Cooper说的那样，好的设计就是让人感觉不到设计的痕迹，理念亦是如此。从很早就接触交互设计用户体验这样的词了，但是那时候的理解比较的肤浅，更像是我们只是接过视觉设计师的画，将它们按照“用户的想法”从新排列组合——这样最低级的工作。而现在我发现我还要做信息构架，甚至规划着需要的功能和不必要的功能，这些都是交互设计师的权利，原来。通用的工具or自己的工具什么是理念。理念就是你信服的东西。不是硬塞给你一只梨子，告诉你没原因他就是梨子。我曾经很详细的询问过一位研究生的学姐，为什么需要创建角色？是不是只是沟通的作用？因为我无法理解，难道创建了一个人物她就真的能说话能表达，能告诉你你做的不对？而我只能看到它作为沟通工具的作用，用来说服那些“逻辑人”程序员。但是Cooper告诉我，我只需要通过角色来精确的锁定脚色目标，以屏蔽不

必要的功能。也就是“我们的人物角色越具体，它们作为工具就越有效。这是因为角色变得具体后就失去了弹性（弹性用户理论）。”哪怕你指定的角色有偏差。书中有一句话就是：详尽而具体的定义角色比让角色非常正确更重要。这句看似矛盾的话却叫我深信不疑。这就是告诉我为什么梨子叫“梨子”，并告诉我这么叫的好处，这样让我信服，所以就叫做了理念。理念就像是一个主心骨，你不要别人的血肉，只要你需要就可以迅速丰满。此时就觉得从《火花集》上面摘抄的一些页面布局的“经典例子”这样的做法是多么的幼稚。自己的东西从书中我学到许多交互设计的方法，结合自身在学校学习的方法，我想我也许可以构画出一系列的流程，用自己的方式。于是，我开始在草稿纸上写下所有的方法，将他们排序标注，区别名词和概念。实际上，整体的设计要比我画的复杂许多，很多时候都是交叉结合而来，画这个图更像是吸收整理的过程，我一直认为别人的东西，那自己的话说出来实际上就是自己的了。如此我想要做一些实践，来验证和深化自己的理解。2013.03.09

4、如果说阿兰·库珀（Alan Cooper）是交互设计领域的教父，那么《交互设计之路》就是一本“圣经”。每一位产品设计信徒来说，我认为这样的形容一点都不夸张。关于这本“圣经”，我找了好久好久，2年来北京大多数的书店和网店中都无法买到它。半月前大雪纷飞的一天，一位在摩托罗拉工作的朋友在偶然之间居然把“圣经”带到了我面前。让我瞬间产生一种如获至宝的喜感。此前看到好多朋友的评论说这本书太老了，实际用途不大。但是通读之后还是学到了很多，其中关于“人物角色”的那一章对我的日常工作带来了直接的影响。相信如果三年前就能读到这本书的话，今天的我一定会更加专业。我推荐你阅读本书：如果你已经是一位极为出色的交互设计师，那么我确实不建议你读这本书来浪费时间。除非你不确定自己是否已经可以做到“极为出色”。如果你是一位有一定经验的交互设计师，那么可以通过本书以快速了解如何与技术化的代码和死板的工程师去打交道（前半部分），后半部分则告诉了我们交互设计师在工作岗位上应该做些什么。如果你是一位刚刚上路的新手，那么OK，这本十几年前的教材千万不要错过。它会帮你快速了解到应该如何去做才能成为一名出色的交互设计师而不是一只笨拙的“跳舞的熊”，如果你懂得思考则会快速发觉如何才能找到产品的核心价值。如果你是一位程序员，呵呵，那么这本书也许会让你觉得讨厌，但是如果静心读完它，会发觉自己有时候原来是那个样子？如果改掉那些不好的毛病，为自己增加一些人性化的思考，那么程序员的生活将不再枯燥，你的晋升之门也悄悄的打开了……昨天和一位跨国IT公司的交互设计师闲聊，发觉他现在的状态极为消极苦闷。因为leader每天都要求这些所谓的“交互设计师”去干一些前端要做的事情，比如实现产品的Flash、JS或搭建一些高保真原型。而真正用来研究产品的用户体验改进及信息架构设计的时间则少的可怜……因此他们的价值和想法始终无法得到体现。那么，如果您是一位IT公司的老总或设计部门的leader，如果你希望自己的软件或网页有更深远的用户影响力，则一定要学习本书，结合书中的理论知识看一看您手下的工程师与设计师们究竟是如何协作、如何接人待物？其中，交互设计师是否因为不协调的工作安排而疲惫并感到毫无创新之力。一个产品的完成取决于所有成员的共同努力；而产品是否能够突破传统开辟一条创新的用户体验之路，则必须要期待、认同、掌握交互设计师的价值。

5、观念更新不是一二，思想还是程序化的多，没有美学 面向商务，好在案例清晰 程序猿的优点完全是头势清晰 扫清辨识概念各种 厉害的角色 不能当书读。所谓回归人本结果还是机器做工，代码统治？一开始我interactive interaction不分的，后来穿插翻课内信息系统分析设计，讲到交互可视多媒体加人工智能的时候才意识到，用户最终交互对象是人，机器是媒介，最终满足的是人与人对话情境，设计软硬件创造context是本我记得辛向阳提过这个事儿。回归人性这个科技理念不像那么回事儿，科技人员已经丧心病狂了提问，从中读出男权意味是怎么回事

6、翻遍了互联网的所有网上书店，就是找不到哪里有卖这本书，基本上都是“缺货”……郁闷，过去还真没有觉得这么需要读一下这本书，现在因为买不到则变得更加饥渴了。。。。。

7、交互设计确实很重要，但Cooper也是太有个性了一点点，呵呵，很多言语都比较刺激读到期望性一节的时候，比较有感触，本人喜欢google，apple的产品，其实不时有一种期望，认为他们可以有优秀的产品推出，或者极度相信，他们有能力将目前产品做的很优秀The Inmates are Running the Asylumby Alan CooperViPIS Journal Club 7 December 2005Reviewed by David Eibling MDThis outline is derived as actual quotes from the Table of Contents and text of Alan Cooper. Please do not reproduce without attributionComputer Obliteracy* 1. Riddles for the Computer Age* 2. Cognitive FrictionIt Costs You Big Time* 3. Wasting Money* 4. The Dancing Bearo a. What's wrong with Software?+ i. Software forgets+ ii. Software is Lazy+ iii. Software is Parsimonious with Information+ iv. Software is inflexible+ v. Software Blames users+ vi. Software

Won't take responsibility* 5. Customer DisloyaltyEating Soup With a Fork* 6. The Inmates are Running the Asylumo a. Driving from the backseato b. Programmers cannot consistently arrive at a successful design* 7. Homo Logicuso a. The jetway testo b. Homo Logicus - wants control, accepts complexityo c. Homo Sapiens - wants simplicity, accepts less controlo d. The 7 habits of highly engineered people (parody of Steven Covey by Po Bronson)* 8. An Obsolete CultureInteraction Design is Good Business* 9. Designing for pleasureo a. Personaso b. The elastic usero c. Realistic estimate of skill levels (most users are intermediate users - who is YOUR user?* 10. Designing for Powero a. Goals versus taskso b. Whose goals?o c. What makes software polite?o d. Basic science - user investigation* 11. Designing for Peopleo a. Scenarioso b. Inflecting the interface+ i. Programmers design for experts+ ii. Marketers design for beginners+ iii. And who are most of your users?o c. Pretend it's magico d. Reality bats lastGetting Back into the Driver's Seat* 12. Desperately seeking Usabilityo a. Redo timing+ i. Old paradigm - "program, bug test, tweak"+ ii. New paradigm - "assess user, design, program, user test, tweak* 13. A managed Processo a. "Consumer-driven death spiral"o b. "Listening to versus following"o c. Use your own filtero d. Scott Jensen says "Fish Can't See Water" (my quote)o e. Document designo f. Design affects codeo g. Design Documents help programmerso h. Design Documents help marketing* 14. Power and Pleasureo a. Company-wide awareness of designo b. Change the processo c. Survivor versus apologist

8、　　UI设计未来发展趋势还是很不错的，东华英丞每周都有关于UI设计的公开课活动，更多ui设计的信息可以关注：http://site.douban.com/214504/　　“英丞设计师之旅”创意思维拓展，成就非凡梦想！突破瓶颈，让您在职业生涯中增添色彩，现报读平面视觉课程，与名师名企接轨，开拓职场人脉。在课程中，迅速拔高创意思维，提升知识面，拓宽设计眼界。同时学会项目汇报及项目沟通的技巧。参与到与老师的一对一项目策划，完成创意方案，给自己迈向成功的第一步。加入英丞微信好友，报读即得课程优惠。

9、很好的一本书从心理角度,需求角度,各种角度来告诉人们交互体验的重要性是我见过的最好的一本关于交互设计的书了

10、这本书可能比较老，没有神马新鲜玩意儿。但对入门者来说还是值得一读的。1.书中花了很大篇幅，举了很多例子说明了不注重交互设计的后果，列出了很多程序员读起来很不爽的观点。其实就是强调“以用户为中心”而设计，而不能让开发人员(高端用户)参与设计。有一点我倒觉得很中肯：“不是技术让人们丧失人性。是技术人员，或者说是技术人员使用的过程，创建了不人性的产品。”2.设计一个产品时，要将用户的形象具体化，也就是书中大肆宣传的角色创建法。3.好的交互设计是目标导向的。目标是用户的最终想要的，而任务是为了完成目标而需要做的中间过程。只有这样才真正抓住了用户的需求，才能让产品拥有用户忠诚度。另外要注意的就是用户的个人目标，书中说的就是“不觉得自己蠢，不出现差错，完成适量的工作，有趣”，这都是设计细节要注意的。感觉这本书有的地方写的比较啰嗦，可以大致浏览一下全书，选择感兴趣的地方细读。书中的案例很不错。

11、对于一些人的评论不敢苟同，作者在导读里已经清楚的告诉阅读方向不能截图部分 转移如下：“对于只想了解软件产品或基于软件的高科技产品存在的问题的读者，可以只读第一篇和第二篇的第四章。在工作中直接参与软件开发过程的读者可以着重阅读第三篇和第五篇。对具体交互设计方法有兴趣的读者们可以着重阅读第四篇从事项目管理的读者，可以着重阅读第五篇“.....把自己不需要当成别人不需要的人，你不会是一个好的交互设计师

12、这本书除了最后的1/4讲的角色、目标、场景有点实际意义，其他的通篇就在强调一件事：设计，程序猿不行，他们不是正常人。这本书是上世纪90年代的产物，现在事情早不一样了，编程已经简单多了，早不是只有nerd才能胜任的工作，我猜如今的大多数程序猿都是正常人。而且作者创办了一家交互设计咨询公司，书的字里行间流露着这样的感觉：你们真的需要设计咨询公司的服务，没我们不行。这点更是让我怀疑这本书的客观性。产品经理们自然会力挺这本书，这本书的原教旨主义浓烈，论证则一笔带过，正是他们的风格。作为一枚程序猿，我必须承认，交互设计方面我是不专业，我热切希望避开这项工作，交给专业的人。不幸的是，目前我还没遇到专业的。大家都不专业，起码我讲道理，会做分析推理，不瞎拍脑子或者桌子，所以，还是我自己来吧。

13、在这本书中，作者基本上没有对具体的开发说什么，主要是想让整个软件开发行业建立起一种制度，让交互设计师也可以在里面有自己的位置。其实很多东西都是不言自明的，很多东西每个人都在说，软件是给用户用的，当然要以用户为中心，为了自己公司的持续发展，又不能完全按照用户的需

