面向对象分析、设计及应用

汪成为　郑小军　彭本昌　著

国防工业出版社

面向对象分析、设计及应用

汪成为　郑小军　彭木昌　著

国防工业出版社

(京)新登字 106 号

内 容 简 介

面向对象方法学是近年来迅速发展的一个研究领域，它对软件工程、系统工程、人工智能、信息科学、认知科学等学科都有重要的影响。本书详细地阐述了面向对象的基本概念，系统地论述了面向对象的分析与设计，综述了面向对象语言的谱系，并介绍了几种常用的面向对象语言，研讨了面向对象方法的几个典型的应用领域。全书分五篇，共十八章，内容丰富，构思严谨，并给出了大量实例，便于读者学习和借鉴。

本书可作为从事计算机研究、开发、应用、教学的理论工作者和工程技术人员的参考书，也可作为计算机、系统工程、自动化、信息处理等专业研究生和高年级本科生的教材。

面向对象分析、设计及应用

汪成为　郑小军　彭木昌　著

责任编辑　陈　洁

*

国防工业出版社出版发行

（北京市海淀区紫竹院南路 23 号）

（邮政编码　100044）

新华书店经售

北京怀柔王史山胶印厂印刷

*

787×1092 毫米　16 开本　印张 20　插页 2　464 千字

1992 年 9 月第一版　1992 年 9 月第一次印刷　印数：0 001—8 000 册

ISBN 7-118-01038-3/TP・131　　定价：19.30 元

前　言

长期以来，人们一直在设法争取问题空间同求解空间在结构上尽可能的一致，也就是使我们分析、设计和实现系统的方法学同我们认识客观世界的过程尽可能地一致。而这正是面向对象方法学的出发点和所追求的基本原则。

面向对象是一种认知方法学。它既提供了从一般到特殊的演绎手段（如继承等），又提供了从特殊到一般的归纳形式（如类等）。面向对象也是一种程序设计方法学。它基于信息隐蔽和抽象数据类型概念，把系统中所有资源，如数据、模块以及系统都看成“对象”，每个对象封装数据和方法，而方法实施对数据的处理。由于面向对象技术对于软件工程学濒临的困境和人工智能所遇障碍都是很有希望的突破口之一，因而引起了计算机界极大的重视，对它的研究遍及计算机软、硬件和应用各个领域。

当前计算机正朝着分布式处理、并行处理、网络化和软件生产工程化的方向发展，而面向对象方法学作为实施这些目标的关键技术之一，显然同计算机领域的总体发展相一致。因此，人们称之为主宰 90 年代的软件开发方法。其实，面向对象技术中有关模块化、数据抽象和信息隐蔽等概念也是继承了 70 年代软件工程的成果，面向对象技术的贡献是在“对象”这一更高的层次上进行了抽象。目前，面向对象技术除了在应用领域的拓广和技术实现的完善等方面正进行着大量的工作外，还试图在一个比“对象”更具有动态概念、更具有“人工”含义的层次上进行抽象，有人称之为面向智能体（Agent Oriented）技术。正如列宁在《黑格尔逻辑学一书摘要》一文中所说的：“物质的抽象，自然规律的抽象，价值的抽象及其他等等，一句话，那一切科学的（正确的、郑重的、不是荒唐的）抽象，都更深刻、更正确、更完全地反映着自然”。

本书力图把面向对象方法系统化，使之成为一个较完整的体系。因此，作者认为首先应使本书包含较为全面的基本内容，同时也要反映这一新的研究领域的现状和前沿课题以及发展方向。国家 863 高技术计划支持了我们从事“智能化决策支持系统”课题，使我们对面向对象技术进行了更深入的研究和应用。由于面向对象方法学正处在发展和完善阶段，再加上作者水平所限，本书不足之处在所难免，欢迎同行们批评指正。

