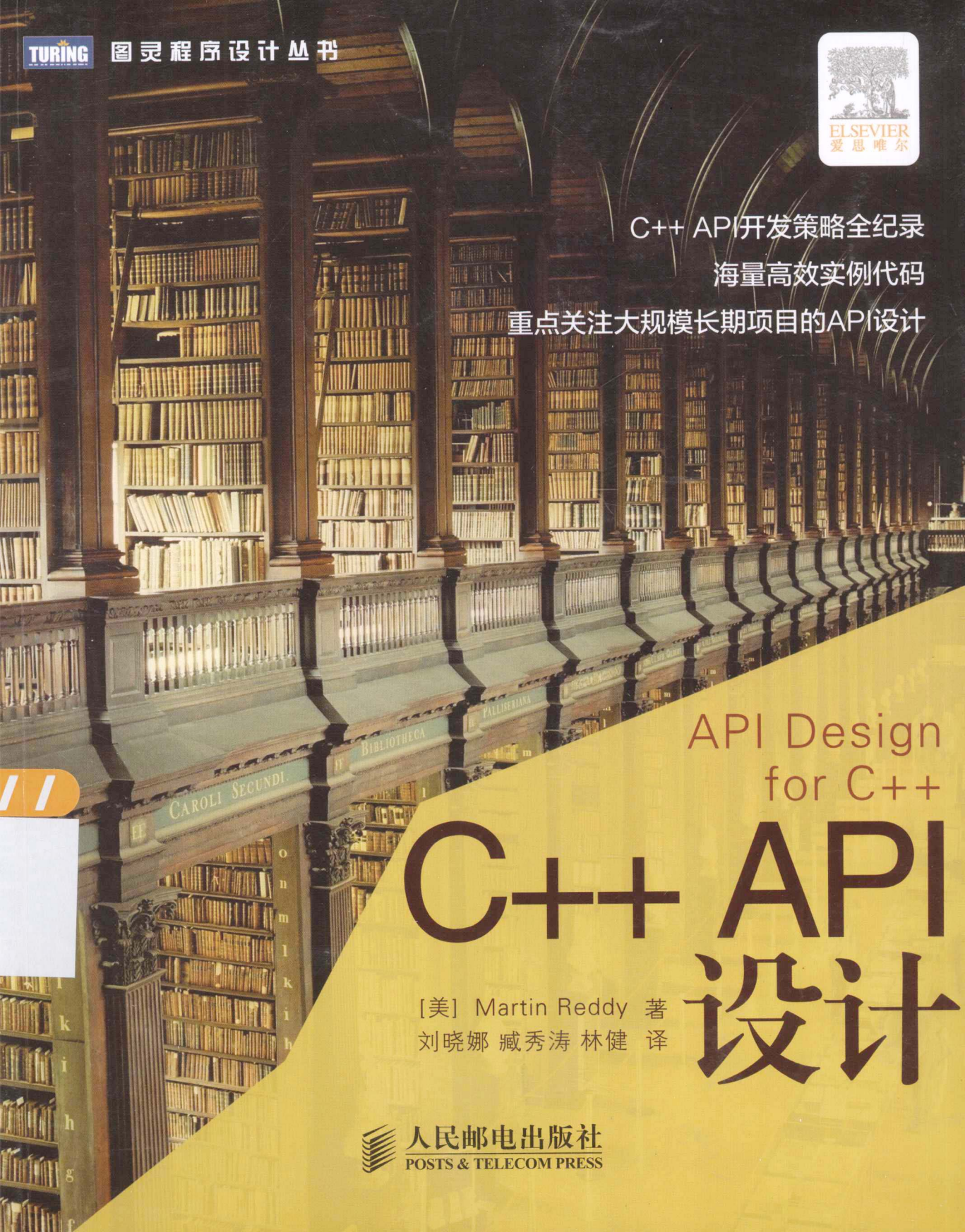
TURING
图灵程序设计丛书
ELSEVIER
爱思唯尔
C++ API开发策略全纪录
海量高效实例代码
重点关注大规模长期项目的API设计
API Design for C++
C++ API设计
[美] Martin Reddy 著
刘晓娜 臧秀涛 林健 译
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