求来做，等等。发展，其实就是在找平衡。书里BS了微软一下下，但哪个公司又不是这样呢？对交互设计师这个职业很期待，国内似乎还很少？不知道。
14、书的名字很长，不过基本反应了本书的目的。看完这本书后，我才发现软件是多么难用，做为一名使用计算机多年的老手，尚且有诸多不便（哦，你还找的到上个星期下载在你硬盘某个角落的文件了么，再上个星期呢）更不提其他非专业人士了。我们习惯花了很多精力把界面做的异常华丽，但是这对产品易用性有多大好处——“尸体彩绘”，这个名字很贴切。，方向错了，做的越多越浪费。当然在现实中完全做到这些很难，但是在每次写程序时脑子里始终有这根铉的话，你的程序的易用性也会好很多了。
15、本书花了200+的页面，无非是想说明一件事情：交互式设计需要基于具体的角色和目标来做，而非传统方式的仅仅基于非常泛的用户改变来做。如果是在90年代，这个想法还是非常前沿的想法，但是在00年以后，可以说在UML，RUP，敏捷方法中都泛滥了角色建模之后，是否真的还有必要仅仅只为了提到这些而写一本书，确实值得商榷。本书一共分为5章，前两章介绍了当前交互式设计中存在的一些问题，第3章对这些问题进行了一些分析，第4章是基于角色和目标的交互式设计，而第5章则属于管理开发的内容（大多数还是陈词滥调）。作为一个对交互式设计不太熟悉，对软件开发的本质不太清楚，不太了解软件工程的人来说，这本书可以作为交互式设计的启蒙教材，引领你进入交互式设计之门。如果愿意遵从本书中介绍的“角色—目标”的办法来做设计的话，在当前行业内也还算比较前沿的理念。“角色—目标”建模的好处在于，它细化了对特定用户的描述，将原来最多处于统计级别的用户描述（不明白上什么意思的话就算了）甚至很多时候连这都做不到，转化为对个性化用户实例化，详细化，具体化的描述，在设计同一产品时，让设计人员面对的不是一个用户群描述，而是一组聚类好的个性化用户的描述，这有助于更好的实现交互式设计。下面来说一下本书的几个硬伤：1.本书前两章花了过多的篇幅。更为关键的是，本书在第4章的解决办法，并不能解决前两章所提出的所有问题。好吧，其实这也没什么，我受不了的，其实是作者没有写明白，第4章他提供的办法到底能解决前两章中的哪些问题，哪些是解决不了的，而且确实，“角色—目标”建模不仅仅不是万能的，就算仅仅只是这个理论本身，也存在一些问题，没有办法解决之前提出的所有问题，虽然我是能看明白，但是不能保证其他的读者也能看得明白。（作者在这里交互式设计没做好）就我的观点来看，做这种“提出问题—分析问题—解决问题”路线的书，要么解决问题的办法能解决书中提出的所有问题；要么你解决不了也可以，但是把哪些能解决，哪些不能解决，为什么不能解决写明白，也行。象作者这样写，实在有点不负责任。2.第三章的分析，看得出作者对这个行业的理解还不深刻，他主观的将传统行业从业人员和程序员剥离，认为这是两个完全不同的行业，包含这完全不同的文化。甚至，他还举出了很多表现，要证明程序员就是和别人不一样。难道程序员是外星人？难道软件开发是外星人的行业？作者显然对行业的发展没有深刻的认识。事实上决定行业思维方式，运做方式不同的，即有从事这个行业的本身的特点，实际上也还包含了这个行业一直以来的竞争激烈程度，竞争方式，以及使用的何种工具等。行业的壁垒是在行业发展中逐步形成的，这并非是完全剥离的，不可理解的，甚至也不是不可转化的，需要的仅仅只是彻底的了解形成原因，跳出自己原来的小圈子就行了。作者在这里成了一个思考不健全的理想主义者，他即不了解行业大局，也没明白是现实和理想的之间的妥协才造就了产品。就我个人认为，本书的作者需要重新回学校去补一补系统化思维的课程。3.本书依然在回避一个问题，或者说没有正视一个问题，又或者说从来没有考虑到一个问题，那就是本书依然将文档作为设计人员和程序员之间交互的接口，而设计人员和程序员之间的冗余太小了。4.目标导向是对的，但是和其他所有的以目标为导向的设计一样，本书对目标的认识存在一些偏差。人为的将目标与行为割裂，事实上这两个看起来，更象是一组小目标实现了一个大目标的关系。我打赌，这样的设计方式必然会造成对目标识别的混乱，要么，这种设计方式在未来系统扩展的时候会出问题。因为就算是被他们定义为目标的东西，实际上还是有层次的，有结构的，有先后关系的，并且还支持着更大的目标。有人直接跳过目标的中间层，将最终目标定义为个人的快乐，这也同样不是一个好办法，因为你还是不知道中间是怎么来的。5.最后一章很突然，让我觉得突然的原因不是说不应该写管理的部分，是应该写，但是似乎本书作者在此处没有任何新的建树，哪怕一点点也没有，这让我感到很奇怪。本书的作者打着“回归人性”的旗号，却实际上仅仅只是把“角色—目标”建模的方法穿了个传统软件工程方法的马甲以后拿出来。如果他真正理解了“角色—目标”建模方法的含义的话，他应该有限制的和传统软件开发方法决裂，因为那个有些非人性。可能也是因为作者对这一块理解上不够深入吧。

16、Cooper大叔是一代吹牛大师，说这个并不是诋毁大叔，他的确是个大师，只不过喜欢部分夸大。大叔的潜意识里仿佛一直在呐喊：“产品是设计师的战争，其它闲杂人等统统让开！”事情果真是这样的吗？Cooper的书中最有技术含量的就是角色-目标的分析技术，在我有限的知识背景下，我是认同他的这个技术的，这个技术也是Cooper交互设计的核心所在，那么只要分析清楚这个技术是谁的武器，就能分析清楚产品究竟是谁的战争了。我对角色的了解开始于软件开发中常用的角色-权限功能，表示具有一类共性的人。Cooper的用法则是指重要的典型用户，我猜想，这里更多的是指那些实际使用产品的人。把眼光放长远点，看看我们是如何知道谁将使用产品的。真正的商业化公司，在开发一个产品时，必须经过一个详细的市场调研过程，有上进心的公司都会有自己的理念，往往会用市场调研来印证自己的核心价值是否被市场接纳，或者说被哪些人所接纳，而一般公司则会研究市场需要什么，然后依据市场需要去做一些事情。不管其目的如何，他们最终都会被抽象为一个个角色，这些角色拥有各自不同的背景、诉求和期望。怎么样，很熟悉吧，这不就是我们熟知的市场细分嘛，那么建立这么多的市场细分或者说是角色后，公司自然要决策出应该为那些角色提供服务，而忽略另外的角色，这也正是Cooper屡次提到的“要为一个人进行设计”（实际上他经常为2-3个角色进行设计）。但不管怎么说，这个活是属于市场部门的专业，单凭一个设计师，搞搞企业级应用尚可，搞产品则会直接面对两个无法逾越的障碍：一是对大量用户市场数据的掌握；二是缺乏对这些数据分析的技能。没这功夫，从哪里谈对角色的精准划分和定义呢？再看目标的问题，我大体是知道企业目标、用户目标和个人目标之间是不同的，Cooper认为达到你好、我好、大家好的局面是可行的，因此在设计中应该兼顾有三。但在实际应用中，这是行不通的。借用Cooper的例子来说，企业目标是利用产品提高管理水平，用户目标是少干活，多拿钱，最好还有油水捞。那么他们的目标只有冲突没有交集，满足企业目标意味着严格管理，尽可能多的搜集数据，控制过程，而这样做无疑是断了用户的财路，增加了用户的录入负担。实际上许多用户抱怨产品难用，并不是产品本身的功能不好使，而是他们需要借这个由头，将产品推翻，恢复马吃夜草的老局面。Cooper含糊其辞的说系统需要放宽对潜规则的管制，比如可以允许用户非常规的操作，如果这个真可行，那么所有产品的最终都会沦落为数据报表生成器。因此，知道用户的诉求，明确角色的目标，并不是革命的胜利，而是万里长征的第一步，那些目标是我们支持的，那些目标是予以忽略的，才是产品决策的另一个关键。解决这个问题，不得不又提出公司的核心价值是什么了。一个以提高企业管理水平为己任的公司，是不会理会潜规则的。而一个为万民造福的互联网公司，则不能不认真对待这些潜规则的诉求，因为少部分人的潜规则，可能是大多数人的显规则。现在可以看出，交互设计师手中拥有的角色-目标武器，居然不能亲手玩转，既不能持枪的瞄准，又无扣动扳机的力量，他们怎可能独自带着温彻斯特步枪横扫整个西部大平原呢？最后提一下，Cooper是当之无愧的大师，看这本书时，我曾多次由于情绪处于过度亢奋，而无法继续阅读。因为他的书将设计的事情正大光明的摆上了台面，而且对设计寄予技术上的高度期望，这对于目前国内设计即是开发，或者设计就是文档，或者设计是界面的现状，具有一种讽刺式进步的力量，是一本不可多得的好书。我打了满分

17、比起《情感化设计》这种纯粹的理念性书籍，这本书的实用性还是很强的。书中对于一个产品的实际开发过程有很精辟的阐述与见解，对于交互设计的各种阻力也分析的很到位。看了之后感觉很多在实际工作中遇到的就是这种情况。

18、#交互设计#爱因斯坦说：“不能用与引发问题相同的思维方式去解决问题”。《交互设计之路》一书终于在第9章开始讲交互设计的方法：“目标导向”=看待问题的新方法+有效的指导原则+有用的工具。（该书前8章一直在讲缺乏交互设计思维的程序员如何如何的失败。）

19、1、人物角色编造虚假用户，然后为它们进行设计。关键字：只为一个人设计，真正的用户并不是弹性的，让人物角色具体化，假象的人物而非真实的人物，精确而不是正确，角色终结了功能争议，使用户角色，而不是购买者角色，角色表，首要人物角色是设计为之服务的中心人物，但不要超过三个。2、目标目标是终结条件关键字：任务和目标的区别，目标导向设计可以得出更好的解决方案满足个人目标的重要性3、场景场景就是对角色如何使用基于软件的产品达到自己目标的简明描述。关键字：日常场景，必要场景，边缘场景，屈折界面，永久的中间用户，假装它有魔力

20、这是一本非常值得细细品味的书个人读书比较囫囵，但这本书个人实在只舍得精读，呵呵作者用非常幽默的文风陈述着交互设计的重要性而它的比喻尤其使得人觉得获益良多“斯德哥尔摩症候群”，跳舞的熊让人记忆深刻而发人深省

21、“尸体彩绘”是这本书中的一个贬义词，这个词很恶心，但是用它来形容目前很多“华丽”的网

站和软件界面其实太恰当不过了。要避免去做”尸体彩绘”，强烈推荐阅读这本书。
22、喻体用的太多，读起来很困难。不是我个人理解的问题，就是文化差异问题。比不上他的另一本著作《软件观念革命》。是本“软读物”。
23、Alan Cooper的《交互设计之路》是本好书。书的价值重心在第4篇（P115-P188）。这部分讲了交互设计的三个工具——人物角色、目标、场景。其中，有颠覆性的观点是关于人物角色方面的认识。作者指出现在常用的“用户”其实是综合各种细分用户的“弹性用户”，而为“弹性用户”设计最后很可能导致一个大家都不喜欢的产品。基于此观点，作者建议为每一个项目创建一个角色表，通过角色描述让人物角色具体化。（比如这么一段描述：Chuck Burgemeister，54岁，男性，商务旅行人士，10万英里俱乐部的会员，几乎每周都飞往各地。）作者甚至认为让角色精确比让角色正确更为重要。角色表中，需要确定首要人物角色，首要人物并不来自于首要细分市场，而是来自于用户遇到的首要问题。这也是为什么Cooper在P@ssport案例中没有选代表商旅人士的Chuck，而是选了代表不会操作电脑的老人Clevis作为首要人物的原因。看完之后，有得也有惑，惑的就是，如何将交互设计中的人物角色与Marketing中的细分客户结合？如何使得设计与营销不会脱节？Cooper比较赞同“三品质概念模型”：由技术解决的可能性；由商务解决的可行性；由设计解决的期望性。可能性指什么能做什么不能做；可行性指什么能卖什么不能卖；期望性指什么是值得期待的。做到第一点，能做出真实的产品；加上第二点，能做出可以卖掉的产品；加上第三点，能做出可以给用户惊喜的产品，从而使得企业长期成功。这是从一个设计人员角度看到的结构，如果从一个营销人员的角度看，后两者是可以合并的。设计还是应该由营销来导向。可能可以举出一些很牛的设计导向的成功案例，但，这并不能说明该由设计来引导营销，而可能是案例中的设计做得太好，替营销做了一些工作。
24、这本书应该给程序员们好好读读.书中并没有讲很多详细的这里该怎么设计,那里该怎么设计之类的,更多的是一种让你思想上的转变,可以让你认识交互设计的重要性.如果对交互设计有兴趣倒是可以读读,可以更清楚的明白交互设计的重要性.
25、用了三天的闲暇时间读完了这本书，有种豁然开朗的感觉。之前也略微看过作者写的《交互设计精髓》，觉得内容很多，不好理解。个人认为这本内容浅显易懂，属于交互设计方面的启蒙书，比较适合想要步入交互设计行业的初学者。书的重点放在论述交互设计在基于软件的产品开发的重要性方面，作者运用了举例的方式，还是比较有说服力的。此外，本书的第四章论述了交互设计方法，简单介绍了用户角色、场景，以目标位导向的设计，容易理解。总的来说，这是一本值得一读的好书哦~
26、一本在我书架上沉睡了三四年的书，每次想看，都翻到前几章节。随着从业经历的丰富，发现越来越能体会到这位交互设计之父的思想。目标导向设计，设计要先于程序开发，我越来越感受到这种模式的重要性，不光是为了参与流程的所有人，也为产品能最终取得成功。好的产品能让用户有非常高的忠诚度，产品在设计时，不应只考虑满足人们的需求，而更重要的是满足人们的期望，想想有时我们在日常生活中，充满了这样的例子，期望总是在人们需求得到满足后产生，在我使用apple的MP3和手机时，我会惊讶的发现，原来mp3和手机可以是这样，他颠覆了我的想法和观念，他的各种软件应用，人性化的操作不仅满足了我的需求，同时让我知道，原来我还期望这样和那样的服务，从此，我就成了apple的死忠，就算有产品能够短时间内抄袭apple的交互设计，但是apple用户的强大忠诚度，让他避免市场份额出现巨大损失，立刻对竞争产品作出反应，推出更加良好的产品。这是很多产品所欠缺的，office是我必备的软件，我几乎每天都在用，但它眼花缭乱的导航栏让我在想插入个表格时几乎要崩溃，如果真的有一款比他交互设计好，或者性价比好的产品，我会毫不犹豫的转身而去，我对office没有任何忠诚度可言。为用户着想，从使用者的角度出发，那些是应该有的功能，哪些是多余，可以省略掉的，书中作者举了很多个实例，解读详细，其实都是我在日常能够接触到的。建立角色，让流程里的每个人都能客观的审视产品，而不是主观的认为角色是傻瓜或者专家，不再让使用者觉得自己非常愚蠢。很多时候，我们的产品不是功能不够，而是它过于强大，过于繁杂，就像我的电视遥控器，上面有几十个小小的按钮，而我现在都不知道那个能一下子切换到画中画功能。很多时候，少即是多。正像作者说的，其实软件开发与拍电影很像，拍电影时，前期的设计工作会相当长，包括剧本，角色的确定，往往高成本的拍摄时间很短，再到后期的剪辑，在拍摄之前，要做足准备工作，知道某个时间节点要做什么，做到什么地步。而我们的软件开发，经常是老板拍脑门说“来，做吧，不成上线再改！”结果大家对这长长的功能列表开始讨价还价，往往最后上线的东西已经是个四不像，违背了当时产品设计时的初衷，又谈何能有抓中市场，抓住用户呢。现在很多软件和互联网公司都注意到了交互设计的重要性，培养自己的交互团队，但给我的感觉是，交互设计师仍然在夹缝中求生

