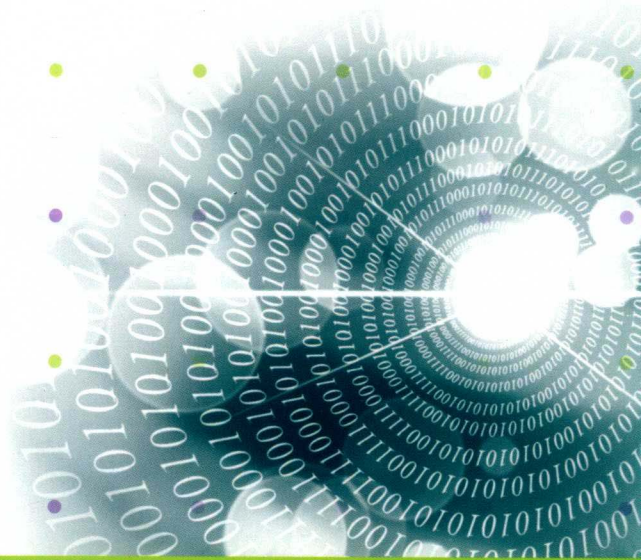
应用型本科信息大类专业“十三五”规划教材



软件工程导论

主编　鲁星　钱小红　曾丹

華中科技大学出版社
http://www.hustp.com

应用型本科信息大类专业“十三五”规划教材

软件工程导论

主　编　鲁　星　钱小红　曾　丹
副主编　雷　渊　宋传磊　崔欢欢
参　编　肖莹慧　王　静　刘胜艳

華中科技大學出版社
http://www.hustp.com
中国 · 武汉

图书在版编目(CIP)数据

软件工程导论/鲁星，钱小红，曾丹主编. —武汉：华中科技大学出版社，2019.2
ISBN 978-7-5680-4979-5

Ⅰ. ①软… Ⅱ. ①鲁… ②钱… ③曾… Ⅲ. ①软件工程-高等学校-教材 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2019)第 033331 号

软件工程导论
Ruanjian Gongcheng Daolun

鲁 星 钱小红 曾 丹 主编

策划编辑：曾 光
责任编辑：舒 慧
封面设计：孢 子
责任监印：朱 玢
出版发行：华中科技大学出版社(中国·武汉) 电话：(027)81321913
武汉市东湖新技术开发区华工科技园 邮编：430223
录 排：华中科技大学惠友文印中心
印 刷：武汉市籍缘印刷厂
开 本：787mm×1092mm 1/16
印 张：11
字 数：282 千字
版 次：2019 年 2 月第 1 版第 1 次印刷
定 价：38.00 元

目录

CONTENTS

第1章 软件工程概述

在学习了“高级程序设计语言”和“数据结构”后，编写小程序不会有太大问题。但要开发一个大型软件，一定存在很多困难，例如在接到项目后应该从哪儿入手、用什么方法、按照哪些步骤进行开发、如何评价一个软件的好坏等，这些都是初次参加大型软件开发的人员要遇到的问题。因此，必须学习软件工程。

1.1 软件技术概述

1.1.1 软件的概述与特点

程序是一系列指令序列的集合，它能被计算机理解和执行。

文档是指用自然语言或者形式化语言所编写的文字资料和图表，用来描述程序的内容、组成、设计、功能规格、开发情况、测试结果及使用方法，如程序设计说明书、流程图、用户手册等。

软件是计算机系统中与硬件子系统相互依存的另一个子系统，是一个包含程序及其文档资料的完整集合，提供用户与硬件子系统之间的接口。随着计算机科学技术的发展，人们对软件的认识也在不断深化，这从下面式子的变化就可以看出：

