普通高等教育“十三五”规划教材

SOFTWARE

软件体系结构与设计实用教程

RUANJIAN TIXI JIEGOU YU SHEJI SHIYONG JIAOCHENG

刘其成　毕远伟◎主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

普通高等教育“十三五”规划教材

软件体系结构与设计实用教程

刘其成　毕远伟　主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书得到全国高等院校计算机基础教育研究会立项支持。本书对软件体系结构和软件设计的基本原理和实例进行了系统的阐述，包括软件体系结构的定义和研究内容、经典软件体系结构风格、分布式软件体系结构风格、MVC 风格与 Struts 框架、软件设计的目标、面向对象软件设计方法、设计原则、设计模式等内容。

本书在介绍软件体系结构和软件设计原理的前提下，特别注重实用性。书中含有大量精心设计的程序实例，方便读者学习。本书集作者多年的教学经验编写而成，语言通俗易懂，内容安排合理，讲解深入浅出。

本书适合为普通高等学校软件工程专业、计算机科学与技术专业以及信息类相关专业本科生和研究生的教材，也可作为软件工程培训教材，以及软件开发人员的参考书。

图书在版编目（CIP）数据

软件体系结构与设计实用教程/刘其成，毕远伟主编. —北京：中国铁道出版社，2018.8

普通高等教育“十三五”规划教材

ISBN 978-7-113-24566-5

Ⅰ.①软… Ⅱ.①刘… ②毕… Ⅲ.①软件－结构设计－高等学校－教材 Ⅳ.①TP311

中国版本图书馆 CIP 数据核字（2018）第 176219 号

书　　名：软件体系结构与设计实用教程

作　　者：刘其成　毕远伟　主编

策　　划：周海燕　　**读者热线：**（010）63550836

责任编辑：周海燕　彭立辉

封面设计：穆　丽

责任校对：张玉华

责任印制：郭向伟

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）

网　　址：http://www.tdpress.com/51eds/

印　　刷：北京虎彩文化传播有限公司

版　　次：2018 年 8 月第 1 版　2018 年 8 月第 1 次印刷

开　　本：787 mm×1 092 mm　1/16　印张：17　字数：419 千

书　　号：ISBN 978-7-113-24566-5

定　　价：45.00 元

PREFACE

前言

“软件设计与体系结构”是软件工程专业的核心课程。根据教育部高等学校计算机科学与技术教学指导委员会制定的《软件工程（本科）专业规范》，本课程主要是在学习“软件工程概论”的基础上，进一步深入学习软件体系结构和软件设计，从而提高软件的质量。本书面向普通高等学校的学生和从事软件开发以及相关领域的工程技术人员，紧密结合软件工程专业规范，覆盖规范中软件设计与体系结构课程要求的知识单元和知识点；同时，充分考虑普通高等院校学生的实际情况，加强实践教学的内容。

作者根据多年的教学和软件开发经验，对本书的内容取舍、组织编排和实例都进行了精心设计。在难易程度上遵循由浅入深、循序渐进的原则，特别考虑到普通高等学校本科学生的实际理解和接受能力。与以往许多软件工程相关教材主要以理论为主不同，本书突出实践性，将复杂的理论融于具体的实例和程序之中。书中的实例都是经过精心设计的，程序代码都认真做了调试，可以直接运行，方便读者理解和使用。同时，为了培养学生自学的能力、获取知识的能力，在编写本书的过程中，作者力图在内容编排、叙述方法上留有教师发挥的空间和学生自学的空间。

全书共分 9 章，第 1 章讲述软件体系结构和软件设计的基本概念；第 2 章讲述软件体系结构的定义、组件与连接件、软件体系结构等内容；第 3 章讲述经典软件体系结构风格，包括主程序-子程序风格、面向对象风格、批处理风格、管道/过滤器风格、层次风格、基于事件的隐式调用风格、仓库风格、解释器风格、反馈控制环风格等；第 4 章讲述分布式软件体系结构风格，包括两层 C/S、三层 C/S、B/S、P2P 以及中间件等；第 5 章讲述 MVC 风格的概念及其应用、Struts 框架的原理；第 6 章讲述软件设计的目标，包括正确性、健壮性、可复用性、可维护性和高效性等；第 7 章讲述面向对象软件设计，包括问题域部分、人机交互部分、数据管理部分和任务管理部分；第 8 章讲述软件设计原则，包括开-闭原则、里氏代换原则、合成/聚合复用原则、依赖倒转原则、迪米特法则、接口隔离原则和单一职责原则等；第 9 章从原理、结构和示意源代码三方面介绍主要的创建型设计模式、结构型设计模式和行为型设计模式。

