# 《面向模式的软件体系结构卷3》

## 图书基本信息

书名：《面向模式的软件体系结构卷3》

13位ISBN编号：9787111169833

10位ISBN编号：7111169832

出版时间：2005-9

出版社：机械工业出版社

作者：克车尔

页数：169

# 《面向模式的软件体系结构卷3》

## 内容概要

在任何类型的软件中，有效管理资源都是至关重要的。从移动设备中的嵌入式软件，到大型企业服务器上的软件，有效地管理内存、线程、文件、网络连接之类的资源对于让系统可以正常且高效地工作都很重要。

我们经常在软件开发生命周期的后期才发现资源管理需求，而在这么晚的时候改变系统设计很困难。所以，在生命周期的早期执行这样的任务就很重要。因为属于不同领域的系统有不同的约束和需求，所以对某个特定系统或者配置很有效的方法对另一个系统就未必那么有效。

本书用模式来描述在系统中有效实现资源管理的方法。这些模式描述得很详细，使用了几个例子，并且和POSA前两卷一样，给出了如何实现它们的指导。此外，这一卷还对资源管理做了透彻的介绍，并给出了两个案例研究，分别把这些模式应用于自组网络计算和移动射频网络。这些模式归于不同的资源管理领域，涉及了完整的资源生命周期：获取、管理和释放。

本书适合软件构架师、设计者和开发者阅读，也可供计算机专业的学生参考。

# 《面向模式的软件体系结构卷3》

## 作者简介

Michael和Prashant同POSA前两卷的作者紧密合作并且积极参与模式社区有好几年了。他们在各自的公司（西门子和IBM）中都在参与新技术和软件构架的研究与咨询。

# 《面向模式的软件体系结构卷3》

## 书籍目录

# 《面向模式的软件体系结构卷3》

## 精彩短评

1、书很薄，但含金量很重。
2、读POSA一二卷原文电子版两遍了，总结的模式架构为大多人认可。象Reactor,Proactor，ACE里用，Twisted里也实现了，模式掌握了在开发中就多了些设计参考和维度。一二卷公司有中文版，翻译得。。。不好，这不是我一个人的感觉。第三卷也在看电子版，在当当看了这本书的翻译作者，就决定买下了。翻译得舒畅，可以不看电子版了。
3、网上很多说这套书的前两卷翻译超烂，吓得我没敢买，只买了这本便宜的瞻仰瞻仰，感觉还行。强烈建议引进原版，这可是继《DesignPatterns》之后被大多数认可的经典书啊！！！
4、recommand
5、作者的功力挺深...我也喜欢译者...总的来说很不错..尤其是作者画的UML图..很是详细,读起来很过隐...
6、翻译得很棒，书的内容也简单明了
7、还不错，发货快
8、多写代码都懂得，没必要看，多用用boost这类库
9、前两本已经看完了，翻译的一般，看着也费劲，时不时还要翻英文版的参考一下，不过还是很有收获的。最开始的时候尝试看英文版，费劲啊，还是需要多练，加油
10、从图书馆借来读的，资源管理架构，这本很薄。
11、POSA3讲述系统资源管理，是POSA系列中翻译最好的一本，第一次采用java作为主要的演示语言，中间件编程必看

# 《面向模式的软件体系结构卷3》

## 章节试读

1、《面向模式的软件体系结构卷3》的笔记-第2页

第一章，其实很多都没有什么他大的用，讲了模式和一些convention。比较有用的就是在考虑资源处理时候的几个方面：
1.性能： response time, throughput
2.scalability: 管理resource以及他们的lifecycle，其实跟我们平常做的差不多，主要考虑scenario，有什么样的case就有什么样的设计。
3.predicability，主要针对实时系统，如果设计的好，对于web server的throughput也需要这方面的考虑。
4.flexibility，灵活可以说是什么都可以，易于配置，管理，monitor，扩展。
5.稳定性
6.一致性，主要是从一个一致性的状态 --> 另一个一致性的状态。感觉这一点不太好搞，难道我们要persist state，transaction？

2、《面向模式的软件体系结构卷3》的笔记-第11页

第二章，
这一章主要是如何获得资源，讲了4个模式
lookup, 参照corba的naming service, trading service以及jndi, ldap。主要目的就是将user与resource provider解耦合。可以通过返回给用户一个proxy来实现一定的灵活性。比较有意思的是讲了点bootstrap，也就是resource provider与resource之间如何联系，可以通过广播，组播之类的机制。如果仔细考虑，不如去看p2p的发现机制。trading service是有描述的，感觉就想query language。最后讲了Federated Lookup其实就是加了一定策略的composite lookuper。 Replicated Lookup：主要实现了HA的cluster，可以在几个point上做cluster：1.proxy中，2. resource provider, 3. resource.其中对于java里边的有几个project适合我们仔细的看看： jini, JXTA，还有DNS的机制。
---
Lazy acquisition
延迟获取我们用的比较多，将资源推迟到用的时候才获取，主要通过proxy来实现的。可以提高startup time，但是可能对第一次使用的时候，速度比较慢。
---
Eager acquisition
实际上主要用于获取资源池的资源，防止抖动，比如说memory，毕竟heap memory分配需要lock和整理的。如果采用资源池，那么资源就是有状态的，需要管理资源的lifecycle。这里边提的比较好的一点是，在启动获取资源的时候，最好以hook的方式，好处是灵活性。还有一个是运行时获取，这个估计实用性不大，也就是靠adaptive。
---
Partial acquisition
感觉这个是比较使用的模式，也是比较复杂的。刚开始获取一部分，然后再on demand或者说根据策略进行获取，让我想起了work manager，如果这个策略实现的比较好，可以真正的实现adaptive，这就是我们所期望的。

实现的主要思想就是分步进行，把大象装进冰箱需要几步:)。实际上这也是一个混杂模式，混合了Eager(刚开始上来就获取必要的一部分）和Lazy(如果eager获得的顶不住了，只要再来点）。他可以考虑异步的资源获取，比如说你可以schedule一个后台的job去获取资源，当user真正要用的时候可能就已经获得资源了，这样和reactor模式结合（感觉reactor真别扭，要说event manager比较好）。这里的错误处理是需要考虑的，实际上我们的错误处理一般来说都不需要“处理”，都是fail through类型的，如果真的要处理就需要state machine了。如果分布，每一步骤还是可配置的，那不就成了SEDA了？

最后举了协议的例子，有点意思。比如说IIOP，先读协议头，然后知道需要读多少数据，看来IIOP也类似于HTTP，都有content-length的概念。是不是协议的设计都需要类似的东西呢？

对于这一点的启发，我发现我们的cluster可以说没有协议的设计，最主要的是没有meta data，也就是header，所以导致了升级太tmd困难了。

3、《面向模式的软件体系结构卷3》的笔记-第1页

序言
资源的获取，管理，释放对长期运行的程序都有很大的影响。尤其是对其性能，规模，可伸缩性，可延续性。比如从堆上分配内存，对于web server或者application server而言，在请求处理代码路径中每一个从堆上分配内存的操作，都降低了服务器的性能和可伸缩性。我觉得这里很有感触，比如说java剧烈的gc，在performance audit的时候，我们都希望最后的throughput到达的时候，gc主要发生在新生代。原文说调用堆管理器分配内存和释放内存都需要lock，但是可能不是全部的lock，这是对于java来说的吧。所以在进入请求代码之前就尽可能多的分配好资源，比如说pool，或者利用cache。可以使用Partial Acquisition.

# 《面向模式的软件体系结构卷3》

## 版权说明