20 世纪 70 年代以前：软件＝程序。

20 世纪 70—80 年代：软件＝程序＋文档。

20 世纪 80 年代以后：软件＝文档＋程序。

在软件的可维护性变得越来越重要的今天，文档的地位也提高到前所未有的高度，并且能够自动化地生成。

与小型软件不同，大型软件具有如下特点：

(1) 规模大。

现在的软件动辄上百兆，需要处理的数据量大，占用的内存也大。对于实时软件，除了规模大以外，还要求可靠性高。

(2) 复杂性高。

大型软件由大量的模块集成，模块之间的关系、调用方式以及数据和文件的关系都相当复杂。

(3) 开发周期长。

大型软件从立项到交付使用，需几十人、几百人经过几个月甚至几年的时间。

(4) 开发、维护和使用人员不同。

(5) 多学科综合。

软件开发人员除了具有必备的软件知识外，还应该具有多方面的专业知识和经验。

1.1.2 计算机软件技术

计算机软件技术是指开发计算机软件所需的所有技术的总称。按照软件分支学科的内容划分，计算机软件技术应有如下几个领域：

(1) 软件工程技术:包括软件开发的原则与策略、软件开发的方法与软件过程模型、软件标准与软件质量的衡量、软件开发的组织与项目管理、软件工程工具和环境等。

(2) 程序设计技术:包括程序的结构与算法设计、程序设计风格、程序设计语言、程序设计方法和程序设计自动化、程序的正确性证明和程序的变换。

(3) 软件工具环境技术:包括人机接口技术、软件自动生成、软件工具的集成、软件开发环境和软件的复用等。

(4) 系统软件技术:包括操作系统、编译方法、分布式系统的分布处理与并行计算、并行处理技术和多媒体软件技术。

(5) 数据库技术:包括数据模型、数据库与数据库管理系统、分布式数据库、面向对象的数据库技术、工程数据库、多媒体数据库、数据仓库和数据挖掘等。

(6) 实时软件技术。

(7) 网络技术:包括网络软件技术、调试工程、网络管理、局域网技术、网络互联技术和智能网络等。

1.1.3 软件复用

1. 软件复用概述

从1968年提出可复用库的思想后,软件复用的概念被推广了。软件复用是指在构造新的软件系统的过程中,对已存在的软件产品(设计结构、源代码、文档等)重复使用的技术。

软件复用有三个层次:知识复用、方法复用和软件成分复用。前两个属于知识工程的范畴,这里只讨论软件成分复用。

软件成分复用包括以下三个级别:

(1) 代码复用:可以采用源代码剪贴、源代码包含和继承来实现。

(2) 设计结果复用:复用某个软件系统的设计模型,适用于软件系统的移植。

(3) 分析结果复用:复用某个软件系统的分析模型,适用于用户需求未改变而系统体系结构变化的场合。

不属于软件复用的范畴:程序的重复运行、执行期间的重复调用等。

软件复用的优点:由于软件复用利用已有的软件成分来构造新的软件,因此大大缩减了软件开发所需的人力、物力、财力和开发时间,并且能提高软件的可靠性和可维护性。

2. 软件复用技术

软件复用技术分为两类:合成技术和生成技术。

1) 合成技术

合成技术是指利用部件(component,组件,构件)合成软件系统的技术。部件是可复用的一小段软件(可为二进制形式),可以是对某一函数、过程、子程序、数据类型、算法等可复用软件成分的抽象,封装了功能细节和数据结构,有详细的接口。

Microsoft 等公司提出了 OLE/COM(object linking embedding/component object model,对象连接与嵌入/组件对象模型)概念,并开发出各种独立的标准组件,用户使用这些组件集成自己的软件,提高了软件的质量,软件维护更加容易,同时降低了软件开发成本。

目前有三个重要的部件技术:

(1) OMG 的 CORBA 技术。

CORBA 技术是异构系统中的分布式部件技术。CORBA(common object request

broker architecture，公共对象请求代理体系结构）是由 OMG（object management group）提出的应用软件体系结构和对象技术规范，其核心是一套标准的语言、接口和协议，以支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象复用。在 1990 年开始制定并且逐步完善部件标准 CORBA 3.0。

（2）Microsoft 的 COM+技术。

COM+是微软公司在新的企业应用体系结构下，将 COM、DCOM 和 MTS 统一起来，形成真正适合于企业级应用的部件技术。COM+容易使人产生误解，以为它是 COM 的新版本，其实 COM+的含义远比 COM 丰富得多。COM+是一种中间件技术的规范，其要点是提供建立在操作系统上的、支持分布式企业级应用的“服务”。COM+是在 20 世纪末随着 Windows 2000 发布才面世的。

（3）Sun 公司 JavaBeans API，基于 Java 的部件技术标准。

有三种方法将部件合成更大的部件：

①连接：标准函数库中的标准函数靠编译和连接程序与其他模块一起合成系统。

②消息传递和继承：Smalltalk。

③管道机制：UNIX 中用管道（pipe）连接命令 shell，使前一命令的输出作为后一命令的输入，用管道机制把多个 shell 命令连接在一起，完成一个更加复杂的系统。

2）生成技术

利用可复用的模式，通过生成程序，产生一个新的程序或程序段，产生的程序可以看成是模式的实例。可复用的模式有两种：代码模式和规则模式。

（1）代码模式：可复用的代码模式存在于应用生成器中，通过特定的参数替换，生成抽象软件模块的具体实体，例如各种程序生成器。

（2）规则模式：利用程序变换系统，把用超高级规格说明语言编写的程序转化成某种可执行语言的程序，例如 IDL-CORBA 的接口定义语言。

1.2 软件危机

1.2.1 软件危机概述

“软件工程”起因于“软件危机”。20 世纪 60 年代末期出现的软件危机，使软件陷入“泥潭”之中。什么是软件危机？软件危机是指在软件开发过程中遇到的一系列严重问题，如开发周期延长、成本增加、可靠性降低等。

例 1-1 IBM OS/360 系统有 346 万条汇编语句，1968 年至 1978 年投入 5000 人年，共改 21 版，结果不能使用。

例 1-2 1962 年美国飞往金星的探测卫星发射失败，原因是控制系统中的一个 FORTRAN 循环语句 DO 5 I=1,3 被误写成 DO 5 I=1.3。由于空格对 FORTRAN 编译程序没有意义，误写的语句被当成了赋值语句 DO 5 I=1.3，一点之差，使卫星偏离轨道，只好下令引爆，导致 1850 万美元的损失。

```
DO 5 I=1,3
循环体
5 K=X/Y+34.6
```