存，在流程中位置尴尬，往往产品也不需要他们负责，他们只是充当辅助性的角色，或者说这种挫败感使他们根本拿不出什么成熟的交互设计方案，或者所有人都在怀疑这个方案。记得以前一个产品经理对我说“ 这做交互的拿不出让人眼前一亮的东西，凭什么叫我们重视他们的想法，天天哭着喊着希望得到重视，其实是自己没本事 ”，所以，我想交互设计师之路，还需要很长的时间去历练自己。
27、最近想看跟交互设计相关的一些书，于是借了这本业界“ 圣经 ”。虽然是十几年前的书了，但对于我这种菜鸟还是有不少启发。作者在书中举了不少例子说明糟糕的交互对人的伤害：让人觉得自己愚蠢，让人在使用产品过程中体验不到快乐。而这一切源于不懂交互设计的程序员控制了产品的开发过程。作者认为程序员对用户缺乏认识，以自己的标准来设计产品（功能为王），忽视用户体验，进而导致产品的失败。解决方法就是在编程开始前由交互设计师进行交互设计。由于作者就是开设计公司的，这种说法难免有自夸自卖的嫌疑，但联想到生活中遇到的那些糟糕的产品设计，你会觉得这么说其实不无道理。举个例子，我通常用印象笔记做读书笔记。有时在某一句话上用黑体、改字体颜色以示强调，但接下来你继续编辑的时候这些格式会延续下去。难道你在编辑文档时强调了一句话，就说明接下来的文字你也想强调？这是一个很糟糕的设计。作者提出的观点在今天看来很普通了，就是用户体验至上。这通常意味着个性化设计，不仅仅是针对一个小用户群，更是针对每一个特定的人。比如让软件记住你的操作习惯，预知你的喜好，进而提供个性化服务。如果产品能提供像人一样体贴的服务，那么成功的几率会很大。这本书另一个有趣的地方是作者提供的三种交互设计工具，分别是：人物角色法、目标、场景设计。人物角色法是根据产品目的及调查结果创建出虚拟角色表（是真实用户的假想原型）。其目的是通过精确地描述用户和用户想完成的事来揭示用户目标，规范、限制设计的范围和本质。目标分为个人目标、实际目标和企业目标。个人目标是增强用户体验，比如容易理解、操作简单等。实际目标就是产品想解决的问题。企业目标是提供更好的服务、提高市场占有率、增加利润等。在这些目标中，个人目标是最重要的。产品就是给人使用，解决人的某种需求。使用者用得爽了，产品不就成功一大半了吗。场景设计分为：日常场景、必要场景和边缘场景。日常场景是你最常使用的那些功能服务，所以这块的交互设计是重中之重。新手如何快速理解、掌握，熟悉之后如何根据喜好进行个性化设置。必要场景是那些不常用，但必须具备的场景。由于不常用，使用者会愿意迎合程序的做事方式，不考虑定制操作，但这也是交互设计的重点。边缘场景其实作者没说清楚，只说这不是交互设计的重点。个人认为从软件上说，边缘场景应该就是程序的异常处理。目标导向和场景设计的理念在今天来说不算新鲜，我感兴趣的是人物角色法。如何创建一份可靠的角色表？多大程度上它能代表你的目标用户？如何评估？作者对此没有说得很详细，但在案例研究中可以看出一些东西来。在索尼Trans Com公司P@ssport系统的案例研究中，是这样描述角色创建过程的：走访参与项目的大多数人，包括项目经理、开发经理、工程师、营销经理、内容经理，明晰产品的目的、目标市场（机舱娱乐系统）的发展历史和技术应用。接着走访目标用户（机组人员。实际上乘客也算）。“ 对所有角色都进行唯一细致而精确的描述。可以根据软件主题领域对她进行审查，以便确认她的确是原型用户。 ”【120】“ 不要将精确的用户分类与真实人物混淆起来。真实人物可以作为原始数据进行参考，但通常没有多大用处，反而对设计过程有害。他们带有干扰设计过程的不良嗜好和特殊行为。 ”【121】“ 发现一个人的目标很特别时，我们会分离出一个角色。没有必要让所有的角色目标都不同，但是每个角色的目的必须有所不同。 ”【128】“ 每份角色表至少有一个首要人物角色。首要人物角色是设计为之服务的中心人物，这个角色的目标必须得到满足，但是不能通过为其他角色设计的界面来满足。首要角色应该有专门为他设计的操作界面。 ”【128】“ 每次机组人员告诉我们一个新的故事，我们就创建一个新的角色...如果发现两个角色目标相同，就将其合二为一 ”【133】如此看来，作者是在产品目标为导向的前提下，通过访谈获取原型用户形象，再提取共性，加上充分、精确的描述创建出典型虚拟用户形象。在这个案例中，有一个很奇怪的地方就是Clevis这种手脚不灵活，不懂操作电脑的老年人居然会进入角色表，还是首要人物！通常来说，飞机乘客应该是商务人员或旅行者比较多，老年人比较少见，主要目标对象应该是商务人员或旅行者才对。不过反过来想其实是对的。首先满足最挑剔的人的需求，是因为其他人有更多的耐心付出时间和精力学习新事物，对操作不便也更有忍受力，如果你能提供他们想要的服务的话。
28、目录第1篇 电脑的逆向文化1 信息时代的谜语2将电脑置于机舱，你会得到什么2将电脑和照相机结合在一起，你会得到什么4将电脑和闹钟结合在一起，你会得到什么5将电脑和汽车结合在一起，你会得到什么7将电脑和银行结合在一起，你会得到什么8电脑更容易导致麻烦9商业软件也同样遭殃11将电脑和军舰结合在一起，你会得到什么12技术的愤怒13整个行业都在拒绝承认14本书的起源142 认知摩

29、作者的观点有些偏激。首先，计算机世界是这样一个人群维护的，如果期望改观现状，必需要从这样的人群中寻找内部的力量。任何群体都不会轻易放弃自己的发言权，在职场上这样的发言权更意味着薪酬和地位，程序员群体也一样。其次，所谓设计师是一个平衡者，而不是一个艺术家。这点在各个领域都是这样，建筑设计师、工业设计设计师都是这样。作者的观点有些偏激，所有的产品（大到三峡大坝，小到火柴）都是易用性、成本、技术可能性、……多种因素综合得到的。个人觉得《设计心理学》（http://www.douban.com/subject/1288844/）的作者对于设计的境界更高一些～

30、这本书讲的道理不多，但是没有什么废话；作者在短小的篇幅里面就说明了交互设计的重要性，并给出了对现有的产品开发流程的简单但是重要的改进建议，以前讨论软件的开发和设计都是从开发者出发的；俨然把产品设计也包括在了里面，生产出来不同层次的废品也在所难免。作者至少指出来这个问题，并给出了解决这个问题的新的入手点，注意，不仅仅是系统架构师，而是交互式产品设计

师。
31、www.cooper.com面向商务人员的——交互设计的重要性，必要性，再到，面向技术人员的——如何做交互设计。传统工业时代：降低成本 现代软件时代：增加销售收入 因为编程和制造完全不同。不能通过降低编程的成本来提高收益！没有交互以前：技术由简单到复杂的缓慢进行当中，他们已经被逐步的计算机化，我们在日常生活中尽力着于此相同的、缓慢的电脑蚕食。出于编程这个岗位的人员，有时会因为沉迷于编程自身的难易程度，，二从来不重视对用户的关怀。程序员在编码的过程充斥了太多的设计细节，选择代码、如何共享、删除等等编程者总是认为，用户应该提高自身的电脑使用技能，而非是自身需要更具科技的成熟改进界面的设置，从而产生了认知摩擦。经理们不注重产品开发的过程，有没有按照预期的功能前进，而是只注重产品完成的期限以及如何的来减少成本。完成！只是简单的完成如何来描述实际产品完成：是否实现了预期的功能描述（第一步完成）、是否被客户所接收（第二部完成）功能并不是第一位，最为根本的是用户背后深藏的需求。而且用户只关注自己的哪些功能能够达到目标。功能并不是越多越好，每一项可能有用的功能都会淡化那些真正有用的功能。失败的产品：电子邮件：没有关联邮件的设置。程序员想当然认为，用户可以手动来操作，把相关的邮件放入同一个文件夹。Gmail做到了！日程软件：没有满足用户的需求，一些咨询公司期望“时间、项目、人员”三要素进行有机结合。软件存在的问题：也就是说，软件要向人工服务一样，用心服务。软件运行时只会遵从程序员的命令，而不会根据用户的使用情况来进行记录，保证下次的使用更加的方便。如果你发现自己在使用某种软件，还不得不在纸张上做笔记，就应该意识到这款软件吝于提供信息。软件不灵活，不会根据用户的需求进行调整软件责备使用者，不应该将责任推给使用者。14、有期望的公司或是产品，才更能存活下去，更能让人产生忠诚度。15、有些客户只提出他们想要的东西，并咩有提起他们正在使用的软件里已经有的东西，他们将这些视为理所当然的功能，而我们却忽略了。16、我们需要做详细的产品设计，用户交互行为，包含了该项目的所有假设。17、大多数人理解不了电脑对程序员的吸引程度。掌握电脑的难度加强了程序员的满足感。他们的感觉很强烈，因而他们认为其他人也有相同的感觉，他们将其他人的挫折理解为没有能力而不是没有兴趣。18、在编程开始之前完整的设计我们的交互产品，将设计责任交给经过训练的交互设计师19、爱因斯坦：不能用与引发问题相同的思维方式去解决问题！1、一些看待问题的新方法。2、一些有效的指导原则。3、一些非常有效的工具。20、只为一个人设计（即目标群体要小而精准），目标用户越多，偏移目标的可能性就越大。如果想得到50%的产品满意度，你就不能让一大批人中的50%对产品满意来达到这个目标，只能通过分离出这50%的人，他她们100%满意来做到。21、用户变得具体后就失去了弹性—即目标比较的精确了，不会出现较多的变化了。。有时候确定到一个具体的角色和人物，我们可以确定她的技能、动机、困难以及她要达到的目标，有了这些信息，我们可以根据软件主题领域对她进行审查，把她套入进去，以便确认她的确是原型用户。积累一定经验后，设计师可以初步归纳出一个有效的角色——这个很重要，在没有通过和用户沟通（节省了很多的成本是时间）的前提下，了解到了软件的真实用户。22、真实的用户——即实际操作软件和真实访谈的用户，很容易将一个用户的能力作为所有用户的代表，这是必须要避免的。23、要把角色用户给设计的精确！！24、每份角色表至少有一个首要任务角色，首要任务角色是设计位置服务的中心人物。经过筛选，我们会得到3--7有用的角色，列出图片、工作描述、目标，还有个人说明，在会议、头脑风暴中无处不在。人物角色其实都是从现实的用户中提取出来的，但并不是简单的是真实用户中的某个人，，可能是几个用户的集合体。一般概括性的整理出10个左右的用户（大致包含了使用此款软件的用户）。建立起这些用户之后，我们就要为这些人物角色创建真实的姓名、身份，这样精确地身份和定位能够为接下来的软件功能设计提供更好的参考。在然后，在一大丢的角色中找到首要角色（挺难的）设计索尼在飞机上的“vido”这个例子非常的详细良好的交互设计”室友在一个人实际使用产品的情况下才有意义，不可鞲只有一如而没有具体的人，两者密不可分。这也是为什么我们设计过程中的两个关键因素是“目标和角色”——意图和人！最重要的个人目标“维护个人尊严：不觉得自己愚蠢”！！良好交互设计的本质是：设计的交互能让使用者在不影响个人目标的情况下，达到他们的实际目标！！！区别“任务和目标”：任务是完成目标的过程，随着技术的进步而有所改变，而目标具有让人高兴的稳定属性！个交互设计方案必须提供所有让ted实现实际目标（能够插上电源就可以看电视的需求），但是设计必须总店关注ted实现个人目标的方法（不要显得自己很愚蠢，因为复杂的操作）个人目标：不觉得自己愚蠢、不出现差错、完成适量的工作、企业目标：增加利润、增加市场份额、击败竞争对手、聘用跟多职员、提供更多的产品和服务、上市若是想要使用者喜欢我们的软件，我们应该把软件的行为设计

