《反模式》

图书基本信息

书名：《反模式》

13位ISBN编号：9787115162793

10位ISBN编号：7115162794

出版时间：2008-1

出版社：人民邮电

作者：布朗

页数：216

译者：宋锐

《反模式》

内容概要

《反模式危机中软件架构和项目的重构》从一个新的角度审视模式，提出了反模式的概念，介绍了在软件开发中常常出现的问题。将设计模式错误应用于不适当的上下文环境。首先，定义了软件开发参考模型和文档模板来说明这些反模式。然后，从开发人员角度、架构角度和管理角度三个方面对这些反模式逐一说明，并说明了与特定反模式相关的背景、原因、症状和后果，让读者可以迅速地检验身边的项目是否出现了这些状况，同时也针对每个反模式给出了相应的解决方案。

《反模式》

作者简介

William J.Brown曾任Saga软件公司研发总监和OMG金融业工作组主席。擅长金融行业大型软件系统的开发。

《反模式》

书籍目录

《反模式》

编辑推荐

《反模式危机中软件架构和项目的重构》适用于从事项目管理和软件开发的相关人员。

《反模式》

精彩短评

1、好书！失败的案例，我们学到更多。但是无可辩驳的是，我们还在不断的重复书中的那些失败的模式。
2、苦涩

3、从第4章开始，读得冷汗直下...
4、把简单的东西弄得晦涩难懂
5、不必读
6、此书竟然绝版，不可思议
7、归纳了常见问题，但是对于问题的解决方案过于抽象，可操作性不高。
8、主要是经典问题。翻译有一些错误。
9、对项目管理，开发过程中提出了很多反模式，以后可别做反模式哦
10、浙江图书馆
11、216页的书,只有180多页有用的...
12、:
TP311.52/4237-2
13、摔完跟头你再回来看，会后大悔
14、从编码设计，架构，管理三个层次来讲述项目失败的各种反“模式”，并给出解决方案或重构方案，个人认为架构与管理两个层次讲解得出色一些。
15、没有什么太大的感觉，可能是因为我道行还是太浅吧
16、还是很多没有深切的体会到

改天再读
17、精通设计模式的人，这些应该都会想到！当然，买了这本书就直接学会了，适合我这种懒人！书的确是贵！
18、排除能力问题，反模式有如下原因：
1、由于开发人员追求短期进度，降低了软件的可修改性。
2、开发人员忽略了软件的非功能性需求，导致产品质量低下。
3、利益相关者之间失败的沟通和协作，使得管理过程滞后于开发过程。
19、想再读一遍
20、从项目管理，到代码上设计出现的一些不好的模式
21、未给满分实在是由于原书年代过于久远了，书中个别模式已经过时。著书时间正是面向对象大力受到推崇的年代，而如今似乎是函数式编程又开始重新受到青睐了。真是十年河东，十年河西。
22、e文太密
23、本来是满心期待的。这本书也快绝版了，在亚马逊第三方“高价”买的。很失望。一个是空，第一部分典型的中国教科书风格。一个是老，软件开发性反模式很大部分聚焦于非面向对象程序员使用面向对象技术产生的问题。还有就是阅读体验非常差，这部分归咎于翻译的原因。比如可以把Architecture By Implication翻译成“实现主导架构”。最大的价值在于提出了这些模式术语，仅此而已。
24、给五星，后面部分还不太懂。。
25、对corba的溢美之词颇多，对于面向对象带有宗教般的感情，带着批判的态度来看吧，总的来讲，还是一本不错的书
26、1995年，GoF（以Erich Gamma为首的“四人帮”）所著的《设计模式——可复用面向对象软件的基础》（Design Patterns--Elements of Reusable Object Oriented... 阅读更多
27、失望，读完一遍没什么收获，白开水的感觉，也许是我还没到那个水平吧
28、给我一个感觉,对于我们这种没有实际生产经验的人,真的是没什么大的作用,留着以后看
29、印象比较深刻的反模式：
手把手教你如何制造一颗原子弹（从头做起只需7天【重新发明车轮模式】）；

一个超级大类（混球模式）；
这些10年都没动过的代码天知道会有什么表现，保留吧（岩浆流模式）；
我们有一个计划，这个计划能拯救我们（一切都会按计划模式）；
……
30、自作聪明后大吃苦头，需要好好看看这本书
31、少了点儿，开头废话多，细节少
32、软件工程经典，值得收藏！