API Design for C++

C++ API 设计

[美] Martin Reddy 著
刘晓娜 臧秀涛 林健 译

人民邮电出版社
北京

图书在版编目（CIP）数据

C++ API设计 /（美）雷迪（Reddy,M.）著 ；刘晓娜，臧秀涛，林健译. -- 北京 ：人民邮电出版社，2013.9
（图灵程序设计丛书）
书名原文：API Design for C++
ISBN 978-7-115-32299-9

Ⅰ. ①C… Ⅱ. ①雷… ②刘… ③臧… ④林… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第138742号

内 容 提 要

如何构建高效、健壮、稳定且可扩展的优质API？对于这一软件工程上的难题，Martin Reddy凭借长期的从业经验，对优质API所应具备的各要素进行了全面分析，针对API的不同风格及模式，以及大型长期项目的内在需求，给出了种种最佳设计策略，从而对API设计过程的规范性及可持续性作出了理论上不可磨灭的贡献。

本书适合具有一定C++编程经验的程序员阅读，也适合对API设计主题感兴趣的读者参考。

◆ 著　[美] Martin Reddy
译　刘晓娜　臧秀涛　林　健
责任编辑　丁晓昀
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷

◆ 开本：800×1000　1/16
印张：23.75
字数：560千字　2013年9月第1版
印数：1－4 000册　2013年9月北京第1次印刷
著作权合同登记号　图字：01-2011-3964号

定价：89.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第0021号

版权声明

译 者 序

随着 C++11 新标准的发布，很多人又对这门争议颇多的语言重新燃起了兴趣。当然，争议仍在继续：支持者认为新标准引入了很多实用特性，简化了开发；而反对者认为新标准加入的诸多新特性，进一步加深了语言的复杂性。放眼国内外的技术社区、论坛和微博，此类争论不一而足。

之所以有这么多争论，主要原因还是 C++的复杂性高，“坑”多。即便如此，C++的应用仍十分广泛，如大型分布式系统、服务器开发和桌面应用等各种领域。Bjarne Stroustrup 都曾戏言，“只有两种编程语言：一种是天天挨骂的，另一种是没人用的”。

正是因为这种现状，关于 C++语言本身和相关最佳实践的书非常之多，很多都是名家名作。典型的如 Bjarne Stroustrup 的 C++程序设计语言，皇皇巨著，钜细靡遗；Stanley B. Lippman 等人的《C++ Primer 中文版》，事无巨细，循循善诱；Scott Meyers 的《Effective C++中文版》则是条分缕析，直指精要。篇幅所限，不能一一列举。

虽然已经有了这么多的名著，《C++ API 设计》一书的作者 Martin Reddy 却是另辟蹊径，从 API 设计的角度向读者演示了最佳的 C++实践。而且本书所及，已经远远超出了标题所指的 C++ API 设计：良好的 API 所应该具备的特性其实是语言无关的，常见的设计模式也可用于不同编程语言；此外，性能、文档、版本控制以及脚本绑定等专题也鲜有类似 C++书籍会拿出较多篇幅专门探讨。

本书作者作为爱丁堡大学博士和 IEEE 高级会员，其研究经历使其讲述问题十分严谨，逻辑性极强；而其在世界顶级动画工作室 Pixar 的软件工程经历使其给出的每个实例都紧密联系实际。本书内容涵盖了大型软件开发设计的各个方面，所给出的建议都来自作者在顶级动画工作室的真实软件工程实战经历。不管是初学者，还是有经验的开发人员，细读此书，我们认为都能有很大的收获。

本书覆盖面广，而且旁征博引，翻译实非轻松。为了保证翻译质量，我们一再向出版社申请推迟交稿时间，翻译过程持续长达八个月之久，一路走来，感觉真的很累。但译罢我们也收获颇多。然水平所限，译文中有不当之处，肯定读者批评指正。

最后，我们要感谢计算所杨帆同学，他认真地审校了稿件，保证了本书的翻译质量。感谢计算所学生朱伯龙，他给我们提供了很多好的修订意见。还要特别感谢图灵编辑李鑫老师、丁晓昀老师和傅志红老师，是他们的严格要求以及理解和支持，才让本书能够以较高质量面世。

刘晓娜　臧秀涛　林健
于中国科学院计算技术研究所
2012.10.30

序　言

首先需要指出，我并不认为自己算得上世界级的 API 设计师或软件工程师。然而，在计算机图形学和几何建模领域，我确实算得上一位资深研究者。由于工作上的机缘，我在 Pixar 动画工作室结识了 Martin。