的接近人的行为，将软件设计成像一位好的工作伙伴日常场景、必要场景、边缘场景有时是否可以不减少功能的情况下，提高产品的易用性。通过“屈折界面”来达到！：将常用的工具放在显眼的界面，不常用放在不起眼的位置，操作界面可以得到显著的简化。用户应该使用必要而最好的功能区完成任务

32、有那么一些时候，我也有“这让我看上去像个白痴！”的尴尬。比如有次在对着某人的家庭影院遥控器上密密麻麻的按键手足无措的时候。这个我当时，曾经，喜欢过的人问我：需要我教你吗？那一刻我窘的无地自容……屡遭各种“没礼貌”的设计冒犯的Cooper老先生在这本200多页的书里单是抒发不满郁闷就有大概四五十页之多，字里行间都是咬牙切齿之痛。抱怨一通之后，Cooper描绘了让我们体验良好的交互设计的特点。它应该是乐于助人的，善解人意的，灵活应变的，专注而智能的，最重要的是，它应该是彬彬有礼的，它给予使用者最大的尊重。要成就这样的产品，除了技术上的“可能性”及商业上的“可行性”，最重要的还是在设计上给予其“期望性”的充分关注。这需要设计者具有目标导向思维，而不是拘囿于各个具体的实现手段上。“逻辑人”的特性使得软件程序员往往过于专注程序的内部实现而不屑于跳脱出来以一个普通人的角度去看待软件的使用体验。与程序员对应的是，产品经理则面临着过于听从客户意见进而被客户控制产品走向的风险。软件交互设计师的工作正是在程序员和普通用户之间传递有效信息，甚至，发掘用户虽然没有说明但却具有的某种潜在需要或偏好，或是用户所说明的表面需求背后的真正目标，从而使得开发出来的软件产品被尽可能多的目标用户接受和喜爱。不管是对于交互设计师，还是程序员，或是产品经理，这本书都很好地承担了交互设计思想的入门书的功用。同时我也注意到，该书里所持的某些观点有失偏颇。首先是由于作者的交互设计至上立场，本书强调开发前的设计与可预测性(predictability)，对推崇可适应性(adaptability)的迭代开发不置可否，但事实上如果考虑到当前变幻莫测的市场，技术，需求，迭代开发未必不是一种更有效的软件开发方式。此外，作者对微软似乎颇有微词，声称微软唯编程至上，并数次把微软放在反面教材里数落。“任何真正愿意做交互设计的公司都能击败微软”（PS: Cooper先生本人就开了个交互设计公司）。但据我所知，微软最近几年在交互设计上的投入不菲，XBOX和Vista即是两个最著名的例子。之前我参加过的一个微软项目也有专人做UX设计。即便考虑到成书时间（2005），也不能排除微软的树大招风所引来的偏见。当然本书最后对于软件产品的设计，开发，测试各个环节的评断还是很全面客观的。不过从内容的可操作性上来说，作者的另外一本《交互设计精髓》更具体一些，也更值得借鉴。

33、比起《情感化设计》这种纯粹的理念性书籍，这本书的实用性还是很强的。书中对于一个产品的实际开发过程有很精辟的阐述与见解，对于交互设计的各种阻力也分析的很到位。看了之后感觉很多在实际工作中遇到的就是这种情况。

34、前半部分说的是程序员和交互界面发生分歧的根本原因后半部分说了一些设计的原则，比如功能够用即可，目标客户精确，定义首要角色和负面角色，满足首要角色需求，在不影响的情况下为其他角色提供快捷服务。区分任务和目标的不同。为达到目标可以改变任务完成的方式。为目标而做设计，程序员一般为任务而做设计。组织行为学中的“保健因素”和“激励因素”，初始默认值是多么重要。软件要礼貌：记住用户习惯，主动适时的提供帮助区分主要功能和不常用功能（功能的组织）用户看到的越少，说明设计的越好。

35、我相信这本书06年以前还可以成为圣经过于强调让别人肯定他的观点，而这些观点都已经被肯定了有些设计是可以借鉴的其中肯定会有些不被认同的观点作为学习者，我会思考但不会去否定书评还要长长长长长长长长长长长长长长长长长长长长长长长长长长长长长长

36、这本书是06年年底买的，之所以想起来买这本书，是因为06年看见了很多关于易用性方面的文章Web2.0虽然在国内还处于“烧钱”、探索状态，但引发了网站开发、设计人员以用户为中心的设计意识，想必这也是易用性、可用性近年来大行其道的主要原因之一吧。说说这本书对我的益处，见仁见智了（偶是IT民工，不过涉猎较多，技术、经济、商业、人文）1）其实大多数情况下，无论是产品经理还是开发人员的想法与真实用户差距都很远 很少有人做网站、做软件产品设计阶段、原型阶段找实际的使用者测试使用，回忆一下我们自己的设计、开发过程，无一不是以创造者的身份来设计、开发产品的。大小不一的会议开了N多遍，技术与产品经理争的面红耳赤。而其最终的结果无非是诸多比较差的方案中的一种罢了；2）这本书告诉你的不是具体的技术，更多的是一种以人为中心的设计理念 正如作者所述，大多数软件都是为方便开发而设计的，都是为技术高手而设计的，当然我们身边有很多技术高手，如果我们所处的是清华、北大、中关村一带的话；但实际上更多的用户做着跟电脑

技术毫不沾边的事情，对于他们我们这些产品的创造者，考虑的真的很少。 说一下这本书的适用对象：1）对交互式设计感兴趣的非专业认识，比如软件开发人员、比如公司老板等等。这些人不需要实际从事交互式设计工作，但可以从理念上培植一下，免得自己落伍，呵呵。 2）期望从事交互式设计的产品设计人员的入门书籍；从本质上说，产品设计是一个非常具有创造性的活动，对人的天分要求比较高。就我目前所接触的范围内，尚没有发现这样的专职产品设计人员。但后天也可以多少弥补一些的，虽然可能比较中规中矩，呵呵。 看完这本书，可以再看看alan cooper的其他经典设计图书，比如那个<<软件观念革命：交互设计精髓>>
37、这本书，快要读完了，上来写个书评吧作为程序员，我的确不知道还有这么重要的一课要上，我一直都认为用户是第一位的，但是怎么才算重视用户？这个我也说不上来，交互设计指出了一条明确的道路，让我有了一些反思，开始在编程中考虑交互问题。以前我只认为我是搞编程的逻辑人，而现在我也有了勇气打出设计这张牌，作为程序员必须懂得设计，至少懂得交互设计。我推荐此书给那些和我相识的人，如果你是程序员，不懂得界面设计，那么可以先看看这本书的观念，让我们对设计有一个新的认识。此书并不怎么告诉你如何做好交互设计，而主要在说交互设计有多重要。
38、我一直都听大家说这本书是比较好的，但我在书店和网站上都找了很多，都没有。可能是因为停版的关系。想麻烦一下大家，如果知道有该书的购买信息，请通知好吗？谢谢！

《交互设计之路》

章节试读

1、《交互设计之路》的笔记-第209页

如果我们将Maister的方法运用到产品行业的话，我们就会发现，所有客户的要求都是"灰发"工作，"头脑"工作是所有因内部驱动而安排的工作。换句话说，保持技术领先，避免客户驱动死亡螺旋是产品经理的职责。你必须像刚起步时那样，在内部寻找答案。

这意味着有远见，有责任心，付出时间，进行控制。

2、《交互设计之路》的笔记-第140页

前面描述的是第一种设计工具——人物角色

下面将要讲述第二种设计工具——目标。即：如何确定目标，如何将它作为有力的设计工具。

人物角色和目标是一枚硬币不可分割的两面。
人物角色为达到它的目标而存在，目标的存在让任务角色变得有意义。

最重要的目标，是具体个人拥有的个人目标。而最重要的个人目标，是维护个人尊严（不觉得自己愚蠢）

3、《交互设计之路》的笔记-全书归纳与总结

此笔记为实况直播，边看边记。目的就是要告诉自己，作为辩护者，豆瓣读书写笔记的功能不糟糕。
第一篇：列举写书年代一些糟糕的设计案例，指明人机交互的重要性。
提出认识摩擦概念，随着科技环境的变化，人们需要一定时间适应新产品。
直接影响最终用户的设计为交互设计。交互设计要减少认识摩擦。
辩护者，使我想起了win8取消开始按钮，一些桌面挂件提供替代挂件。
谈论电脑文化，或叫它工程师文化，制造了新的统治阶级。
第二篇：产品完成没有固定标准，所以大多以交付期限为准。
为了按时交付，功能列表成了讨价还价的商品。
盲目的迭代、测试降低了产品的易用性，赶跑了幸存者。
事先没计划编写的坏软件，费钱费时。
一大段抱怨。
产品的三品质：技术上的可能性，商业上的可行性，设计上的期望性。
解决客户需求，获取忠诚度，苹果，设计让产品有期望性。
第三篇：设计应先于编程，一旦开始编程，就不要改变设计。启示录的观点同。
数落程序员和工程师文化，依照代码设计程序，没人性。
（Bezos向程序员卖萌，才有了onclick）
改变软件开发过程：专业的交互设计师设计产品；先设计，后编程。
第四篇：定义人物角色，瞄准单一用户群体，具体化人物形象，编造用户故事
建立角色表，确立首要角色，为首要角色设立操作界面
案例：访谈调研，创建角色
良好的交互设计的目标，个人目标优先，在此基础上实现实际目标
目标是结果，稳定；任务是过程，易变
通过分析目标去解决问题，为角色的目标而设计

个人目标：情感、尊严，个人的时间空间
企业目标：利润，市场，竞争，市值
实际目标：个人目标和企业目标的桥梁，具体的商业活动、客户需求
错误目标：程序员的、机器的目标一切都是程序员的错
软件应该更像人一样礼貌：这一段可以作为产品检验清单。153——161
场景：对角色如何通过产品实现目标的描述。分为日常、必要、边缘
屈折界面，功能优先级排列
永久中间用户：大多数人
现实扭曲力场和注意沟通，软硬结合，精简界面（少即多）
第五篇：否定一些可用性导向的思想，这些思想都不注重交互性
概念完整：利益多方愿景一致
交互设计编写成文档，定义产品，描述交互过程，用户对象
交互设计师为产品质量负责
总结：重视交互设计。编程前，花大量时间设计，设计师对产品质量负责。
良好的交互设计体现人性化一面，实现多方利益、目标一致。