《反模式》

精彩书评

1、无论是普通开发人员，还是各级技术管理人员，都值得一读，并以此总结反省自己。唯一的缺点是那些模式的名字取得太晦涩。
2、怎么说呢？书绝对是好书，可惜的是这真的不适合当作自学或教科书，无愧于它的生产效能大奖阿。如果你没有项目的经验，那么这本书真的不适合你，不要再浪费时间了。每一个反模式，作者都写的简短而有力，如果你有过实际的经验，我相信你读过之后都会有一拍脑袋，醍醐灌顶，恍然大悟的那种感觉。作者深厚的实践功底为这本书灌注了活力，尤其当你还在项目中时，仿佛你正对话于一个经验丰富的架构师，从中的获得可想而知。
3、《反模式》这本书终于出版了 1995年，GoF（以Erich Gamma为首的“ 四人帮 ”）所著的《设计模式——可复用面向对象软件的基础》（Design Patterns--Elements of Reusable Object Oriented Software）一书出版了。这本书在面向对象编程/设计领域具有划时代的意义。不夸张地说，这本书就是面向对象程序员的红宝书，对于面向对象程序员来说，应该人手一册。没有读过这本书，对于面向对象的理解还停留在很低的层次；读过了这本书，才算真正理解了一些面向对象的精华。很自然，这本书一时间洛阳纸贵，后来还陆续出现了很多以不同语言的实例来解释GoF设计模式的著作，例如《Java设计模式》、《C#设计模式》等等。但是物极必反，任何事情都不能走极端，否则就会造成不良后果。在项目开发不适当的时机使用设计模式，或者在不适当的场合使用设计模式，都会给项目开发带来很多不必要的复杂性，从而增加大量的沟通成本，甚至会严重影响开发效率。以前听说过一个笑话：一个程序员发邮件给Erich Gamma，说在他们最近的一个项目中，他尽力使用了23种GoF设计模式中的21种，还有两种怎么也找不到适用的场合。他感到很苦恼，希望能够得到 Erich的帮助。这个笑话说的当然是一种最极端的情况。尽管如此，大量的实践证明，不当使用（不适当的时机、不适当的场合）设计模式会给项目开发带来很大的麻烦。自己挖坑把自己埋进去，这是很多一知半解的面向对象架构师经常做的事情。言必称架构、言必称模式正是中毒很深的迹象。1998 年出版的《反模式——危机中软件、架构和项目的重构》（AntiPatterns——Refactoring Software, Architectures, and Projects in Crisis）这本书对于滥用设计模式的趋势来说可谓是一剂很及时的解毒良药。反模式其实也是一种模式（正如设计模式是一种模式一样），不过它主要考察的是这种模式所带来的不良后果。这本书系统地总结了很多种类的反模式，以及如何识别出这些反模式并且加以避免。这些反模式涵盖了软件开发、软件架构和项目管理等方面。虽然这本书并不是专门针对设计模式展开讨论的，但是不当使用设计模式很多时候都会导致出现反模式。平衡感是优秀的面向对象架构师应该拥有的素质，《反模式》这本书可以使架构师拥有更好的平衡感。1999年出版的Martin Fowler所著的《重构——改善既有代码的设计》（Refactoring: Improving the Design of Existing Code）这本书对于如何改善现有代码的设计提出了非常具体的解决办法。Martin Fowler在这本书中认为，不应该在最初编码时就使用设计模式，而应该将设计模式作为重构的目标。实际上，设计模式正是Erich等人在对一些开发框架进行重构的过程中总结出来的。后来2004年出版的Joshua Kerievsky所著的《重构与模式》（Refactoring to Patterns）在《重构》的基础上更加详细地探讨了重构与设计模式之间的关系，以及如何在重构的过程中逐渐引入设计模式。2000年出版的Kent Beck所著的《解析极限编程——拥抱变化》（Extreme Programming explained--Embrace change）响亮地提出了简单设计、不为明天而设计的口号，进一步为滥用设计模式的趋势消了毒。我并不将这些书看作是彼此独立的著作，而是把它们看作一个系列，因为它们的内容有很大的关联性。这几本书代表了一个伟大的时代。在这几本书中，国内最晚出版的就是《反模式》，现在它也终于出版了，补上了最后一块拼版。对于国内年轻的面向对象程序员来说，真是一种福气。虽然距离它的英文版出版已经过去了将近10年，现在读起来，其中的内容读起来仍然感觉非常亲切，书中所批判的那些问题在我们周围仍然经常发生。举个例子，尽管《反模式》这本书在1998年就已经指出了委员会设计的严重问题，但是2001年仍然出现了EJB 2这样典型的委员会设计的产品。EJB 2给JavaEE社区带来的危害一直到2005年之后才逐渐消除。为何那些早已指出的问题还会一再重复出现？很显然，这本书说出了一些在软件开发中存在的本质性问题。这是一本非常少见的读第二遍仍然感觉有新意，其价值不会随时间而消退，历久弥新的技术著作。书中的内容今后10年之内都不会过时，这是我的保守估计。
4、书确实是好书，能开阔眼界。翻译的也还不错，不过由于高级技术人员和普通程序员关注的焦点和领域不同，导致书中谈到的东西和某些术语显得有些陌生，可能这也是一部分读者抱怨翻译的不够好的原因吧。书买了一段时间了，一直没有看，其实我挺喜欢这种200多页的书的，可以速度读完有所