作为计算机图形学研究人员，我要经常撰写包含高深数学理论的论文。曾几何时，我也在著名大学学习如何做一名计算机科学家，并且编写过许多重要代码。正是这样的背景，使我有机会领导一群软件工程师共同为 Pixar 打造新一代的动画软件。我曾天真地认为这项工作不会像做研究那么难，毕竟就定义而言，研究是对未知事物的创造，而工程是对已经熟识的课题加以实现。然而，我大错特错了。

随着工作的深入，我逐渐意识到，软件工程的确是我至今遇到过的最大挑战，所以几年之后，我决定放弃这份工作，重新回到对计算机图形学的研究之中。

我从《C++ API 设计》这本书中获益匪浅。许多经验教训都是历经痛苦的磨砺才总结出来的，而它们都囊括于 Martin 的这本见解深刻而又简单实用的书中。Martin 并不是站在那些学院派的软件研究者的角度来探讨这些问题（尽管他确实引用了大量这些人的成果和见解），而是从一位一线软件工程师和管理者的视角来阐述。他深刻体会到优秀软件设计的重要性，并且善于阐明这些软件设计的优点。书中提供了很多实现优秀软件设计的有效方法。

特别值得赞赏的是，Martin 不仅关注 API 设计，而且特别注重软件的生命周期，故而书中还囊括了诸如版本更新、向后兼容策略以及分支方法等话题。

总之，本书对那些从事软件编写或者软件管理的人来说有着巨大的价值，它堪称一部包含了种种久经考量的最佳软件实践的全集式著作。

Tony DeRose

Pixar 动画工作室高级研究员和研究小组负责人

前　言

编写大型C++应用既复杂又棘手，设计健壮、稳定、易用且耐用的可重用C++接口更是难上加难。在这些努力中获得成功的最佳途径就是遵循优秀的应用编程接口（Application Programming Interface，API）设计原则。

API为软件组件提供逻辑接口，同时又隐藏实现该组件所需要的内部细节。它提供模块的高层次抽象，并通过多个应用共享相同的功能，以促成代码复用。

现代软件开发高度依赖于API，从底层应用程序框架到数据格式API，以及图形用户界面（GUI）框架。事实上，常见的软件工程术语，诸如模块化开发、代码复用、组件化、动态链接库（DLL）、软件框架、分布式计算以及面向服务的架构（SOA），都隐含了对高超的API设计技能的需求。

你也许了解一些流行的C和C++ API，比如标准模板库（STL）、Boost、微软Windows API（Win32）、微软基础类库（MFC）、libtiff、libpng、zlib、libxml++、OpenGL、MySQL++、奇趣公司的Qt、wxWidget、GTK+、KDE、SkypeKit、POSIX pthreads、英特尔的线程构建模块（Threading Building Blocks）、Netscape插件API以及Apache模块API。除此之外，Google的许多开源项目也使用C++，在sourceforge.net、bitbucket.org和freshmeat.net网站上也有很多开源的C++代码。

在软件开发中，从桌面程序到移动计算和嵌入式系统，再到Web开发，都能见到上述API的踪影。例如，Mozilla Firefox网页浏览器就是在80多个动态库的基础上构建的，其中每一个动态库都提供了若干个API的实现。

因此，优雅而健壮的API设计对现代软件开发至关重要。它区别于标准应用程序开发的一个重要特征，就是对变更管理的巨大需求。众所周知，在软件开发中，变化是一个不可避免的因素。新的需求、功能要求以及错误修复，这些要求导致软件从最初的设计以来要经历一系列从未预料的演变。然而，若API由数百个最终用户应用程序共享，那么变化就将引发巨大的动荡，并最终可能致使客户抛弃该API。因此，良好的API设计的首要目标是：在为客户提供所需功能的前提下，使新发布的API对客户代码造成的影响最小，理想状况下应该是零影响。

本书的价值

如果有其他软件工程人员需要依赖你所编写的C++代码，那么你就是API的设计者，本书就是为你而编写的。

接口是你编写的最重要的代码，因为如果接口出现问题，那么修复它的代价比修复实现部分的代价要高很多。举例来说，接口的变化要求对所有依赖此接口代码的应用进行更新，相比之下，当客户端应用采用新版本的API时，仅略微更改的实现则可以被透明地、毫不费力地集成。从经济的角度看，