在此谨向关心、支持和参加“智能化决策支持系统”课题研制的全体同志致敬，向在本书的编写过程中付出巨大辛劳的安瑞飞、王秀清、赵学龄等同志致谢。

汪成为、郑小军、彭木昌
于北京系统工程研究所
1992 年 2 月

目　录

第三篇 面向对象设计

第四篇 面向对象语言

第五篇 面向对象应用

第一篇 绪 论

第一章 面向对象技术的形成与发展

1.1 引 言

长久以来，人们一直在解决这样一个不合理的现象，即：我们认识一个系统的过程和方法同我们用于分析、设计和实现一个系统的过程和方法不很一致。

我们对一个系统的认识是一个渐进过程，是在继承了以往的有关知识的基础上、多次迭代往复而逐步深化的。在这种认识的深化过程中，既包括了从一般到特殊的演绎，也包括了从特殊到一般的归纳。而目前我们用于分析、设计和实现一个系统的过程和方法大部分是“瀑布”型的，即后一步是实现前一步所提出的需求，或者是进一步发展前一步所得出的结果。因此，当越接近系统设计（或实现）的后期时，如要对系统设计（或实现）的前期的结果作修改就越加困难了。同时也只有在系统设计的后期才能发现在前期所铸成的一些差错。当这个系统越大、问题越复杂时，由于这种对系统的认识过程和对系统的设计（或实现）过程不一致所引起的困扰也就越大。

当一个有数十年实践经验的工程师回首往事时，经常把自己以往所完成的一些工程称为“遗憾工程”。这是因为当系统设计开始时（概念设计阶段），用户对系统的需求是比较笼统和抽象的，而那时设计者的设计“自由度”较大，设计者可选用当时已有的原理、方法、工具、环境、构件和器件去设计系统。但随着时间的推进，用户对系统的需求是越来越细致和具体了，如图 1.1 曲线 A 所示 。但设计者对系统进行修改的“自由度”却越来越小了，如图 1.1 曲线 B 所示。还应看到，随着时间的推进，原理、方法、工具、环境、构件和器件的水平是日益提高的，如图 1.2 曲线 A 所示，而设计者采用新技术的“自由度”却受到了限制，如图 1.2 曲线 B 所示。因此，系统越大、周期越长、成为“遗憾工程”的可能性就越大。

为了解决上述这种不合理的现象，就应使我们分析、设计和实现一个系统的方法尽可能地接近我们认识一个系统的方法，换言之，就是应使描述问题的问题空间和解决问题的方法空间在结构上尽可能地一致，也就是使我们分析、设计和实现系统的方法学原理与我们认识客观世界的过程尽可能地一致。这就是面向对象（Object-oriented ）方法学的出发点和所追求的基本原则。

和人们认识世界的规律一样，面向对象的基本方法学认为：客观世界是由许多各种各

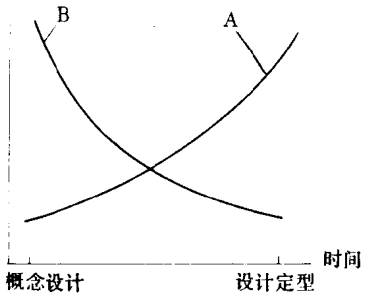

图 1.1　用户和设计者的矛盾

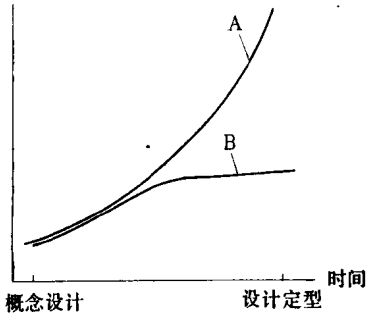

图 1.2　设计者采用新技术的可能性

样的对象所组成的，每种对象都有各自的内部状态和运动规律，不同对象间的相互作用和联系就构成了各种不同的系统，构成了我们所面对的客观世界。当我们设计和实现一个客观系统时，如能在满足需求的条件下把系统设计成是由一些不可变的（相对固定的）部分所组成的最小集合，则这个设计就是优秀的，而这些不可变的（相对固定的）部分就被看成是一些不同的对象。

根据上述对面向对象的基本方法学的粗浅介绍，今后我们所讨论的“对象”至少应当具有以下的特征：

（1）模块性。一个对象是一个可以独立存在的实体。从外部看这个模块，只了解这个模块具有哪些功能，至于这个模块的内部状态，以及如何实现这些功能的细节都是“隐蔽”在模块的内部的。一个模块的内部状态是不受（或很少受到）外界的影响的。同时，一个模块的内部状态的改变也不会影响到其他模块的内部状态。因此，各模块间的依赖性是很小的。所以，各种模块才有可能较为独立地为各个系统所选用；