本书由刘其成、毕远伟主编，其中：第 1～8 章由刘其成编写，第 9 章由毕远伟编写，刘霄、王莹洁、王凯、苏浩、鲍凯丽、庞书杰参与了部分内容的编写。刘其成设计了全书的结构，并做了全书的统稿工作。

在本书的编写过程中，参阅了大量书籍和网站等资料，得到了童向荣教授及中国铁道出版社的支持和帮助，在此表示衷心感谢。

本书得到全国高等院校计算机基础教育研究会立项支持。

尽管书稿几经修改，但由于作者学识有限，书中仍难免有疏漏与不当之处，恳请各位同仁、读者不吝赐教。

编　者

2018 年 5 月

CONTENTS

目录

第1章 概 述

学习目标

- 掌握结构化和面向对象软件工程方法学。
- 了解软件设计的概念。
- 了解软件体系结构的概念。

通常把软件生命周期全过程中使用的一整套技术方法的集合称为软件工程方法学，它包括方法、工具和过程 3 个要素。常用的软件开发方法包括结构化方法和面向对象方法。软件体系结构和软件设计是软件生命周期中的重要阶段。

1.1 软件工程方法学

软件生存周期指一个计算机软件从功能确定、设计，到开发成功投入使用，并在使用中不断地修改、增补和完善，直到停止该软件使用的全过程。通常把软件生命周期全过程中使用的一整套技术方法的集合称为软件工程方法学，它包括方法、工具和过程 3 个要素。常用的软件开发方法包括结构化方法和面向对象方法。

1.1.1 结构化方法

结构化软件工程一般把软件开发和运行过程划分为下面几个主要阶段：系统分析、软件设计、程序编写、软件测试、运行和维护，如图 1-1 所示。软件工程强调各阶段的完整性和先后顺序，根据不同阶段的特点，运用不同的手段完成各阶段的任务。软件开发人员遵循严格的规范，在每一阶段工作结束时进行阶段评审，得到该阶段的正确的文档，作为阶段结束的标志，并作为下一阶段工作的基础，逐步实现各阶段的目标，从而保证软件的质量。

1. 结构化软件工程

① 系统分析：确定要开发软件系统的总目标，完成该软件任务的可行性研究；解决软件“做什么”的问题，正确理解用户的需求，并进行分析、综合，给出详细的定义。

② 软件设计：解决软件“怎么做”的问题，结构化的软件设计可分为概要设计和详细设计两部分。概要设计是把软件需求转化为软件系统的总体结构和数据结构，结构中每一部分都是意义明确的模块，每个模块都和某些需求相对应。详细设计也称过程设计，对每个模块要完成的工作进行具体描述，给出详细的数据结构和算法，为源程序的编写打下基础。

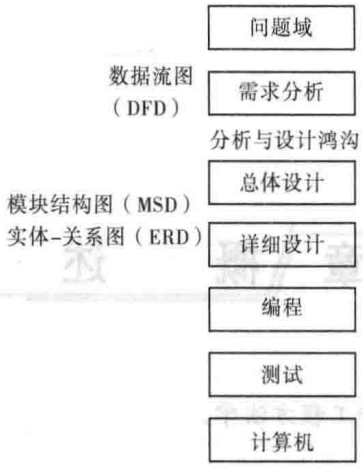

图 1-1 传统软件工程分析与设计

③ 程序编写：根据软件设计阶段的结果，以某一种特定的程序设计语言在计算机上编写程序，真正实现一个具体的软件系统。写出的程序应该是结构良好、清晰易读的，并且要与软件设计相一致。

④ 软件测试：查找和修改系统分析、软件设计和程序编写过程中的错误，以保证软件的质量，包括单元测试、集成测试和有效性测试。单元测试是查找各模块在功能和结构上存在的问题并加以纠正；集成测试将已测试通过的模块按一定顺序组装起来进行测试；有效性测试按规定的各项需求逐项进行测试，判断已开发的软件是否合格，能否交付用户使用。