收获。可惜的是，当我真的决定花时间来看的时候，由于自己的经验有限，共鸣的不多。这本书其实主要是写给有一定项目管理经验的技术人员的，主要介绍了开发，架构和管理方面的常见反模式和相应的解决方案。建议先阅读过重构一书以后再来看这本。看得吃力的我决定先囫囵吞枣了，然后将其放到已阅的类目中去，等我经验上了新的台阶后再来读，一定会更有收获。虽然物有所值，但相对页数来说，价格是有点贵。

5、看了下，有点看不下去不知道是不是翻译和文化的原因，本应该是比较有趣和顺畅的，中译本读起来有点晦涩。书的前三分之一都再说一些概念，无非就是“ 反模式 ”是开发中糟糕的模式，以及出现反模式的原因，也无非就是懒惰，无知，贪婪，匆忙... 然后就讲了一系列的反模式，其实就是对面向过程开发人员转到面向对象后可能会出现的一些列问题，但是他的每个模式讲的极其简单。其实说到底就是缺少面向对象的思想。

6、本来是满心期待的。这本书也快绝版了，在亚马逊第三方“ 高价 ”买的。很失望。一个是空，第一部分典型的中国教科书风格。一个是老，软件开发性反模式很大部分聚焦于非面向对象程序员使用面向对象技术产生的问题。还有就是阅读体验非常差，这部分归咎于翻译的原因。比如可以把Architecture By Implication翻译成“ 实现主导架构 ”。最大的价值在于提出了这些模式术语，仅此而已。

《反模式》

章节试读

1、《反模式》的笔记-第1页

开发性反模式

The Blob 胖球：一个类垄断了处理过程。
Lava Flow 岩浆流：代码中存在历史原因造成的没有功能、无法理解的代码。
Functional Decomposition 功能分解： 非面向对象开发人员使用面向对象语言来设计／实现一个应用时常出现
Poltergeist 恶作剧鬼：短暂出现的类，用于启动其他更为持久的类中的某些操作。
Golden Hammer 金锤：团队对特定解决方案或特定供应商的产品具备了很高的能力，没够新产品或开发都用它来解决。解决：定义边界来开发软件系统，开发人员要跟上技术发展、
Spaghetti Code 面条代码：程序或系统几乎没有软件结构。
Cut-And-Paste Programming 剪贴编程。

架构性反模式

Stovepipe Enterprise 烟囱企业：企业的多个系统独立设计、缺乏共同性，阻碍了复用。“ 自动化孤岛 ”。重构：在多个层次上协调，定义参考模型。
Stovepipe System 烟囱系统：单系统中缺乏通用的子系统抽象，子系统被使用多种集成策略和机制随意组合在一起。重构：定义一个共同的服务接口（构建架构）。
Vendor Lock-In 供应商锁定：重构：隔离层。
Architecture By Implication 实现主导架构：系统缺乏架构规范：语言和库的使用，编码标准，内存管理等。重构：有组织的方式进行系统定义。多个视图。4+1 Model View，逻辑视图，用例视图，过程视图，实现视图，开发视图。
Design By Commitee 委员会设计：太多人参加设计导致概念不清。重构：改革会议过程，1）提出问题 2）安静地写下答复 3）扔纸团 4）随机读出答复 5）达成共识 6）去除重复 7）确定优先级 8）讨论。反例：SQL、CORBA
Reinvent the Wheel 重新发明轮子：从头开始建立定制的软件。绿地系统假设。重构：从先前存在的架构中提取有价值的信息，架构挖掘：1）对各种代表性技术进行建模，以便产生相关软件接口的规范。2）对设计进行贵拉来建立通用接口规范。3）对设计进行精炼。

管理性反模式

Analysis Paralysis 分析瘫痪：分析阶段追求完美和完备。重构：增量式开发：内部（基础结构）外部（功能）。
Death By Planning 规划致死：过规划。在规划、制定进度表和捕获进度方面缺乏注重时效的方法。重构：项目计划应该显示出主要的交付品。完成度应该是总体粗略的度量而不是精细的度量。
Corncob 玉米棒子：难以相处的人。重构：战术（转移责任给反对者，隔离问题（采用大多数人意见），质疑问题（要求Corncob澄清观点，证明论据））、战役（离线地。矫正性会谈，介绍新职）、战略（Croncob支持组，把corncob放到一组，空部门，开除）。
Irrational Management 非理性管理：管理者的决策不是深思熟虑的战略，而是膝跳反应。项目拷问，对某个话题进行争论。重构：1）承认问题，寻求帮助。2）理解开发部署（了解部署的技术能力和个性特点）3）提供清晰的短期目标。4）团队拥有共同目标。5）对过程进行改进。6）推动交流。7）管理交流机制。8）异常管理。9）采用有效的决策制度方法：情况分析，决策分析。
Project Mismanagement 项目管理不善：关键活动被忽视或减少：不充分的架构、代码审查不足，测试覆盖不充分。重构：限制风险：管理风险、常见项目失败点、质量风险；建立共同理解。在架构层次定义模块依赖。在设计层次，定义跨越模块的控制用例。

2、《反模式》的笔记-第12页

好书

3、《反模式》的笔记-第1页

翻译不好

《反模式》

版权说明