（2）继承性和类比性。人们是通过对客观世界中的各种对象进行分类及合并等方法而来认识世界的，每个具体的对象都是在它所属的某一类对象（类）的层次结构中占据一定的位置。因此，下一层次的对象应具有上一层次对象的某些属性，在面向对象方法学中，我们把它称为是下一层次的对象继承了上一层次的对象的某些属性。另一方面，当人们发现一些不同的对象具有某些相同的属性时，也常常把它们归并成一个类，在面向对象方法学中，我们把它称为是通过对象间的类比而实现了归类；

（3）动态连接性。在客观世界中，由于存在各式各样的对象以及它们之间的相互连接和作用，从而构成了各种不同的系统。因此，我们把对象和对象间所具有的一种统一、方便、动态地连接和传递消息的能力与机制称为动态连接性；

（4）易维护性。任何一个对象都是把如何实现本对象功能的细节隐藏在该对象的内部。因此，无论是完善本对象的功能，还是改正功能实现的细节，都被圈于该对象的内部，而不会传播给外部，这就增强了对象和整个系统的易维护性。

1.2　面向对象技术的形成、现状与发展

20 世纪 70 年代末，面向对象方法学的一些基本概念已在系统工程领域内萌发出来

了。如对于系统中的某个模块或构件可表示为问题空间的一个对象或一类对象。到了 80 年代，面向对象的程序设计方法得到了很快的发展，并显示出其强大的生命力。因此使面向对象技术在系统工程、计算机、人工智能等领域得到了广泛的应用。人们普遍认为，在更高的层次上、更广泛的领域内开展对面向对象技术的研究将是 90 年代的一个热点。

在 70 年代末，一些系统分析和设计人员已从实践中归纳出一系列符合面向对象方法学原理的一些分析及设计的步骤，只是尚未得到充分和有效的软件工具的支持，这些步骤是：

(1)确定构成该系统的各个组成部分(即对象)及它们的属性；

(2)确定每一组成部分(即对象)应完成的功能；

(3)建立每一组成部分(即对象)与其他组成部分(也是对象)的相互关系；

(4)建立各个组成部分(即对象)间的通信关系和接口形式；

(5)进一步协调和优化各个组成部分的性能及相互间的关系，使得该系统成为是由不同的组成部分(即不同的对象)的最小集合所组成的；

(6)分析、设计及实现每个组成部分(即对象)的功能实现细节。

程序设计语言的发展过程是抽象程度不断演变和提高的过程。对程序设计语言的发展影响较大的几次抽象是过程抽象、语法抽象、数据抽象和进程抽象，但实际上更具有本质意义的是对知识的抽象。70 年代初，具有面向对象方法雏形的软件系统 SIMULA 诞生了。SIMULA 允许用户创建一个面向对象的系统，但它是使用传统的面向数据/ 过程的语言(ALGOL)编写的。不久，在 1976 年推出了 Smalltalk-72，1978 年推出了 Smalltal-76，1981 年推出了 Smalltalk-80，由此逐步发展和完善了面向对象的程序设计语言的概念。例如，在 Smalltalk-72 中，列表结构和控制结构都具有了对象的概念，但类的概念尚未形成。在 Smalltalk-76 中就有了类的概念，并且用面向对象技术的观点去说明程序接口的性质。由 Xerox Learning Research Group 所研制的 Smalltalk-80 系统，则是较全面地体现了面向对象程序设计语言的特征。由于 Smalltalk-80 的推广使用，使得面向对象的方法和原理很快地推广到与信息技术有关的其他各个领域。

在此还必需提到 Ada 语言及其环境在推广面向对象方法和技术上所起的作用。Ada 是美国国防部为解决 60 年代末到 70 年代初的软件危机而投入了巨大的人力和物力所研制的一种语言和环境，目前 Ada 已成为世界上许多国家的标准军用语言，并且已扩展到非军用的其他各个领域。Ada 并非是一种标准的面向对象的语言，但由于 Ada 所具有的模块化、信息隐蔽、数据抽象、并发任务执行等特点，所以 Ada 所倡导的是一种非常接近面向对象方法学原理的系统设计思想。因此，由于 Ada 的推广和应用，以及它所取得的良好效果，也强有力地推动了面向对象技术的发展。目前人们已普遍认为：Ada 是一种基于对象(Object-based)的语言。