4、《交互设计之路》的笔记-第128页

每份角色表至少有一个首要人物角色。首要人物角色是设计为之服务的中心人物，这个角色的目标必须得到满足，但是不能通过为其他角色设计的界面来满足。首要角色应该有专门为他设计的操作界面。如果发现首要人物角色超过三个，就意味着我们的问题过于庞大，我们在试图一下子完成太多的事情。创建角色可以限制我们实际为他们进行设计的用户类别。如果角色过多，我们引入角色的意义也就失去了。

5、《交互设计之路》的笔记-第150页

四种目标——个人目标和实际目标，企业目标和错误目标

1.个人目标——不觉得自己愚蠢、不出现差错、完成适量的工作、有趣（或至少不觉乏味）

个人目标，总是真实的，对每个人有这样或那样的影响。
个人目标，比其他目标重要。
个人目标精确，因为它们属于个人，虽然它们很少被讨论。

如果软件让人觉得愚蠢，即便其他目标得到满足，他们的自尊也会受到挫伤，效果急剧下载。
任何破坏个人目标的系统，即使其他目标满足的再好，也终将失败。

2.企业目标——增加利润、增加市场份额、击败竞争对手、聘用更多职员、提供更多的产品和服务、上市

企业对软件有自己的要求，与个人的个人目标处于同一水平。

设计师根据这个目标，来专注于大方向，免得被任务和错误目标所迷惑。

补充：“保健因素”——有效激励的前提，但是本身不能起到激励作用。比如：办公室的灯光是“保健因素”。一个人不会因为灯光好而去工作，但是如果没有灯光，人们是不会去工作的。
——“保健因素”引申为“保健目标”，“企业目标”和“实际目标”都是保健目标

企业目标和个人目标都很重要：
它们对于企业和个人来说，都是至高无上的，哪个目标都不容忽视，不能实现其中的一个目标的软件终将失败。

3.实际目标——是连接企业目标和个人目标的桥梁

处理客户需要的“ 实际目标 ”，把追求收益的“ 企业目标 ”和提高工作效率的“ 个人目标 ”结合起来。

实际目标——避免会议、处理客户要求、记录客户订单、创建业务的数字模型

4.错误目标
——是达到终点的途径，不是终点，而目标通常才是终点。

如：
节省内存、减少击键次数、在浏览器中运行、

容易学习、保证数据完整性、提高数据录入速度、

提高程序执行速率、使用酷的技术、增强美感

6、《交互设计之路》的笔记-第1页

"难用"问题是由这些产品中存在着高度的"认知摩擦"引起的工程技术,营销能力和设计能力构成高科技企业竞争力的三角品质模型,其中"渴望性"是获得客户忠诚的关键要素.为创建好的产品,懂得程序员的心理是非常重要的.崇尚技术而轻视设计是编程文化的主要特征之一.

7、《交互设计之路》的笔记-第171页

前面讲述了人物角色，并强调目标比任务更重要。

知道了人物角色和目标之后，我们就可以开始对任务进行检查。

把任务合并进来的工具，叫做“ 场景 ”

场景——就是对角色如何使用基于软件的产品达到自己目标的简明描述。

场景须知：
1.比起深度，有效的场景需要在广度方面更加完整。
即：将场景从头到尾进行描述，比起每一步骤详尽地描述更重要

2.将重点放在那些能深入设计工作的场景，避免陷入边缘状况。
即：只编写两类场景：日常场景、必要场景

一、日常场景

日常场景最为有用，也最为重要。
日常场景是使用者需要执行的主要任务，这些任务还会需要经常执行。

通常，大多数使用者，只有有限的日常场景。一般是1-2个，超过3个的情况很少。

日常场景需要最灵活的交互支持。
——1.需要将操作指南编写到程序中。
——2.使用者会需要定制日常使用的交互操作来适合个人工作习惯和工作偏好。

二、必要场景

必要场景是“ 所有 ”那些不常用但必须具备的场景。
——1.也要求有灵活的操作指南
——2.因为不常用，所以不会考虑去定制操作，会更愿意迎合程序的做事方式。

三、边缘场景

我们可以在产品设计过程中忽视边缘场景。

不是说在程序中省略相应的功能，而是说我们可以很粗略的设计交互。将它们放在操作界面的背后。

——处理边缘场景的能力，决定了“ 程序 ”的成功和失败；
——处理日常场景和必要场景的能力，决定了“ 产品 ”的成功和失败。

8、《交互设计之路》的笔记-第122页

作为设计工具，让角色精确比让角色正确更为重要。就是说，详尽而具体地定义角色比让角色非常正确更为重要在定义角色时我们追求精确，但是我们要排除平均状况。平均用户从来不会真正“ 平均 ”。平均角色会损害精确角色所具有的优点：具体。角色的巨大力量在于它们的精确程度和具体性。用集合的方式来处理会损害角色的有效作用。

9、《交互设计之路》的笔记-第71页

第2篇中的3个章节共同描述了一幅坏软件使得用户和企业都苦恼不堪的画面。coper把差软件比作是“ 跳舞的熊 ”，甚至比它更坏。站在企业的角度看，坏软件浪费了企业有限的资金、造成了客户的叛离，等等。

chapter 3：浪费金钱

1.讲述了项目中出现的期限管理问题和帕金森定律。管理人员总是担心“ 产品能否及时完成 ”和“ 产品是否让人满意 ”两大问题。相对于后者，前者显然更为直接明显，于是管理人员开始了期限管理，希望越快越好，并且认为这有诸多好处，如降低成本、促使项目组加紧工期等。（神哪，很多员工都是希望能拖就拖的！）接着，就描述了项目呈现的帕金森定律：90%的时间能完成90%的工作，但剩余的10%工作还需要另一个90%的时间来完成。从现实中看，确实有许多低素质的员工总是不断的拖拉、延缓项目的进度，项目经理以交付日期作为施压的手段其实也是无奈之举啊。

2.功能需求的讨价还价。聪明的程序员知道怎么为自己的工作安排时间和进度，当看到前端交付的设计之后，他们往往针对这些设计讨价还价，要么是删除一些设计来保证按时完成，要么是推迟日期来保证设计。于是就有了各方的博弈，装腔作势、激烈争论神马的，直至最终交付。另外，coper给交互设计提了个中肯的建议：功能多未必就好！这点深表同意啊！的确有很多软件实在是让人眼花缭乱，

让人迷路。如果功能多到让用户觉得迷惑，估计它离死也不远了。

chapter 4：跳舞的熊

本章就是把坏软件比喻为跳舞的熊，甚至不如它。又举了N多例子，其实是chapter 1的延续和深化，就不多赘言了。

chapter 5：客户叛离

这是一个困扰不少企业的难题，大家总在想：为什么我的客户越来越少呢？

1.作者首先引进了“ 可能性 ”（技术）、“ 可行性 ”（商务）和“ 期望性 ”（设计）三个概念。coper用它们来分析Novell公司从辉煌到衰落的原因。Novell是一家技术力量强大的企业，在相关领域率先开发出产品并满足了用户需求，是有着比较好的“ 可能性 ”和“ 可行性 ”，但是它也有个致命的确定，就是“ 期望性 ”太低。随着竞争对手在交互设计上的突破，Novell轻易的丢失了自己的市场份额。

2.三家公司对比。作者分析了Novell之后，顺势又将微软和苹果加入了对比的行列。看了他的讲述之后，我才对windows的流行若有所悟，原来，一般的“ 可能性 ”+一般的“ ”期望性“ +强大的“ 可行性 ”是可以成就一家企业的，不过也补充了另一个原因，就是微软齐全的服务（指软件），很少有公司能做到这一点。但这也开了个不好的头，许多公司拿着不好的产品试图重温微软的梦，寄希望于营销，结果跌的好惨！

coper还是很有先见之明的，他在06年就预测了苹果日后的崛起，因为它有着极好的 ”期望性“ 和 ”可行性“ ，即使技术不是那么强大。因为良好的设计给它带了极大的客户忠诚度，以至于用户愿意花费时间去等它做出改变。

3.面市时机。许多企业欣赏在第一时间满足客户需求，抢占市场份额，让后来的或者是实力弱的公司只能喝汤的做法。这没什么错误，但是如果先驱者一直不重视设计工作，仅仅是不断增加新功能，那么也会给别人后来居上的可能，因为这是一个致命的弱点（从Novell的例子可以看出）。而后发公司想要夺取市场份额，就必须有着自己卓越的设计能力，否者只靠增加新功能是无用的，而且也加不过市场原来的老大！（可以看看身边的例子）

10、《交互设计之路》的笔记-第177页

即便我们不能摆脱限制，通往绝路的旅程也可能照亮原先隐藏的机会。

11、《交互设计之路》的笔记-第115页

爱因斯坦说过：不能用与引发问题相同的思维方式去解决问题。

1.一些看待问题的新方法
2. 一些有效地指导原则
3. 一些非常有效地工具

12、《交互设计之路》的笔记-第1页

认知与摩擦

13、《交互设计之路》的笔记-第204页

《交互设计之路》

最后一章——不顾一切的追求可用性：

展示每一种具体方法在什么地方起作用、如何与目标导向的交互设计相配合。

一、设计的时机——编程之前

二、用户测试

精心的用户测试，能够发现设计师的不正确假设。将设计出来的东西交给用户，根据反馈反复设计，这样总比不做要强。

可用性测试，是对付那些顽固的程序员的有力工具，让他们看到他们的程序的确有问题。这一方法对管理层也有同样的效果。

三、多学科团队

以相同比例的用户、程序员、经理、市场人员、可用性专家组成的团队，无法实现编程前的设计。

因为每个成员的目标和关心点都不一样，这会让拥有产品最终控制权的程序员起着主导作用。

四、程序员做设计

事实上，很多程序员设计得很好，而设计不好的程序员知道自己的弱点，会回避设计工作。
需警惕的是，程序员做设计时，他们的努力几乎肯定是基于“ 逻辑人 ”独特的个性。产生了难以使用、不合适的产品，而其他程序员却真的喜欢。

五、你是怎么知道“ 交互是好还是不好 ”？

交互设计师根据经验、训练和判断，进行正确的评价。
他们有应用于各种情况的原则、习惯用语、工具，而且他们会通过其他信息将这三者结合起来。

六、界面风格指南

风格指南虽有所帮助，但是它们不能解决目标导向交互设计所能解决的问题。

一种通用的视觉语言和固定的控件会有帮助，但是，只有这些并不能解决问题。

七、利益冲突

Alan Cooper“ 并不 ”鼓动大家无视风格指南，而不管用户界面的混乱局面。

只是对待“ 风格指南 ”应该存在考量的态度。

八、焦点小组

焦点小组技巧用于创新产品的效果让人怀疑。
现今，大多数基于软件的产品很有创新性，不适合焦点小组。

焦点小组在某些产品上很有效，但不要把它们当做对高认知摩擦产品的可靠评估。

九、视觉设计

十、工业设计

十一、很酷的新技术

无论使用什么样的技术组合，技术需要经过设计才能成为完整的解决方案。不能简单的认为新技术比起其他技术能更好地解救我们。

十二、迭代

采用“ 消耗策略 ”的迭代，不仅代价昂贵、消耗时间，而且让人憎恨。因为这种策略实际上时虐待使用电脑技术的人。典型采用消耗策略的如：微软

14、《交互设计之路》的笔记-第186页

逻辑扫描仪案例，功能：重新定向，图片。
小圆有点像图钉，一直认为拟物设计最好用。

15、《交互设计之路》的笔记-第145页

一、任务不是目标

目标不是任务。目标是终结条件，而任务是达到目标所需要的中间过程。

区别任务和目标非常重要，人们很容易将它们相互混淆。

有一种简单的方法可以区别任务和目标。任务随着技术的变化而变化，而目标具有让人高兴的稳定的属性。

目标是稳定的，任务是易变的。
很显然，我们应该为目标进行设计，而不应该为任务而设计。

二、程序员做“ 任务导向 ”的设计

区分两种问题：“ 有哪些任务？ ”VS“ 有哪些用户目标 ”

三、目标导向设计

当交互设计师通过分析目标去解决问题时，通常会找到非常不同，但是却好得多的解决方案。

四、个人目标与实际目标

前面提到，良好的交互设计的本质，是要让用户达到设计目标，但是同时，不能破坏他们的个人目标。

探求Ted的目标：

实际目标：想看电视节目。
故，花钱购买了一台新的电视机，因而他自然的想享用这台电视机具备的所有新功能。

个人目标：不希望他的新电视机让他感觉羞耻；不希望觉得自己愚蠢，不希望出现差错；他希望尽快有成就感，从中得到乐趣。

从交互设计师的角度来看，这些个人目标比Ted的实际目标更重要。

交互设计方案必须“提供”所有让Ted实现“实际目标”的途径，但是设计“重点”关注Ted实现“个人目标”的方法。

五、平等付出原则

16、《交互设计之路》的笔记-第81页

不带有色眼镜审视过去的几个月，那情景有点像一部古老的怪人滑稽电影，或者像一出没有浪漫情调的肥皂剧。这是我能给予的最高评价。当然，如果这就是我的真实看法，我还不会这样详尽地对其进行描述。事实上，我是真的动情了。我感觉有一种道义上的冲动，让我制止在那些无用的活动中浪费人们的生命。 看得出来小作者真得很喜欢交互设计,用动情这个词丝毫不过分，真挚！:p