设计拙劣的接口会极大缩短代码的生存期。因此，写出高质量的接口是一项必备的工程技能，也是本书的焦点所在。

Michi Henning 曾指出，如今 API 设计比起 20 年前显得更为重要。因为近年来有更多的 API 被设计出来，这些 API 提供更为丰富和复杂的功能，并且被更多的最终用户应用所共享（Henning, 2009）。尽管如此，当前市面上着重介绍 C++ API 设计的书籍却不多。

需要指出的是，本书并不是一般的 C++编程指南——市面上这类优秀书籍已经有很多本了。本书当然会涵盖许多面向对象的设计材料并提供很多有用的 C++技巧和窍门，但更侧重清晰地描述 C++中模块接口的技巧。因此，我不会深入讨论如何实现接口背后代码的问题，诸如具体的算法选择，或是应用于函数体花括号内代码的最佳实践。

然而，本书将涵盖 API 开发的整个领域，从最初设计到实现、测试、文档编写、发布、版本控制、维护以及最终弃用。我还会讨论一些特殊的 API 主题，如创建脚本化 API 和插件式 API。虽然总体上许多主题也和软件开发相关，但这里着重强调 API 设计。举例来说，当讨论到测试策略时，我将重点介绍自动化的 API 测试技术，而不会试图囊括最终用户应用的测试技术，如 GUI 测试、系统测试或人工测试。

我曾领导过若干由多个合作机构共享的研究性代码的 API 开发，还有奥斯卡获奖影片所使用的内部动画系统 API，以及全球上百万用户所使用的开源客户端、服务器 API 等。本书的内容正是基于以上这些实际经验的，正是这些实践让我明白，高质量 API 设计是成功的关键所在。因此，本书旨在提供工业强度的 API 设计技术与策略的实践精华，它们均源自一系列实践经验。

本书读者对象

虽然本书不是 C++初学者的指南，我还是竭尽全力使文字更通俗易懂，并且清楚地解释所有的术语和行话。因此，无论你是想提高设计技能掌握 C++基础的新手程序员，还是想获得新的专业技能以完善已有才能的高级工程师和软件架构师，本书都具备一定的参考价值。

在写作本书时，我拟定了 3 类特定的读者群体。

- **在职软件工程师和架构师**。正在从事特定 API 项目的初级和高级开发人员，需要一些如何构建优雅且持久设计的实用建议。
- **技术经理**。负责开发 API 产品的开发经理，以及想深入理解 API 设计的开发过程和技术问题的产品经理。
- **学生和教学人员**。计算机科学和软件工程专业的学生，正在学习如何编程，或者正在寻找总结了全面、大型项目实践经验的软件设计资料的学生。

为什么选用C++来描述API设计

虽然有很多通用 API 设计方法学（可适用于任何编程语言或编程环境）可以讲，但最终都需要使用一门特定的编程语言来表述。因此了解特定语言的特征以促进规范的 API 设计是非常必要的。所以，本书专门使用一种语言（C++）描述 API 设计的问题，而非分散内容使其适用于所有语言。然而，想要使用其他语言（如 Java 或 C#）开发 API 的读者仍然可以从本书中获得许多通用的深刻见解。本书

的直接目标读者是编写并维护 API 的 C++工程师，他们的 API 要供给其他工程师使用。

目前，C++仍是大型软件项目中使用最广泛的编程语言之一，并且日渐成为注重代码性能项目的首选语言，因此，你可以在自己应用中选用的 C 和 C++的 API 种类非常多（前面我已经列出一些）。本书重点关注如何使用 C++编写优秀的 API，并引入了丰富的源代码示例以更好地阐述这些概念。也就是说，本书会涉及一些 C++特有的主题，例如模板、封装、继承、命名空间、操作符、const 正确性（const correctness）、内存管理、STL 的使用、Pimpl 惯用法，等等。

另外，在本书出版期间，C++也正经历着巨大的变革。新版的 C++规范处于 ISO/IEC 的标准化进程中。目前，多数 C++编译器遵循 1998 年首次发布的标准，即 C++98。随后的标准于 2003 年发布，修正了前版的几处缺陷。自那时以来，标准委员会一直致力于制定一个重大的新版本规范。在标准被正式批准生效并确定发布日期之前，该版本一直被非正式地称为 C++0x。当你读到本书时，新的标准可能已经发布了。但是，在我编写本书的期间，它仍然被称为 C++0x。