除了 Smalltalk 和 Ada 外，我们还应提到的就是 Modula-2 了。在 Pascal 语言推出后的 10 年，于 1978 年定义了一种 Modula-2 的语言，并且于 1979 年首次研制出 Modula-2 的编译程序。在这种语言中特别强调了模块的概念，并且还提出了模块是可嵌套的特性。在每一个模块内(也可是在不同的模块内)的各个对象间的联系是通过 IMPOT 和 EXPORT 功能实现的。显然，Modula-2 所提出的这些概念也对后来的面向对象技术的发展做出了贡献。

从 80 年代起，人们基于以往已提出的有关信息隐蔽和抽象数据类型等概念，以及由 Modula-2，Ada 和 Smalltalk 等语言所奠定的基础，再加上客观需求的推动，逐步地发展和建立起较完整的面向对象的软件系统(OOS——Object Oriented Software System)的概念和机制。按照面向对象的软件系统的方法学，我们对一些将要用于面向对象技术中的基本概念给出以下定义：

- 信息(Information)是对事物的一种表示和描述。
- 软件(Software)是描述信息处理的信息。
- 软件系统(Software System)是一种软件处理工具。
- 程序设计环境(Programming Environment)是一个用于设计、生产和使用软件系统的环境。
- 对象(Object)是一个由信息及有关对它进行处理的描述所组成的包。
- 消息(Message)是对某种对象处理的说明。
- 类(Class)是对一个或几个相似对象的描述。
- 实例(Instance)是被某一个特定的类所描述的一个对象。因此，每一个对象都是某个类的一个实例，而类是对各个实例的全部相似性的描述。
- 方法(Method)是描述对象对消息的响应。

对于上述的这些概念，本书将在以下的各个章节内陆续给出更为精确的定义和详细的说明。但是我们已初步认识到，对象是一种普遍适用的基本逻辑结构，是一个以有组织的形式含有信息的实体。它既可以表示一个抽象的概念，也可以表示一个具体的模块，既可以表示软件，也可以表示硬件。于是，面向对象的方法学既提供了一个分析、设计和实现系统的统一方法，又提供了描述、设计和实现硬件和软件系统的统一框架。

把以上所述的这种面向对象的软件系统的基本概念和运行机制应用到其他的各个领域后，就得到了一系列相应领域的面向对象的技术和应用，这在 80 年代得到了很大的发展：

(1)面向对象的设计(OOD——Object Oriented Design)。系统的设计过程可看成是把系统所要求解的问题分解为一些对象及对象间传递消息的过程；

(2)面向对象的语言(OOL——Object Oriented Language)。在这种语言中，可以把数据和处理数据的过程结合为一个对象。对象既可以象数据一样被处理，又可以象过程一样描述处理的流程和细节。例如：在 Smalltalk-80 中，把一切都看成是对象；在 Ada 中，把包(Package)看成是对象；在 Modula-2 中，把模块看成是对象；

(3)面向对象的数据库(OODB——Object Oriented Data Base)。把对象作为存取和检索的单位，把传统数据库定义的数据定义语言和数据操作语言融为一体，这种概念也是构成语义数据库和知识库的基础；

(4)面向对象的智能程序设计(Object Oriented Intelligent Programming)。把知识系统中的智能实体作为对象，一个对象所具有的知识是该对象的静态属性，一个对象所具有的知识处理方法则是该对象的智能行为的体现；

(5)面向对象的体系结构(Object Oriented Architecture)。能支持对于对象的描述、存储和处理的计算机体系结构。一些新的计算机体系结构都在试图能支持面向对象的程序设计和软件系统的概念，如某些多处理器系统、神经网络计算机及连接机制等。

面向对象技术在80年代已经得到了很大的发展，并且已在计算机科学、信息科学和系统科学中得到了有效的应用，显示出其强大的生命力。面向对象方法学的最基本点是尽可能地模拟人的思维方式，并且和方法学相适应的面向对象技术也提供了这样的可能性：即可以随着对某个系统的需求逐步具体的过程而逐步地设计和实现这个系统。

人们预计在90年代内，面向对象技术将会在更深、更广、更高的方向上取得进展：

(1)更深的方向。如面向对象技术的理论基础和形式化描述；用面向对象技术的概念设计操作系统等；

(2)更广的方向。如面向对象的知识表示；面向对象的仿真系统；面向对象的多介质(Multi-media)系统；面向对象的灵境(Virtual Reality)系统等；