⑤ 运行和维护：软件系统运行过程中会受到各种人为的、技术的、设备的影响，这就要进行软件维护，使软件适应变化，不断地得到改善和提高，以保证软件正常而可靠地运行。软件维护包括纠正性维护、适应性维护、完善性维护和预防性维护：纠正性维护是在运行中发现了软件中的错误而进行的修改工作；适应性维护是为了适应变化了的软件工作环境而做出适当的变更；完善性维护是为了增强软件的功能而做出的变更；预防性维护是为未来的修改与调整奠定更好的基础而进行的工作。

软件总是有体系结构的，不存在没有体系结构的软件。可以把软件比作一座楼房，具有基础、主体和装饰：基础设施软件是操作系统，主体应用程序实现计算逻辑，用户界面程序方便用户使用。从细节上来看，每一个程序也是有结构的。早期的结构化技术将软件系统分成许多模块，模块间相互作用，自然地形成了体系结构。但是，采用结构化技术开发的软件，程序规模不大，采用自顶向下、逐步求精，只要注意模块的耦合性就可得到相对良好的结构，所以并未特别研究软件体系结构。

2. 结构化软件工程方法存在的问题和缺点

与以前随心所欲的个人化脑力劳动方式相比，结构化软件工程方法是一个较大的进步，在一定程度上解决了软件危机的问题。但是，结构化技术开发的软件，具有稳定性差、可维护性差和可复用性差的缺点。

软件系统产生的错误有很大比例是系统分析不准确导致的，主要原因是在开发初期，用户缺乏计算机方面的知识，难以清楚地给出所有需求，而开发人员缺乏用户的业务知识，很难给出切合实际的软件系统描述，这样就会造成开发出来的软件功能与用户需求不符。软件系统应该具有

可维护性，适应用户需求的变化。传统的结构化软件工程方法在开发过程中很少考虑可维护性，从而很少允许用户需求发生变化，容易导致各种问题的发生。

结构化软件工程方法要等到开发后期（即软件测试阶段）才能得到可运行的软件，如果在这时发现软件功能与用户需求不符，或发现有较大错误，会造成很大的损失和浪费，甚至导致软件项目开发的失败。

1.1.2　面向对象方法

为了解决结构化软件工程方法中存在的问题，应该从考虑问题的方法上着手，尽可能地使分析、设计和实现一个系统的方法接近认识一个系统的方法。面向对象的软件工程仍采用结构化软件工程的某些成熟思想和方法，在软件开发过程中仍采用分析、设计、编程、测试等技术，但在构造系统的思想方法上进行了改进。

软件系统所解决的问题涉及的业务范围称作该软件的问题域。面向对象软件工程强调以问题域中的事物为中心来考虑问题，根据这些事物的本质特征，抽象地表示为对象，作为系统的基本构成单位。因此，面向对象软件工程可以使软件系统直接地映射问题域，使软件系统保持问题域中事物及其相互关系的本来面貌。

1．面向对象软件工程

面向对象的软件工程包括面向对象分析（OOA）、面向对象设计（OOD）、面向对象编程（OOP）、面向对象测试（OOT）和面向对象维护等主要内容，如图 1-2 所示。

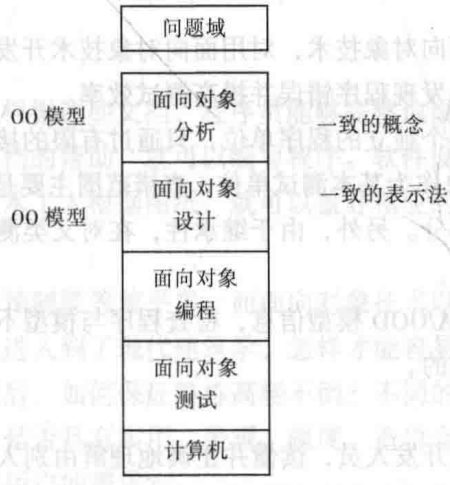

图 1-2　面向对象软件工程分析与设计

（1）面向对象的分析

OOA 直接针对问题域中客观存在的各种事物，建立 OOA 模型中的对象。用对象的属性和服务分别描述事物的静态特征和动态特征。

OOA 模型还描述问题域中事物之间的关系。用一般/特殊结构描述一般类与特殊类之间的关系——继承关系，用整体/部分结构描述事物间的组成关系——聚合/组合关系，用实例连接表示事物之间的静态联系——一个对象的属性与另一对象属性有关，用消息连接表示事物之间的动态联系——一个对象的行为与另一对象行为有关。