尽管如此，C++0x 已经达到标准化进程的高级阶段，我们可以满怀信心地预言一些新的特性。事实上，一些主流的 C++编译器已经开始实现许多建议的新特性。在 API 设计方面，某些新特性可以用来构建更加优雅和健壮的接口。因此，我一直努力在整本书中强调和解释 C++0x 中的 API 设计。所以，本书在未来几年中应该依然具有参考价值。

术语及排版规约

传统上，我们使用术语“用户”（user）表示使用软件应用的人，比如 Microsoft Word 或者 Mozilla Firefox 的用户。而在本书所指的 API 设计背景下，此术语表示借助 API 创建应用的软件开发人员。换言之，本书通常提到的是 API 用户并不代表应用程序用户。与此同时，术语“客户”（client）在这一点上也具有相同的语义。注意术语“客户”除了指代使用 API 的人之外，还表示调用 API 函数的其他软件。

虽然有许多文件格式后缀可以用来标识 C++源文件和头文件，如.cpp、.cc、.cxx、.h、.hh 和.hpp，但本书将统一使用.cpp 和.h。本书会交替使用术语“模块”（module）和“组件”（component）表示一对.cpp 和.h 文件。很明显它们不等价于一个类，因为一个组件或模块可能包含多个类。本书使用术语“库”（library）表示一个物理集合，即组件形成的包。也就是说，库 > 模块/组件 > 类。

面向对象编程社区经常使用的术语“方法”（method）并不是严格的 C++术语，它源自 Smalltalk 语言。在 C++中与之等价的术语是成员函数（member function），但是也有一些软件工程师喜欢使用更为具体的虚成员函数（virtual member function）。本书并不特别关注这些术语的微小差别，故将交替使用方法和成员函数。同样，虽然在 C++中，数据成员（data member）是更准确的表述，但是本书将术语“成员变量”（member variable）视为它的同义词。

考虑到印刷习惯，本书使用等宽字体编排所有源代码示例以及正文中出现的所有关键字。在本书给出的示例中，我习惯使用大写驼峰式命名法（upper camel case）书写所有类和函数的名字，即使用 `CamelCase` 而非 `camelCase` 或 `snake_case`。不过，我会保留引用的外部代码的大小写形式，如 `std::for_each()`。本书遵循在数据成员名前添加 `m` 前缀以及在静态变量前添加 `s` 前缀的习惯约定，如 `mMemberVar`、`sStaticVar`。

应当指出，书中的源代码往往只是代码片段而并非功能完整的样例。我还会省略示例代码中的注

释。这样做是为了简洁和清晰。特别地，我经常会省略头文件前后防止重复引用的预处理警戒语句。本书认定读者都已清楚C/C++头文件需要将所有内容包含在预处理警戒语句[①]（preprocessor guard）中，并且最好将所有的API声明都包含在统一的命名空间下（见第3章和第6章）。换言之，本书假定给出的每个头文件都默认包含如下代码：

```
#ifndef MY_MODULE_H
#define MY_MODULE_H

//需要#include 的文件……

namespace apibook{

//API 声明……

}

#endif
```

提示

本书中，我也会强调各种有关API设计的小提示和关键概念。提供这些标注是为了让你能够快速搜索到想要重读的概念。如果你的时间非常有限，可以只浏览书中的这些小提示，然后阅读提示附近的内容以深入理解你最感兴趣的主题。

本书的网站

本书也有一个支持网站：http://APIBook.com/。该网站提供了本书的全部信息以及一些辅助资料，如书中示例的完整源代码。读者可以随意下载并使用这些样例。样例设计得尽可能简单，但仍然极具价值。本书使用跨平台的CMake构建系统来编译和链接这些示例，因此它们可以在Windows、Mac OS X及Unix操作系统上运行。

我也会在此网站上发布所有对该书的新的修订以及勘误表，还有互联网上其他相关的API资源的链接，如有趣的工具包、文章和实用程序等。

本书的网站上也提供了一个由我编写的名为Diff的API组件。此程序可以通过可视化的对照方式，比较两个版本的API，并检查编码和注释上的不同之处。你也可以用它生成某个特定发行版本的变更报告，这样你的客户就能准确知道需要注意什么了。该工具适用于Windows、Mac OS X及Linux。