(3)更高的方向。如从思维科学的高度来丰富面向对象方法学的本质属性，突破现有的面向对象技术的一些局限、研究统一的面向对象的范式(Paradigm)等。

第二章　面向对象的方法学

2.1　认知方法学

我国著名的科学家钱学森从80年代初起发表了一系列有关思维科学(Noetic Science)的论文。他指出:"思维科学的目的在于研究人认识客观世界的规律和方法。因此我现在建议思维科学的一个别名是'认识科学',英文的Cognitive Science①。当然国外所说的认识科学的范围比这里讲的要窄……。"钱学森教授认为:"思维学又可细分为抽象(逻辑)思维学、形象(直感)思维学和灵感(顿悟)思维学三个组成部分。"他接着指出:由于认知科学受"认知即计算"(Cognition as Computation)的影响较深,因此目前的认知科学还很少涉及灵感思维的问题。

我们认为:面向对象技术是属于思维科学中的一项工程技术,面向对象方法学是属于思维科学中的一项技术科学。80年代内,面向对象的方法与技术对抽象(逻辑)思维的研究起了一些支撑作用(但还很不充分)。90年代内,面向对象的技术与方法将成为研究抽象(逻辑)思维的强有力的手段,并且有可能通过对多介质系统和灵境系统等的研究而对形象(直感)思维的研究也起到一些支撑作用,但还看不到它对灵感(顿悟)思维学研究的影响。因此,我们在此只讨论面向对象技术的认知方法学中有关抽象(逻辑)思维和形象(直感)思维的一些问题。

抽象思维是以抽象的概念为基础的,形象思维是以具体的形象为基础的。从人们认识世界和获取知识的认知过程来看,不论是抽象思维还是形象思维,主要是通过从一般到特殊的演绎方法和从特殊到一般的归纳方法来进行的。数百年来,人们对抽象思维的研究已取得一系列的成果,对于形式逻辑、辩证逻辑和数理逻辑都已建立了有关演绎和归纳的较完整的理论和方法体系。在形象思维中,这种演绎或归纳都是在形象间的"相似"这一关系上进行的。"客观事物发展过程中存在着同和变异。只有同,才能有所继承;只有变异,事物才能往前发展"。当人们利用形象思维去认识客观事物和改造客观事物时,首先是"按照相似原理和关系,把所要研究的问题区分成一定的相似系统与类别","分类之后,进一步对事物进行详细的解剖和分析",解剖分析后再进行"综合优化",并在事物的运动中和运动的相互关系中去考察客观事物的静态相似和动态相似的关系、宏观相似和微观相似的关系、纵向相似和横向相似的关系。

面向对象技术就是遵循上述的认知方法学的基本概念而建立自己的方法学基础的。面向对象方法学认为:客观世界是由各种"对象"所组成的,任何事物都是对象,每一个对象都有自己的运动规律和内部状态,每一个对象都属于某个对象"类",都是该对象类的一

---

① 我国科技界把Cognitive Science译为"认知科学"。——作者

个元素。复杂的对象可以是由相对比较简单的各种对象以某种方式而构成的。不同对象的组合及相互作用就构成了我们要研究、分析和构造的客观系统。

面向对象方法学认为：通过类比，发现对象间的相似性，即对象间的共同属性，这就是构成对象类的根据。在按“类”、“子类”、“父类”（或“超类”）的概念构成对象类的层次关系（或树形结构）时，如不加特殊说明，则处在下一层次上的对象可自然地继承位于上一层次上的对象的属性。

面向对象方法学认为：对于已分成类的各个对象，可以通过定义一组“方法”来说明该对象的功能，也即是：允许作用于该对象上的各种操作。对象间的相互联系是通过传递“消息”来完成的，消息就是通知对象去完成一个允许作用于该对象的操作。至于该对象将如何完成这个操作的细节，则是封装在相应的对象类的定义中的，对于外界是隐蔽的。

综上所述，面向对象的方法学是比较自然地模拟了人类认识客观世界的方式。从方法学的角度来看，如果有可能利用现有信息技术的手段来实现形象思维的载体和表示方法，则面向对象方法学就不只是适用于对抽象（逻辑）思维的表示和处理，也有可能适用于对形象（直感）思维的表示和处理。