无论是对问题域中的单个事物，还是对各个事物之间的关系，OOA 模型都保留着它们的原貌，

没有转换、扭曲，也没有重新组合，所以 OOA 模型能够很好地映射问题域。OOA 所采用的概念及术语与问题域中的事物保持了最大限度的一致，不存在鸿沟。

（2）面向对象的设计

OOA 针对问题域，运用面向对象的方法，建立一个反映问题域的 OOA 模型，不考虑与系统实现有关的编程语言、图形用户界面、数据库等因素，使 OOA 模型独立于具体的实现。OOD 则是针对系统的一个具体的实现，运用面向对象的方法进行设计。首先把 OOA 模型做某些必要的修改和调整，作为 OOD 的一个部分；然后补充人机界面、数据存储、任务管理等与实现有关的部分。这些部分与 OOA 采用相同的表示法和模型结构。

OOA 与 OOD 采用一致的表示法是面向对象软件工程优于传统软件工程方法的重要因素之一。从 OOA 到 OOD 不存在转换，只有局部的修改或调整，并增加几个与实现有关的独立部分。因此，OOA 与 OOD 之间不存在传统软件工程中分析与设计之间的鸿沟，OOA 与 OOD 能紧密衔接，降低了从 OOA 到 OOD 的难度和工作量。

（3）面向对象的编程

认识问题域与设计系统的工作已经在 OOA 和 OOD 阶段完成，对系统需要设立的每个对象类及其内部构成（属性和服务）与外部关系（继承、聚合等）有了透彻的认识和清晰的描述，不会有问题遗留给程序员去思考。

OOP 相对比较简单，它用一种面向对象的编程语言把 OOD 模型的每个部分书写出来，程序员用具体的数据结构来定义对象的属性，用具体的语句来实现算法。

（4）面向对象测试

在测试过程中继续运用面向对象技术，对用面向对象技术开发的软件，进行以对象概念为中心的软件测试，可以更准确地发现程序错误并提高测试效率。

由于封装性，对象成为一个独立的程序单位，只通过有限的接口与外部发生关系，可以减少错误的影响范围。OOT 以对象作为基本测试单位，查错范围主要是类定义之内的属性和服务，以及有限的对外接口所涉及的部分。另外，由于继承性，在对父类测试完成之后，子类的测试重点只是那些新定义的属性和服务。

OOT 还可以通过捕捉 OOA/OOD 模型信息，检查程序与模型不匹配的错误。这一点是传统的结构化软件工程方法难以达到的。

（5）面向对象的软件维护

维护人员一般不是当初的开发人员，读懂并正确地理解由别人开发的软件是件困难的事情；同时，用传统的结构化软件工程方法开发的软件，各阶段文档表示不一致，程序不能很好地映射问题域，从而加重了维护工作的困难。面向对象的软件工程方法为软件维护提供了有效的途径。程序与问题域是一致的，各个阶段的表示是一致的，从而大大减小了理解的难度；无论是发现了程序中的错误而逆向追溯到问题域，还是需求发生了变化而从问题域正向追踪到程序，道路都是比较平坦的。

软件工程从传统的结构化软件工程进入到现代的面向对象软件工程后，需要进一步研究整个软件系统的体系结构，寻求质量最好、建构最快、成本最低的构造过程。在引入了软件体系结构的软件开发之后，应用系统的构造过程变为面向对象分析、软件体系结构、面向对象设计、面向对象编程、面向对象测试和面向对象维护，可以认为软件体系结构架起了面向对象分析和面向对

象设计之间的一座桥梁。

2. 面向对象软件工程的主要优点

① 符合常规的思维方式。面向对象软件工程把问题域的概念映射到对象及其关系，符合人们通常的思维方式，避免了结构化软件工程从问题域到分析阶段的映射问题。

② 连续性好。面向对象软件工程的分析、设计和编码采用一致的模型表示，各阶段文档表示一致，后一阶段可直接利用前一阶段的结果，避免了结构化软件工程各阶段表示方法不连续的问题（数据流图与模块结构图需要转换），降低了理解的难度，减少了工作量并降低了映射误差。另外，程序与问题域一致，发现了程序中的错误，可以方便地逆向追溯到问题域；需求发生了变化，可以方便地从问题域正向地追踪到程序。