① 避免出现重复#include头文件的情况。——译者注

致　谢

我敬重的几位同事的技术审校和反馈让本书受益匪浅。感谢他们抽出时间审读本书早期的手稿并且提出了经过深思熟虑的改进建议。我要特别感谢 Paul Strauss、Eric Gregory、Rycharde Hawkes、Nick Long、James Chalfant、Brett Levin、Marcus Marr、Jim Humelsine 及 Geoff Levner。

在同一些杰出的软件工程师和经理共事后，燃起了我对优秀 API 的设计热情。由于曾就职于几家不同的公司和机构，我接触到了软件设计的各个方面：软件开发理念以及问题的解决方案。通过各种各样的经历，我有幸遇到一些具有独特才能的人，并且从他们那里学到很多东西。我要向下面这些人表示衷心的感谢。

斯坦福研究院：Bob Bolles、Adam Cheyer、Elizabeth Churchill、David Colleen、Brian Davis、Michael Eriksen、Jay Feuquay、Marty A. Fischler、Aaron Heller、Lee Iverson、Jason Jenkins、Luc Julia、Yvan G. Leclerc、Pat Lincoln、Chris Marrin、Ray C. Perrault 以及 Brian Tierney。

Pixar 动画工作室：Brad Andalman、David Baraff、Ian Buono、Gordon Cameron、Ed Catmull、Chris Colby、Bena Currin、Gareth Davis、Tony DeRose、Mike Ferris、Kurt Fleischer、Sebastian Grassia、Eric Gregory、Tara Hernandez、Paul Isaacs、Oren Jacob、Michael Kass、Chris King、Brett Levin、Tim Milliron、Alex Mohr、Cory Omand、Eben Osbty、Allan Poore、Chris Shoeneman、Patrick Schork、Paul Strauss、Kiril Vidimče、Adam Woodbury、David Yu、Dirk van Gelder，Brad West 以及 Andy Witkin。

Bakery 动画工作室：Sam Assadian、Sebastien Guichou、Arnauld Lamorlette、Thierry Lauthelier、Benoit Lepage、Geoff Levner、Nick Long，Erwan Maigret 以及 Baris Metin。

Linden lab：Nat Goodspeed、Andrew de Laix、Howard Look、Brad Kittenbrink、Brian McGroarty、Adam Moss、Mark Palange，Jim Purbrick 以及 Kent Quirk。

我要特别感谢 Yvan G. Leclerc，我在斯坦福研究院工作的初期，他对我的人生产生了巨大的影响。Yvan 是我的第一位经理，同样也是一位真正的朋友。他教会我如何成为一名优秀的员工经理，如何成为一名思维缜密的科学家和工程师，以及与此同时如何充分享受生活。令人无比悲痛的是，像 Yvan 这样了不起的人却英年早逝了。

同时还要感谢 Morgan Kaufmann 出版社，感谢他们尽心尽力地复查、审稿、排版及出版本书。如果没有他们的支持和付出，本书不可能面世。我要特别感谢 Todd Green、Robyn Day、André Cuello 以及 Melissa Revell。

最重要的是，我要感谢我的妻子 Genevieve M. Vidanes，感谢她鼓励我写作本书并忍受我无数个夜晚匍匐在键盘前打字。因为这是我的第二本书，所以她十分熟悉写作对我们个人生活所产生的影响。虽然她在整个写作过程中一直支持我，但也时常提醒我暂停并休息一下。Genevieve，感谢你对我始终如一的爱和支持。

目　录

第1章 API简介

1.1 什么是API

API（Application Programming Interface）提供了对某个问题的抽象，以及客户与解决该问题的软件组件之间进行交互的方式。组件本身通常以软件类库形式分发，它们可以在多个应用程序中使用。概括地说，API定义了一些可复用的模块，使得各个模块化功能块可以嵌入到最终用户的应用程序中去。

你可以为自己、为你所在机构中的其他工程师或大型开发社区编写API。它可以小到只包含一个单独的函数，也可以大到包含数以百计的类、方法、全局函数、数据类型、枚举类型和常量等。它的实现可以是私有的，也可以是开源的。有关API的一个重要的基本定义是：API是一个明确定义的接口，可以为其他软件提供特定服务。

现代应用程序通常都是基于很多API建立起来的，而这些API往往又依赖于其他API。如图1-1中示例应用程序所示，该应用程序用到了3个类库（1、2、3）的API，而这3个API中有2个又用到了另外2个类库（4和5）。举例来说，浏览图片的应用程序可能会用到加载GIF图片的API，而该API本身则可能又依赖更底层的压缩或解压缩数据的API。