由于面向对象方法学具有很强的类的概念，因此它就能很直接地模拟人类在认识过程中的由一般到特殊的演绎功能或由特殊到一般的归纳功能，类的概念既反映出对象的本质属性又提供了实现对象共享机制的理论根据。

2.2 程序设计方法学

2.2.1 引言

从60年代末开始，计算机软件的作用已得到普遍的重视，各种大型的、复杂的软件系统已陆续问世。但是，随着软件系统规模的扩大和复杂性的增加，软件的开销（开发时所耗费的人力、物力和运行时所占用的硬件资源及运算时间）也惊人地增加了，软件系统的可靠性和可维护性却明显地降低了。人们惊呼这是一种危机，是一种软件危机！产生这种危机的根本原因是：用冯·诺依曼机（Von Neumann Machine）所求解的问题的问题空间结构与冯·诺依曼机所用的求解问题方法的方法空间结构的不一致性。这种不一致性主要表现在以下几方面：

1. 语言的鸿沟

人们在认识问题、分析问题时，是把问题分解为一些对象以及各对象间的组合和联系。面向对象的语言能比较自然地反映人们思考问题的方式，因此也比较容易为人们所理解。但冯·诺依曼机所能直接接受的是面向过程的语言，在这种机器上运行的传统软件是遵循数据/过程的风格编写的。它是由一组被动的数据和一组能动的过程所组成的。其中，数据表示某种信息，而过程表示对数据的处理，激活后的过程即为进程。这种面向过程的语言是一种比较难以理解的、与人的自然语言距离较远的语言。我们可以用图2.1来表示各种语言的性质。当然，最理想的计算机语言应该是最接近人类自然语言的语言。而目前在计算机语言和人的自然语言之间还存在着一条鸿沟。

2. 程序设计的鸿沟

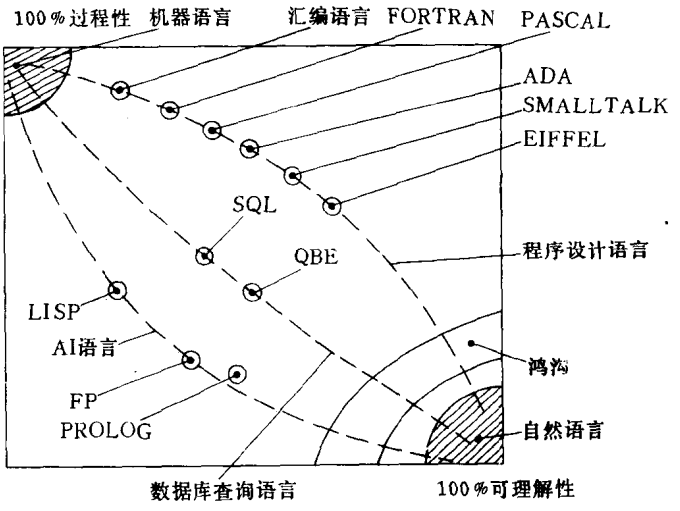

图 2.1 计算机语言与自然语言间的鸿沟

在 1.2 节中已经提到，程序设计环境是一种处理软件系统的软件系统，是一个用于设计、生产和使用软件系统的环境。因此，程序设计过程也就是一个软件的开发过程。近 30 年来，人们对软件的开发过程和开发方法进行了大量的研究，目的是为了提高软件的生产效率和质量。到目前为止，最典型的传统方法是遵循结构化的需求分析(SA——Structured requirement Analysis)、结构化的系统设计(SD——Structured system Design)和结构化的程序设计(SP——Structured Programming)的过程和方法。

结构化的系统分析是建立在系统生命周期(System Life Cycle) 的概念之上的，即任何一个系统从概念到实现都是通过以下几个步骤进行的：问题定义、可行性研究、分析、系统设计、详细设计、实现和维护。每一步骤所要解决的关键问题及遵循的准则分别叙述如下：

- 问题定义

  关键问题：问题是什么？

  准则：对问题域和目的的说明。

- 可行性研究

  关键问题：存在可行的解答吗？

  准则：投入/ 产出的概略分析，

  确定系统域和目的。

- 分析

  关键问题：为解决该问题必须做些什么？

  准则：系统的逻辑模型，

  数据流图，

  数据字典，

  算法。

- 系统设计

  关键问题：一般而言应如何解决该问题？

  准则：几种不同的求解方法，