17、《交互设计之路》的笔记-第190页

几个有用的设计概念：屈折界面、永久的中间用户、词汇、头脑风暴、横向思维

一、屈折界面

“屈折界面”的技巧——可以在不牺牲功能的情况下，提高产品的易用性。

将常用的控件和数据放在显著的位置，将其他空间和数据放在不起眼的位置，由此来简化操作界面。

二、永久的中间用户

多数用户既不是初学者，也不是专家，而是在两者之间。

三、“假装它有魔力”

使用一种称为“假装它有魔力”的创造性思维练习。

四、词汇表

如果我们其中一部分人队术语的理解不同于其他人，那么我们讨论问题的效率则不高。

这就需要一个共同的词汇表。

Alan Cooper坚持将设计元素分解到无法再分解，然后给每一个部分采用新的命名。

五、突破语言障碍

我们将设计过程、任务、软件分解成定义完好、具体的模块，然后赋予它们没有关联意义的新名字。这些新名字通常也很幽默，这样不至于导致人们对名字过于较真。

六、现实检测

“我们做不了”已经成为口头禅。而交互设计师，必须对所有不能完成的假设保持健康的审视态度。

通常的做法是：假设所有事情都是可能的，然后开始设计。

避开所有假设条件，可以更清晰地看到角色和目标，我们可以想象那些通过传统的方式无法得到的解决方案。

“界面更少”，并不是说“功能更少”（虽然有时会是这样）。
Alan Cooper的意思是：用户应该只做必要的而又最少的操作去完成任何一项任务。

18、《交互设计之路》的笔记-第22页

我相信在因特网上可以销售任何东西并获利和成功。实现的秘笈就是在线商店必须比传统商店提供更令购物者满意的可衡量的服务，价格只是满意内容的很小一部分

19、《交互设计之路》的笔记-第200页

第一种设计工具：人物角色
第二种：目标
个人目标在商业目标之上，维护个人尊严,不觉得自己愚蠢。
目标是终结条件：任务是达到目标是中间过程
任务随着技术的变化而变化，而目标具有稳定的属性。目标是稳定的，任务是易变的。
场景：角色如何使用基于软件的产品达到自己目标的简明描述。
场景是从初期调研阶段收集的信息中建立起来的。
日常场景和必要场景。
日常场景需要最灵活的交互支持，必须让新用户能迅速掌握。
必要场景：是那些不常用的，但是必须具备的场景。

边缘场景：处理边缘场景的能力，决定了程序的成功和失败，但是处理日常场景的和必要场景的能力，决定了产品的成功和失败。

客户：不关心你的长期利益，不知道如何设计产品。
如果你被客户驱使，你的产品在版本之间就会发生很大变化，而不可能有序地成长。最终，产品里充满了不协调的部分和随机的功能。
如果成为客户驱动的服务，那么在短期内赚钱比较容易，但是会停止成长，等于放弃为了，放弃自己引导潮流的地位。（例子：如诺基亚死守windows mobile8）

20、《交互设计之路》的笔记-第206页

永远应该优先满足的是用户的需求。毕竟，虽然其他人站在那里要求行动，但客户是惟一手里拿着支票的人。每个商业人员没法不受他的影响!

21、《交互设计之路》的笔记-第230页

Part 9 为快乐而设计

角色展现它们自己的方式很像地壳构造事件通过对沉积岩层的研究向地质学家展现它们自己：一种化石的存在定义一个地层，而一个地层又定义了一种化石的存在。（其实也可以用年轮来做例子。）

Part 10 为能力更强而设计

良好交互设计的本质是：设计的交互能让使用者在不影响个人目标的情况下，达到他们的实际目标。

22、《交互设计之路》的笔记-第120页

-=让人物角色具体化=-
我们的人物角色越具体，他们作为工具就越有效。这是因为角色变得具体后就失去了弹性。我们不只是说Emy使用办公软件，而是说Emy用Excel2010写说明文档。我们不只让Emy开车上班，而是让她开着深黑色的1991年丰田佳美去上班，后车厢里固定着一个灰色的儿童座位，后保险杠上有一条难看的划痕。我们不仅仅说Emy上班，而说，她在田纳西州Memphis城的环球航空公司财务部米白色小隔间里做科员。这样和别人区别开来的详细描述是非常有力的设计和沟通工具。因而，我们对所有角色都进行唯一细致而精确的描述。

23、《交互设计之路》的笔记-第129页

对操作水平的实际了解：

用户操作水平的范围很宽，角色让操作水平变得很具体。

传统上，人们广为认同的用户技能模型是：金字塔模型。
即：
1）顶部是“ 高级用户 ”，人们认为他们精通电脑操作，但是缺少编程方面的培训；
2）中部是“ 电脑文化用户 ”，他们具有操作电脑的基本技能，但是对有些很酷的功能不太熟悉；
3）下部是“ 初级用户 ”，他们是一些头脑迟钝，面对电脑不知所措的人。

但——金字塔模型是一种错误的假设，它的分类有些牵强，它并不能代表它想代表的人群。过于简化的市场模型无助于解决涉及问题。

角色终结了功能争议：即将角色具体化，让项目组的成员均习惯将角色看做真实的人来设计她需要的功能。而不仅仅泛谈“ 用户 ”可能需要的功能。

是用户角色，而不是购买者角色：

创建角色时最容易犯的错误，是将与产品接近的人作为角色，而不把实际使用的人作为角色。

角色表：

我们为每一个项目创建一个角色表。
角色表里一般有3-12个不同的角色。
我们并不为所有的角色进行设计，但是这些角色可以代表我们关注的人群。
有些角色列入到角色表，仅仅是为了表明：我们将不为他们做设计。

首要任务角色：

每份角色表至少有一个首要人物角色。
首要人物角色是设计为之服务的中心任务，这个角色的目标必须得到满足，
但是，不能通过未其他角色设计的界面来满足。
首页角色应该有专门为他设计的操作界面。

如果发现首要人物角色超过三个，就意味着我们的问题过于庞大，表明我们在试图一下子完成太多的事情。
创建角色可以限制我们实际为他们进行设计的用户类别。
如果角色过多，我们引入角色的意义就失去了。

经过筛选，我们通常得到3-7个游泳的角色。
我们通常将它们汇总在一页纸上，
列出他们的姓名、图片、工作描述、目标，还会经常加入一些个人说明。

这份一页纸的文档无处不在，把角色表打印出来，每次头脑风暴会议、详细设计会议、客户代表出席会议等各种讨论时分发。
目标就是让角色变得不可抗拒。

129页及之后有具体案例分析

24、《交互设计之路》的笔记-第88页

据我个人的观察，我总结出4条程序员与普通人不同的基本思维和行为方式：程序员牺牲简单而获得控制权；程序员牺牲成功换取理解；程序员只关注可能性，不管概率；程序员行为就像体育生。

25、《交互设计之路》的笔记-第1页

26、《交互设计之路》的笔记-第118页

摘录：

《交互设计之路》

1.想让更多用户喜爱的结果可能会将一个潜在的好产品置于死地；
但是，如果将设计目标锁定在一个角色，那么，这个角色的人群会对你的产品相当满意。

比如：豆瓣么？前期的小众产品，让小众相当满意的产品。

2.让人们喜欢你的产品，即使只是少部分人，这就是成功之道。

3.开心的用户是非常有效而有价值的资产。通过将目标聚焦在一小群用户，你可以在目标市场中建立狂热的用户忠诚度。

客户忠诚度，将帮助你在困难时期渡过难关。

忠诚的客户，不仅会跋山涉水来购买你的产品，他们还是最有力的营销工具。

忠诚的客户，会将你的产品介绍给他们的朋友。当你的产品开始走俏时，就可以靠“客户的忠诚度”将产品延伸到别的市场。

27、《交互设计之路》的笔记-第90页

“登机通道测试”——向左：驾驶舱；向右：客舱

向左转的一类人：他们对操控和了解技术有强烈欲望；

向右转的一类人：他们愿意思考，相信客机会安全到达目的地。

程序员——逻辑人——总是向左转
用户——普通人——总是向右转

看着这段，我作为普通人会下意识的右转吧。
可是，即使程序员是逻辑人，他们会向左转想操控飞机么？

还是说，Alan Cooper只是从这个例子来展开讨论？

接着看下文，“程序员心理学”。

交互设计的目标，是创建 强有力- 且让用户使用起来愉悦的 - 基于软件的 - 产品。

产品成功的先决条件：

1.正确理解用户的心理；

2.理解软件产品创建者——程序员的心理活动；（这一点更重要，且容易被第一点干扰）

作为交互设计师，我们不仅要理解用户的心理，还必须理解程序员的心理。

只有进入到他们的思想深处，找出 激励他们去创建让用户满意的交互产品的 方法，才能真正解决问题。

28、《交互设计之路》的笔记-第55页

硅谷有一则古老的笑话，问题是，“怎样才能在软件上小获一笔财富？”答案当然是，“与巨富同行！”隐藏在即使管理得非常好的软件开发项目背后的成本，也足以使Donald Trump(美国著名房地产开发人物)踌躇。因而，从长远角度看，帆船比赛和吸毒习惯，比缺乏合适控制的编程更廉价。
没有做好交互设计的软件，在使用中往往会带来巨大的成本浪费甚至引发灾难(如糟糕的导航系统引发空难的案例)。
而设计产品的时候应当注重的不是其功能，而是使用该产品的用户的目标(割草机的案例挺不错)。
程序员在编程的时候往往会将自己的Geek风格融入产品，而产品经理(或是交互设计师)的工作就是要避免这类事的发生。

29、《交互设计之路》的笔记-第113页

第3篇《用叉子喝汤》：光看标题大概就知道这是要说一件不靠谱的事了。对！作者在第3篇的3个章节里共同讲述了由程序员做设计会产生的不靠谱。

chapter 6：精神病人管理着精神病院

这是一个形象的比喻，让读者明白程序员掌握设计工作是件多么不应该的事。这章里coper讲了几个故事，表达出没有设计将导致项目歇菜的思想。因为程序员是不会真正思考清楚“目标是什么”、“为谁服务“等这一类核心问题的。接着，coper还深入分析了一个软件为什么需要设计员和程序员同时存在。这是因为，软件的一个方面——内在，它需要一定的技术，要考虑电脑的需求；另一个方面——外表，它需要考虑人的需求。这是两种截然不同的属性，正常是不应该由一个人完成的，所以说程序员做好前者的工作，而后者就交给交互设计师来完成吧。

chapter 7：逻辑人

此章介绍了什么是程序员。看过这章之后，不知道的大概会觉得他们是monsters，呵呵。
1.程序员牺牲简单换取控制权：以做飞机为例，程序员为了获得控制权，情愿去掌握复杂的操作技术，不要简单舒适的享受。2.牺牲成功换取理解：成功对他们吸引不大，程序员更喜欢了解其中的运作机制。以闹钟为例，他们会选择拆毁一个闹钟来了解它的运作。3.程序员只关注可能性而不考虑概率：举个例子，”被雷劈死“这件事概率应该足够小吧？！常人是不会为此做出一套预案的。但程序员不是这么想的，只要有可能就必须要对策。这大概也是一些软件多累赘而跑不起来的原因吧。4.程序员像体育生：coper认为人家思想古怪，爱与人争吵，喜欢显摆技术。

chapter 8：过时的文化

coper这章从程序员习惯、态度等软文化的高度上讲述”用叉子喝汤“。来看看程序员的著名表现吧：代码重用而不顾交互效果！他们为了效率或者偷懒，擅自更改设计，这是现实中一直存在的，确实让人深恶痛绝！除此，他们喜欢分享共同文化，在各个领域、行业植入自己的逻辑思维，所以就有这么多不好用的软件了。基于此，作者感慨到：这是红果果的文化差异啊！(有差异就会有隔离...)