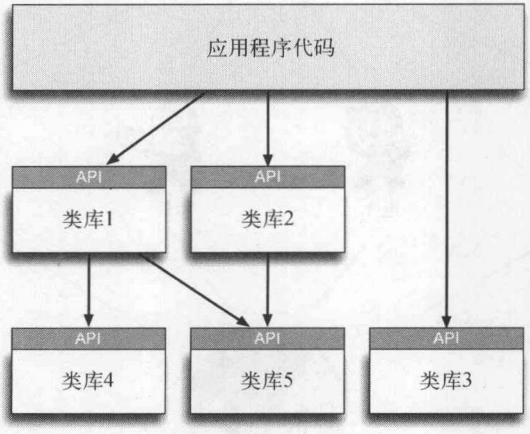

图1-1 从层次化API中调用例程的应用程序。每个方框代表一个软件类库，灰色部分表示其公共接口，对于类库而言即是其API，白色部分表示隐藏在API后面的具体实现

API开发在现代软件开发中随处可见，其目的是为某个组件的功能提供一个逻辑接口，同时隐藏该模块内部的实现细节。举例来说，我们用来读取GIF图片的API可能仅仅提供一个`LoadImage()`方法，后者接收一个文件名作为参数，并返回一个2维的像素数组。所有文件格式和数据压缩的细节全部隐藏在这个看似简单的接口之下。这个概念也在图1-1中进行了说明，即客户端代码只能够通过该API的公有接口访问。API公有接口如图1-1中每个方框顶部的灰色区域所示。

1.1.1　契约和承包人

我们打个比方来对API进行说明。假设你要建造自己的家，如果你想完全靠自己建造房子，可能需要全面理解建筑结构、给排水管道、电气设计、木工技术、石工技术，以及诸多其他知识。而且还需要亲自动手完成每一步工作，并且对每项任务的各个细节都要把握周到，比如是否有足够的木材铺设地板，是否用对与螺丝匹配的螺丝帽等。最后，由于该工程只有你一个人在干，一段时间只能做一件事，因此完成项目的总体时间可能会非常久。

另一种解决方案则是，雇用一些专业的承包人来承担关键任务（见图1-2）。你可以雇用建筑师设计房子蓝图，雇用木匠做所有木工活，雇用水管工安装水管和下水道系统，雇用电工安装电气系统。如果采取这种方法，你要分别和每个承包人沟通——告诉他们你希望完成什么工作，并协商确定价格——然后他们便可以为你做这些工作。如果幸运的话，也许你的某个朋友正好是一位承包人，并且他又愿意免费为你提供服务。这样一来，你就彻底从房屋建设所需要掌握的各种琐事中解脱出来，成为一名高层管理者，选择最好的承包人完成你的目标，并确保将每个人的工作整合到一起，这样便可建造出你心中的理想家园。

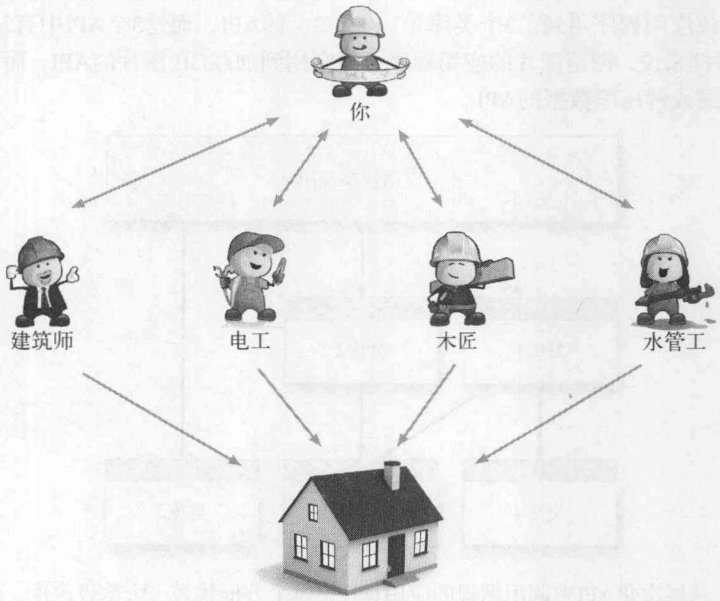

图1-2　使用承包人执行特定任务来建造房子