③ 可维护性好。面向对象方法具有封装性，可以使对象结构在外部功能发生变化后保持相对稳定，并使改动局限于一个对象的内部。修改一个对象时，很少影响其他对象，避免了波动效应。具体来说，两方面的原因决定了维护对象是容易的。首先，概念独立性（封装）意味着很容易判断出产品的哪一部分必须进行修改，以达到某一确定的维护目标——无论是对完善性维护还是对纠错性维护。其次，信息隐藏确保了对象本身的修改不会在该对象以外产生影响，因此大大降低了回归错误的数量。

④ 可复用性好。面向对象方法的继承性和封装性可以很好地支持软件复用，并易于扩充。

1.2 软件设计与体系结构

1. 软件设计

软件设计形成一套文档，根据这些文档，程序员能够完整地设计出应用程序。软件设计的结果可以使程序员不需要其他文档的帮助，就可以编写程序。软件设计形成的文档就像是建筑物和机械零件的图纸；建筑商和技术工人根据图纸，就可以盖好相应的建筑物和加工好相应的零件。

2. 软件体系结构

结构化技术是以砖、瓦、预制梁等盖平房，而面向对象技术以整面墙、整间房、一层楼梯的预制件盖高楼大厦。土木工程进入到了现代建筑学，怎样才能容易地构造体系结构？什么是合理的组件搭配？重要组件更改以后，如何保证整栋高楼不倒？不同的应用领域（学校、住宅、写字楼）分别需要什么样的组件？是否具有实用、美观、强度、造价合理的组件，从而使建造出来的建筑（即体系结构）更能满足用户的需求？

同样，软件工程也从传统的结构化软件工程进入了现代的面向对象软件工程，需要进一步研究整个软件系统的体系结构，寻求质量最好、建构最快、成本最低的构造过程。

软件体系结构是设计抽象的进一步发展，满足了更方便地开发更大、更复杂的软件系统的需要。随着软件系统规模越来越大、越来越复杂，对软件总体的系统结构设计和规格说明比起对计算的算法和数据结构的选择变得重要得多。

软件体系结构发展的第一个阶段是“无体系结构”设计阶段，以汇编语言进行小规模应用程序开发为特征。第二个阶段是萌芽阶段，出现了程序结构设计主题，主要特征是使用了控制流图和数据流图。第三个阶段是初期阶段，出现了从不同侧面描述系统的结构模型，以 UML 为典型代

表。第四个阶段是高级阶段，以描述系统的高层抽象结构为中心，不关心具体的建模细节，划分了体系结构模型与传统的软件结构的界限。该阶段以 Kruchten 提出的“4+1”模型为标志，目前概念尚不统一，描述规范也不能达成一致。

解决好软件的质量、复用性和可维护性问题，是研究软件体系结构的根本目的。

习　题

1. 是否存在没有体系结构的软件？采用结构化技术开发的软件是否具有体系结构？
2. 软件设计的含义是什么？
3. 简述软件体系结构的发展阶段。
4. 研究软件体系结构的根本目的是什么？

第 2 章 软件体系结构

学习目标

- 理解软件体系结构的定义。
- 掌握组件和连接件的概念。
- 了解软件体系结构的研究内容。

软件体系结构是控制软件复杂性、提高软件系统质量、支持软件开发和复用的重要手段之一，是软件工程的一个重要的研究领域。软件体系结构的主要研究内容包括软件体系结构的分析、设计与验证，评价方法，建模研究，描述语言等形式化工具，发现、演化与复用，软件体系结构风格，基于软件体系结构的软件开发方法，特定领域的软件体系结构，软件产品线体系结构等。

2.1 软件体系结构的定义

从字面上理解，软件体系结构表示软件的体系结构。

体系结构（Architecture）一词起源于建筑学。建筑体系结构包含两个因素：① 基本的建筑模块：砖、瓦、灰、沙、石、预制梁、柱、屋面板……② 建筑模块之间的粘接关系：如何把这些“砖、瓦、灰、沙、石、预制梁、柱、屋面板”有机地组合起来形成整体建筑？建筑设计原则是坚固、实用、美观。建筑设计不仅是一门科学，而且是一项艺术。