30、《交互设计之路》的笔记-第230页

尽管交互设计师占据主导地位，他们绝不可能比程序员更了解程序的内部。他不可能从程序员的角度看得更清楚，为了获得成功，他必须从另一个角度看待问题。

交互设计师能够与程序员平起平坐的基础是，提交精确而完整的交互设计文挡。设计师只有提交让人信服的解决办法，程序员才会逐渐信任和依靠他们。

31、《交互设计之路》的笔记-第90页

作为交互设计师,我们不仅要理解用户的心理,还必须理解程序员的心理.有这样一个在程序员之间广为流传的笑话,只存在3个数组:0,1和无穷大.程序员牺牲简单换取控制权
程序员热衷于理解内部机制
程序员只关心可能性而不考虑概率
程序员像"体育生"

32、《交互设计之路》的笔记-第33页

前两章主要是讲述了现今的软件给人们生产、生活带来了不计其数的烦恼，主要是因为他们不良的交互设计造成的，而这一点现在却自然而然的被社会大众接受（好吧，我就是“社会大众”了）。

chapter 1：本章主要是讲述例子，各种交互不便的例子。
机舱的电子导航造成飞机失事、单反相机不便使用、电子闹钟反馈差劲、汽车的智能系统不受控制、自动取款机的不便操作等等，所有这些给作者或者别人带来巨大的烦恼。对于这一部分，我确实觉得有些发明是那么不方便使用，也比较赞同作者说的话，但是又窃以为这样一拍子打死所有发明，尤其是高精尖方面的，好像有失偏颇。大概是因为目标客户划分的缘故吧？再补充...

chapter 2：围绕认知摩擦，展开叙述。
1.什么是认知摩擦？作者定义为“它是当人类智力遭遇随问题变化而变化的复杂系统规则时遇到的阻力”。简而言之，应该就是使用新科技时的阻力吧。
2.coper又为我们讲述了他理解的设计，这点倒是于我心有戚戚焉，可以看出作者对于程序员童鞋插手前端设计是非常不爽的，他们该专注自己的程序设计工作。因为交互设计师的职责是为用户考虑，而程序员却极容易把自己看做是用户进而干扰前端设计。
3.为读者讲述了什么是交互设计。其实这个问题我认为是仁者见仁的，不过大致会达成基本共识的：交互设计应该包含：用户需求分析+行为流程设计+界面设计。神马翻页、动画等feedback不过是界面设计的内容而已。作者举了数据报表的例子，这个对理解有挺大帮助的。
4.软件添加新功能的代价。企业认为添加新功能的成本很低，因此喜欢大量扩充功能、频繁的更新改版，而其中有一大批是不怎么实用的，或者是增添之后整个信息架构显得很凌乱，这点在手机客户端表现的十分明显哪！最近对支付宝的改版就有这种感觉，“超级转账”和“摇一摇”的确很好，不过你好歹规划一下自己的架构吧？！（好像现在已经改了...）
5.coper认为新技术将人群进行了分层，有适应了的，也有淘汰了的。他还非常不赞成对用户进行划分的做法，认为不应该有“初级用户”、“高级用户”之类的。对于大众型产品，完全理解；但是对于用户差异性很大的产品呢？比如贵金属投资神马的，该不该划分呢？呵呵...

33、《交互设计之路》的笔记-第123页

弹性用户

“用户”一词还不够精确，会让人觉得是“弹性用户”。

也就是说：
有时候，弹性用户被定义为有足够电脑知识的高级用户；
有时候，弹性用户又被定义为无知的、首次使用电脑的用户。

为弹性用户设计，等于给了程序员根据自己需要随意编程的许可。

记住：真正的用户不是弹性的。

Alan Cooper说：在我们的设计过程中，我们从不使用“ 用户 ”一词。我们用“ 人物角色 ”一词来指定某个具体的人物。

让任务角色具体化：

1.给角色命名，是成功定义一个角色的关键。没有名字的角色毫无用处，因为如果没有名字，这个角色不会在人们的心中成为一个具体的人；

2.用不同种族、性别、国籍和肤色的人作为角色。且更侧重可信度，而不是多样性。
比如：
如果一位角色是计算机技术人员，则把角色叫做Nick，23岁，脸上长满青春痘，在高中时代是视听俱乐部的会员；
而不是叫做Hellene，一个5英尺11英寸高，身材苗条，曾在好莱坞Beverly Hills高中念过书的女孩。

3.给每一个角色一副肖像，会让参与产品开发的人们觉得每一个角色更加真实。

——让人觉得真实、定义完整的用户角色是非常有力的工具，是抑制程序员曲解用户角色作用的关键所在。

假想的任务：

1.不要将精确的用户分类与真实人物混淆起来。

2.真实人物带来的主要问题：
-----1）真实人物作为原始数据进行参考，但是通常没有多大用处，反而对设计过程有害；
-----2）真实用户带有干扰设计过程的不良嗜好和特殊行为。而这些特点并不适用于同一类的其他人。

精确而不是正确：

也就是说，详尽而具体地定义角色比让角色非常正确更为重要。

（这和交互设计的目标相反，即：目标的正确性总是比精确性更重要）

程序员非常关心边际状况，这会影响他们对角色的选择过程。
但，不管编程是不是再范例的边缘定义，而设计却是在中心定义的。

所以，如果怀疑一个角色与中心是否足够接近，这个角色就不应该被放在考虑范围之内。

角色的巨大力量在于：他们的精确程度和具体性。用集合的方式来处理会损害角色的有效作用。

角色是我们设计最有力的设计工具。它是所有后续目标导向设计的基础。

角色让我们看到设计问题的范围和性质。

角色清晰地揭示用户目标，因而我们能看到产品必须做的事情，同时我们也清楚产品不该做什么。

精确定义的角色，准确地告诉我们用户的电脑操作水平。因而，我们不会再为专家设计还是为新手设计的抉择之间犹豫不决。

34、《交互设计之路》的笔记-第1页

不错

35、《交互设计之路》的笔记-第171页

场景是从初期调研阶段收集的信息中建立起来的。通常，在与用户访谈和直接观察中，我们会得到很多有关任务的信息。目标比较稳定持久，但是任务是流动的，可改变的，很多任务在电脑系统中并没有必要。在编写场景时，我们需要找出那些因历史原因而存在的任务，并将它们剔除。

比起深度，有效的场景需要在广度方面更加完整。换句话说，将场景从头至尾进行描述，比将每一步骤详尽地描述更重要。

我们应当将重点放在那些能深化设计工作的场景，而避免陷入边缘状况。我们只编写两类场景:日常场景和必要场景。日常场景最为有用，也最为重要。它们是使用者需要执行的主要任务，这些任务还会需要经常执行。例如，在一个缺陷跟踪系统中，查找缺陷和填写新缺陷报告单就是日常场景。任何技术支持人员都会在一天内多次地执行这两项任务。

通常，大多数使用者只有有限的日常场景。一般不过是-个或两个，超过三个的情况很少。必要场景是所有那些不常用，但是必须具备的场景。清空数据库和发出异常请求就属于必要场景。必要场景也要求有灵活的操作指南。但是，使用者不会追求熟练使用这些场景。因为不常用，使用者会愿意迎合程序的做事方式，不会考虑去定制操作。因而，程序员们也不必在必要场景上花费与日常场景相当的精力和资源。这种区别就像一部新车的豪华内饰和发动机舱内的简陋修饰一样。

虽然多数产品必要场景也不是很多，但是它们一般都多于日常场景的数量。当然，有第三种场景:边缘场景。程序员自然会很重视边缘场景，但是我们可以在产品设计过程中忽视边缘场景。这可不是说相应的功能可以在程序中省略，而是说我们可以很粗略地设计交互，将它们放到操作界面的背后。处理边缘场景的能力，决定了程序的成功和失败，而处理日常场景和必要场景的能力，决定了产品的成功和失败。

如果使用者频繁执行某一任务，这个任务的交互就应精心雕琢。同样，如果一个任务是必需的，但是不常执行，它的交互也需要细心设计，虽然目的不同。对于为处理边缘状况而产生的不常用的任务，不需要精心设计。时间和金钱永远不会充裕，边缘状况是我们可以安全地节省资源的地方，并把它们集中起来发挥最大用处。我们必须支持所有场景，但只需要为那些重要和常用的场景进行设计。

36、《交互设计之路》的笔记-第118页

让人们喜爱你的产品，即使只是少部分人，这就是成功之道。目标用户越多，偏移目标的可能性就越大。如果想得到50%的产品满意度，你就不能让一大批人中的50%对产品满意来达到这个目标，只能通过分离出这50%的人，让他们100%满意来做到。可以再进一步，瞄准10%的市场，让他们成为产品的狂热追随者，你能获得更大的成功。这似乎有悖常理，但是只为一个人设计是让大众满意的最有效方法。

37、《交互设计之路》的笔记-第74页

Doblin Group公司的Larry Keeley为高科技企业设计了饶有趣味的三品质概念模型。
可能性是指技术上是否可能，可行性是指商业上能否盈利。
程序员喜欢增加产品的功能，提高软件的性能，他们个人的目标常常与企业的目标不完全一致。
商务人员喜欢拥有市场份额，而不注重产品的具体特性。
期望性由设计师带来，设计通过让产品成为人们真正需要的东西，而带来销售利润。它是平衡程序员和商务人员目标的重要品质。
期望与需求的区别：需求是用户短期内的一种需要(被动性)，而期望是用户的一种长期意愿(主动性)。
掌握了用户的期望，就能得到用户的忠诚度，从而在同类产品的竞争中才能存活下来。而这种期望的把握就体现在产品的交互设计上。

案例：Novell, Apple, Microsoft
有人说，大多数产品仅仅是满足了用户的需求，而Apple的设计则是挖掘未知的用户需求，就有点像这里讲的期望性。这正是Apple的牛X之处。

38、《交互设计之路》的笔记-第22页

为向用户提供强力而愉悦的软件，交互设计师首先创建概念，然后考虑行为，最后考虑界面。

39、《交互设计之路》的笔记-第190页

这是真正涉及到交互设计知识的一个章节，总共分3章进行讲述。

chapter 9：为快乐而设计

1.人物角色：通过调查真实用户，总结归纳出代表某一类真实用户的假想原型，一个产品可以有一个或多个人物角色（不超过4~5个）。这么做有利于避免不慎将某一有着独特特性的真实用户当做测试对象。在创建了角色之后，应当将其具体化。举个例子：“ Emily上班 ”是没有意义的，需要这样表述：“ Emily在Maimi的环球航空公司财务部白色小办公室做科员 ”。为什么这样说呢？角色必须真实，没有弹性，有着自己特定的目标。详尽的描述可以避免摇摆不定的情况发生，可以终结争议，尤其到后期可审查某一类人是否是目标用户。这也就是coper所说的精确性，让设计员清楚知道它的行业、目标、行为特点等等信息，不要给人捉摸不定的感觉是关键。

2.为目标市场设计，务必做到专而精。coper的这个观点有人认同也有人不认同，some App就是面向大众的，some就是面向细分市场的。但是我还是比较倾向于选好自己的受众，尤其是刚出道的企业。对于一些大企业也是，当App里什么功能都有了，是好是坏很难说。在设计过程中，还要注意选对首要角色，这一类人应当优先满足，有侧重，让他们觉得能快乐的使用。书中举了sony的例子，有助于理

解。为快乐而设计，应该指的就是专注于角色目标，不要泛化，因为这样才可能带来快乐。

chapter 10：为能力更强而设计

1.良好的交互设计肯定具备这么一个特点：能让使用者在不影响个人目标（如不觉得自己愚蠢）的情况下，达到他们的目标。个人目标是很关键的要素，这是衡量交互设计好坏的关键因素。如果用户在使用过程中觉得自己愚蠢、工作繁琐、无趣，那就危险了。

2.礼貌的App：对用户感兴趣；尊重用户；拥有常识；预知用户需要；反映敏捷；专注提供自己的核心功能；即时回报（切忌过多选择、步骤）。这是对一个软件好坏的评价标准，虽然不可能完全都具备，但是其中权重大的部分满足了，这就是个好应用。由此也可以预见，在增强能力的同时还有打造一个好App是不容易的。

chapter 11：为人而设计

1.场景的区别：coper把场景分为日常场景、必要场景、边缘场景等几类。日常场景说白了就是日常使用的几个功能模块，一般为2~4个，比如手机银行，日常场景就是查询、转账、信用卡还款这么几个。必要场景应该就是更多里面的吧，比如删除缓存、设置密码神马的。场景的划分还是挺有必要的，为日常场景应当提供最好的交互，必要和边缘的再区别对待吧。

2.屈折界面，少即是多。什么是屈折界面？一个产品提供很多功能，但并不是用户在所有场合或所有时间都需要它们。对于任何给定的场景，用户角色将使用一小部分功能，并通常把它们至于显要的地方，其他的可以隐藏或者放到不起看的地方。什么是少即是多？总体而言，在保证核心功能的基础上，努力做到界面少、流程少、页面元素少，让用户可以轻松、自由切换、游转，这本身就是矛盾的两方面，因此也是最难的部分。“ 酷炫 ”界面那是美工的工作，所占权重小之又小。

40、《交互设计之路》的笔记-第159页

一、为礼貌而设计

如果我们想让使用者喜欢我们的软件，应该把软件的行为设计得接近人的行为。

如果想让使用者通过我们的软件提高工作效率，应该将软件设计成像一位好的工作伙伴。

“ 礼貌 ”绝对不仅仅只是说“ 请 ”、“ 谢谢 ”。

如果交互对人尊重、宽厚、富有帮助，使用者就会喜欢这个软件，拥有愉快的体验。这与操作界面的构成没有关系。

二、什么是礼貌

三、什么让软件有礼貌

Alan Cooper 对礼貌软件的描述：

礼貌的软件，对我感兴趣
礼貌的软件，尊重我
礼貌的软件，主动提供帮助
礼貌的软件，拥有常识

礼貌的软件，会预知我的需要
礼貌的软件，反应敏捷
礼貌的软件，会预知我的需要
礼貌的软件，反应敏捷
礼貌的软件，会解决自己的问题
礼貌的软件，提供有用的信息
礼貌的软件，有洞察力
礼貌的软件，有自信
礼貌的软件，很专注
礼貌的软件，灵活应变
礼貌的软件，即时回报
礼貌的软件，让人信任

41、《交互设计之路》的笔记-第220页

交互设计师从哪里来

在我的公司里，几位设计师的工作背景是技术编写文档、软件项目管理、技术支持、图形设计。在我的公司，很多设计师具有人文专业的学历，也有一些设计师具有物理、建筑、计算机和工业设计专业的学历。
有技术支持和文档写作的经验，设计师能够从一般用户需要的角度思考问题。软件产品经理知道程序员在开发过程中的需要和担心。图形和工业设计师热爱优美的设计，有能力去设计。在高科技领域工作过，受过人文教育的设计师能够将技术知识与表述想法的能力结合起来。

这么一看，只有在高科技领域工作过受过人文教育的设计师这条路最容易走，然而我在接受人文教育的时候一点也不走心，反而因为它们和逻辑无关而感到无趣，不过还好，我觉得现在还是可以回头的，种一棵树最好的时间是十年前，而后是现在。#路还很长#

42、《交互设计之路》的笔记-第1页

*** 行为设计告诉你软件的元素应该如何表现和交流。交互设计师们从外向里工作，从用户的目标开始。
*** 概念设计首先要考虑什么才是对用户有价值的。
*** 交互设计师首先创建概念，然后考虑行为，最后考虑界面。

*** 所有电脑的首要目标是不要自我感觉像个傻瓜。
*** 每一个新增加的新功能的建议，都必须通过严格的论证。这叫做闭环负反馈，就像汽车轮胎的摩擦在驾驶系统中形成闭环负反馈，松开方向盘时，它会回转让汽车恢复直行。
*** 多少功能，哪些功能最合适。
*** 斯德哥尔摩综合症：爱上绑架的人。
*** 湿湿的狗：不常用功能堆积在一起的手机。
*** 目标描述比功能描述更能说明问题。
*** 期望性和需求很容易混淆，但是两者是截然不同的概念。
*** 逻辑人与普通人：
**** 逻辑人
需要控制权——接受以复杂为代价
想理解事物内部机制——可以接受失败的代价
关心可能性——接受以前准备工作为代价
**** 普通人
需要简单——愿意放弃控制权
想成功——可以不去理会事物内部的机制
关心概念——接受偶然的失败为代价
*** 著名代码（开放）宗师Eric Raymond——“好的程序员直到编写什么，伟大的程序员直到重用什么。”
*** 爱因斯坦：“不能用引发问题相同的思维方式去解决问题。”
*** 只为一个人设计（汽车）四不像。
*** 我更侧重可信度，而不是多样性。（人物构造）
*** 精确而不是精准。
*** 不管编程是不是在范例的边缘定义，而设计却是在中心定义的。
*** 平均用户从来不会真正的“平均”。平均角色损害精确角色所具有的优点。
*** 营销角色是基于地理范围和销售渠道，而设计角色纯粹基于用户。
*** 发现一个人的目标很特别的时，我们会分离出一个角色。没有必要让所有的角色目标都不同，但每一个角色必须有所不同。
*** 不知情同意：每一步操作者都必须选择其中一项，而选择的范围和结果却不得而知。
*** 精确反映了软件内部的选择方式，但却没有任何说明和提示信息。——尸体彩绘（内容空洞却花哨的操作界面）
*** 少提供选择，多提供信息
*** 单层分组——将一个层次的信息进行分组。单层分组不需要人们判断类型，而是把类型信息变成帮助提示信息。单层分组将类别作为影片的属性。 ——索尼Trans conn个哦概念四的passport系统
*** 0、1、无穷大的数字陷阱：在程序员看来，30比0和1都大，等同于无穷大，看来因为产生的展示无穷多电影的想法是问题所在，所以他们不得不将它们按类别划分。
*** 人物角色为达到它的目标而存在，目标的存在让人物角色变得意义。
*** 人们会尽权力去达到商业目标，但这仅是在达到他们的个人目标之后。
*** 良好的交互设计的本质是：设计的交互能让使用者在不影响个人目标的情况下，达到他们的实际目标。
*** 平等付出的原则：为得到相应更好的功能和技能，愿意付出更多的时间。
*** 解决问题的第一步是承认它的存在。
*** 目标导向设计：
**** 个人目标：
不觉得自己愚蠢 不出现差错 完成适量工作 有趣（或至少不觉乏味）
**** 企业目标：
增加利润 增加市场份额 击败竞争对手 聘用更多职员 提供更多的产品服务 上市
**** 实际目标：
避免会议 处理客户要求 记录客户订单 创建业务的数字模型

**** 错误目标：
节省内存 减少击键次数 在浏览器中运行 容易学习 保证数据完整性 提高数据录入速度 提供程序执行效率 使用酷的技术 增强美感 在不同的平台保持一致“ 不要将技术与目标混淆起来 ”
*** 任何于人脑足够接近的东西都被当做人来看待，如果任何东西都表现足够的认识摩擦，那种直觉会起作用。
*** 人们的很多认识与他们的经验想左，但是这些认识在我们进化的环境中是正确的，他们追求的目标打破了自己的正常状态，但是非常适应环境。
——>把软件行为设计得接近人的行为
——>软件有礼貌：对我感兴趣；尊重我；主动提供帮助；拥有常识；预知我的需要；反应敏捷；解决自己的问题；提供有用的信息；有洞察力；有自信；很专注；灵活应变；即时回报；让人信任
*** 从用户目标的角度我们有机会抓住创新设计的机会。
*** 把任务合并进来的工具，我们叫做场景。分为日常、必要和边缘三种。
*** 其它几个有用的设计概念：屈折界面、永久的中间用户、词汇、头脑风暴和横向思维。
*** 当技术变化时，任何一般也会变化，但是目标却保持恒定。迫使技术发生变化，突出目标。
*** 词汇表：创建详细而精确的词汇表。
*** 典型的工程开始于对制约条件和边界条件的描述；而我们，假设所有事情可能，然后开始设计，更清晰地看到角色和目标。
*** 受范性反应：图表慢慢变大。
*** 用户界面上的每一项元素对用户来说都是负担，用更少的东西做更多的事情总是好的。
*** 好的服务员到餐桌的次数更少，好的设计师总是设计更少的界面。
*** 交互设计师根据经验，训练和判断进行正确的评价。他们有应用于各种情况的原则，习惯用语、工具，而且他们会通过其他信息把这三者结合起来。
*** 按钮容易发现发现和按下，并不意味着用户容易知道应该按下哪个按钮。这是认知摩擦的问题，不是工业设计的问题。
*** 让交互变得更友好的关键是减少电脑与用户之间的不确定性。
*** 通过迭代可以获得更好的交互性。不能乱用、滥用。
*** 你总是可以推迟短期性思考，但是你永远不能推迟长期性的思考。
*** 交互设计师必须将设计个哦姑娘作做成书面文档。将与用户的交互定义得非常详尽和精确。
*** 总体上理解技术局限，有设计的热情。有技术疾驰和文档的写作经验，从用户角度思考问题。知道程序员在开发过程中的需要和担心，热爱优美的设计，
有能力去设计，将技术知识与表达想法的能力结合起来。
*** 我们勾画问题、列出角色，说明设计必须解决的问题。在后....

43、《交互设计之路》的笔记-第88页

Alan Cooper将程序员称为“ 逻辑人 ”，第7章主要讨论程序员与普通人的基本思维和行为方式的4条不同点。分别是：

1.程序员牺牲简单，来获得控制权；

2.程序员牺牲成功，来换取理解；

3.程序员只关注可能性，不管概率；

4.程序员行为就像体育生。

看到这4条，我不禁有些疑惑。

毕竟，我这个原本可能成为女程序员的人，看到上面4点会不自觉的审视自己。

接下来想看看Alan Cooper是如何来讨论的。

44、《交互设计之路》的笔记-2認知摩擦

開始閱讀

交互設計師首先創建的是概念，然後考慮行為，最後考慮界面。

45、《交互设计之路》的笔记-第140页

进一步地说，最重要的目标是具体个人拥有的个人目标。是真实的人而不是抽象的企业与你的产品打交道，因而必须将人们的个人目标置于企业目标之上。人们会尽全力达到商业目标，但是这仅是在达到他们的个人目标之后。最重要的个人目标是维护个人尊严:不觉得自己愚蠢。良好交互设计的本质是，设计的交互能让使用者在不影响个人目标的情况下，达到他们的实际目标。目标不是任务。目标是终结条件，而任务是达到目标所需要的中间过程。区别任务和目标非常重要，人们很容易将它们相互混淆。有一种简单的方法可以区别任务和目标。任务随着技术的变化而变化，而目标具有让人高兴的稳定属性。例如，从圣路易斯到旧金山旅行，我的目标是快速、舒适、安全。如果是在1850年前往加州的金矿，我很可能乘坐一辆新的大篷马车，为了安全起见，我会随身携带一支温彻斯特来复枪。而在1999年从圣路易斯前往加州硅谷，我会乘坐一架波音777客机，而也是为了安全起见，我会把枪留在家里。我的目标没有变化，但是由于技术的变化，我的任务却发生了截然相反的变化。

46、《交互设计之路》的笔记-第5页

很多

47、《交互设计之路》的笔记-第44页

48、《交互设计之路》的笔记-第124页

在使用角色之前，程序员与负责交互设计的经理之间的对话可能是这样的：
程序员：“ 如果用户想打印这个怎么办？ ”
经理：“ 我觉得我们真的不需要在第一个版本中加入打印功能。 ”
程序员：“ 但是有人可能要打印这个。 ”
经理：“ 嗯，是，不过我们就不能推迟打印功能的加入吗？ ”

使用用户角色初期：
程序员：“ 如果用户想打印这个怎么办？ ”
交互设计师：“ Rosemary对打印这个东西不感兴趣。 ”
程序员：“ 但是有人可能想打印。 ”
交互设计师：“ 可是我们是为Rosemary设计的，而不是某人。 ”

当程序员接收角色之后：
有所进步的程序员：“ Rosemary会想打印这个吗？ ”
高兴的交互设计师：“ 不。不过Jacob需要一个季度打印一次报告。 ”
有所进步的程序员：“ 那好，既然他们很少打印，让我们节省一些时间和精力。让我们购买现成的商业打印许可，不要求编写花哨的打印功能。 ”

高兴的经理：“ 那我们的计划可以缩减两个星期了！ ”

49、《交互设计之路》的笔记-第115页

为快乐而设计，为能力更强而设计，为人而设计

人物角色：
最有力的工具总是在概念上简单，但是在应用的时候需要一些技巧，交互设计工具也是一样。我们最有效的工具很简单：精确的描述用户和用户想要完成的事，技巧在于如何确定和使用这些描述。

只为一个人设计：

目标用户越多，偏移目标的可能性就越大。如果想得到50%的产品满意度，你就不能让一大批人中的50%对产品满意来达到这个目标，只能通过分离出这50%的人，让他们100%满意来做到。可以再进一步，瞄准10%的市场，让他们成为产品的狂热追随者，你能获得更大的成功。这似乎有悖于常理，但是只为一个设计是让大众满意的最有效的方法。

开心的用户是非常有效而又价值的资产。通过将目标聚焦在一小群用户，你可以在目标市场中建立狂热的用户忠诚度。

给角色命名是成功定义一个角色的关键。没有名字的角色毫无用处。没有名字，一个角色不会在人们心目中成为一个具体的人。

《交互设计之路》

版权说明