计算机硬件系统的“体系结构”包含两个因素：① 基本的硬件模块：控制器、运算器、内存储器、外存储器、输入设备、输出设备……② 硬件模块之间的连接关系：总线等。计算机体系结构的风格有以存储程序原理为基础的冯·诺依曼结构、存储系统的层次结构、并行处理机结构、流水线结构、多核 CPU 等。

体系结构的共性包括一组基本的构成要素（组件）、这些要素之间的连接关系（连接件）、这些要素连接之后形成的拓扑结构（物理分布）、作用于这些要素或连接关系上的限制条件（约束）、质量（性能）。

对于软件体系结构（Software Architecture，SA），组件是指各种基本的软件构造模块（函数、对象、模式等），连接件将它们组合起来形成完整的软件系统，物理分布是指软件系统拓扑结构，约束是指限制条件，性能是指软件质量。

软件体系结构已经在软件工程领域中有着广泛的应用，但迄今为止还没有一个公认的定义。许多专家学者从不同角度对软件体系结构进行了定义，较为典型的有以下几种：

① Dewayne Perry 和 Alex Wolf 认为软件体系结构是组件（具有一定形式的结构化元素）的集

合，包括处理组件、数据组件和连接组件。处理组件负责对数据进行加工，数据组件是被加工的信息，连接组件把体系结构的不同部分组合连接起来。

② Mary Shaw 和 David Garlan 认为软件体系结构是软件设计过程中，超越计算过程中的算法设计和数据结构设计的一个层次。软件体系结构处理整体系统结构设计和描述方面的一些问题，如全局组织和全局控制结构，关于通信、同步与数据存取的协议，设计组件功能定义，物理分布与合成，设计方案的选择、评估与实现等。

③ Kruchten 认为软件体系结构有 4 个角度，从不同方面对系统进行描述：概念角度描述系统的主要组件及其关系，模块角度包含功能分解与层次结构，运行角度描述一个系统的动态结构，代码角度描述各种代码和库函数在开发环境中的组织。

④ Hayes Roth 认为软件体系结构是一个抽象的系统规范，主要包括用其行为来描述的功能组件和组件之间的相互连接、接口和关系。

⑤ David Garlan 和 Dewne Perry 认为软件体系结构是一个程序/系统各组件的结构，它们之间的相互关系以及进行设计的原则和随时间进化的指导方针。

⑥ Barry Boehm 认为软件体系结构包括软件和系统组件、互联及约束的集合；系统需求说明的集合；基本原理，用以说明组件、互联和约束能够满足系统需求。

⑦ Bass、Ctements 和 Kazman 认为一个程序或计算机系统的软件体系结构包括一组软件组件、软件组件的外部的可见特性及其相互关系。软件外部的可见特性是指软件组件提供的服务、性能、特性、错误处理、共享资源使用等。

⑧ 张友生在《软件体系结构》一书中的定义：软件体系结构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素的描述、这些元素的相互作用、指导元素集成的模式以及这些模式的约束组成。软件体系结构不仅指定了系统的组织结构和拓扑结构，并且显示了系统需求和构成系统的元素间的对应关系，提供了设计决策的基本原理。

归纳起来，软件体系结构提供了一个结构、行为和属性的高级抽象；从一个较高的层次来考虑组成系统的组件、组件之间的连接，以及由组件与组件交互形成的拓扑结构；这些要素应该满足一定的限制，遵循一定的设计规则，能够在一定的环境下进行演化；反映系统开发中具有重要影响的设计决策，便于各种人员的交流，反映多种关注，据此开发的系统能完成系统既定的功能和性能需求。

总之，软件体系结构的研究正在发展，软件体系结构的定义也必然随之完善。

软件体系结构是可复用的模型，软件体系结构级的复用意味着体系结构能在具有相似需求的多个系统中发生影响，这比代码级的复用有更大的优点。

2.2 组件与连接件

2.2.1 组件

生活中，组装一台计算机时，人们可以选择多个组件，例如内存、显卡、硬盘等，一个组装计算机的人不必关心内存是怎么研制的，只要根据说明书了解其中的属性和功能即可。不同的计算机可以安装相同的内存，内存的功能完全相同，但它们是在不同的计算机中，一台计算机的内存发生了故障并不影响其他的计算机；也可能两台计算机连接了一个共享的组件：集线器，如果