除了不能正常运行的软件，软件危机还反映在如下几个方面：

(1) 对软件成本、开发成本和开发进度的估计不准确，软件成本在计算机系统总成本中所占的比例逐年上升；

(2) 用户对已完成的软件系统不满意的现象时常发生；

(3) 软件产品的质量往往靠不住；

(4) 软件通常没有适当的文档资料，维护困难；

(5) 软件开发生产率的提高速度远跟不上计算机应用的普及和深入的趋势。

1.2.2 软件危机产生的原因

在1946年第一台计算机诞生后的很长一段时间里，人们都是用计算机来解决一些“小”问题，编制一些小程序。随着计算机软、硬件的发展，人们用计算机来解决的问题越来越“大”，程序规模也越来越大，而开发大型软件与编制小程序有一定的区别：

(1) 人员。小程序从确定要求、设计、编制、使用直到维护，通常由一个人完成；大型软件则必须由用户、项目负责人、分析员、初级程序员、资料员、操作员等组成开发团队来协同完成。

(2) 文档。小程序是编制者脑中的“产品”，很少有书面文档；大型软件则是集体劳动的“产物”，必须有规范化的文档，便于开发和维护。

(3) 产品。小程序通常是一次性的，如果需做大的修改，则宁可舍弃旧程序而重新编写；但大型软件的开发耗费了大量的人力与物力，所以不可能轻易抛弃，而总是在旧软件的基础上一再改动，以延长它的使用寿命，因此“版本”在不断升级。

大型软件的开发提出了许多新的问题，而开发方法却还停留在编制小程序的方法上，经验和技巧已不能满足开发大型软件的需要，导致软件开发过程混乱；使用的开发方法和技术不当，没有适当的文档，不易交流，维护困难，开发成本高，软件质量低等，这些问题是造成软件危机的主要原因。

1.2.3 软件危机的解决方法

以“工程化”的思想来指导软件开发。软件危机使人们认识到，软件的研制和开发不能像以前那样——开发过程混乱、无规范化的文档、个体作坊式的开发，而必须立足于科学理论的基础上，像生产产品、研制一台机器或建造一座楼房那样，以“工程化”的思想来指导软件开发，解决软件开发过程中面临的困难和混乱，从根本上解决软件危机。

从技术上，以软件工程技术、程序设计方法和技术为基础，力求将软件工程与知识工程、人工智能技术结合起来，以构造基于知识的软件开发环境。

从管理上，以管理学为依托，对开发人员、成本、项目、文档等加强管理，对软件开发全过程进行控制。

1.3 软件工程

1.3.1 软件工程的概念

软件工程是指用工程的概念、原理、技术和方法来开发和维护软件，把经过时间考验证明是正确的管理技术和当前能够得到的最好的技术、方法结合起来，指导计算机软件的开发和维护的工程学科。

软件工程采用的生命周期方法学是指从时间的角度对软件开发和维护的复杂问题进行分解，把软件生存的漫长周期依次划分为若干阶段，每个阶段都有相对独立的任务，然后逐步完成每个阶段的任务。

软件生命周期是指从软件开发项目的提出到软件产品完成使命而报废的整个时期。

软件生命周期划分为三个大的阶段：软件定义阶段，包括问题定义、可行性研究和需求分析三个子阶段；软件设计阶段，包括总体设计、详细设计、编码和测试四个子阶段；软件维护阶段，使软件在运行期间满足用户的需要。

软件生命周期可以用瀑布模型来表示，如图 1-1 所示。

图 1-1 瀑布模型

传统的瀑布模型的特点如下：

(1) 阶段间有顺序性和阶段性；

(2) 推迟实现的观点；

(3) 质量保证的观点。

经验表明，越早潜伏的错误越晚发现，纠正错误所花费的代价也越大。因此，及时审查和纠正错误，是保证软件质量、降低软件成本的重要措施。

1.3.2 软件工程原理

B. W. Boehm 总结出七条软件工程基本原理。

1. 严格按照计划进行管理

有人经统计发现，在不成功的软件项目中有一半左右是由于计划不周造成的，可见，把建立完善的计划作为第一条基本原理是吸取了前人的教训而提出来的。

在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作。这条基本原理意味着，应该把软件生命周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Boehm 认为，在软件的整个生命周期中应该制定并严格执行六类计划：项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划、运行维护计划。不同层次的管理人员都必须严格按照计划各尽其职地管理软件开发与维护工作，绝不能受客户或上级人员的影响而擅自背离预定计划。

2. 坚持进行阶段评审

软件的质量保证工作不能等到编码阶段结束之后再进行。这样说至少有两个理由：第一，大部分错误是在编码之前造成的，例如，根据 Boehm 等人的统计，设计错误占软件错误的 63%，编码仅占 37%；第二，错误发现与改正得越晚，所需付出的代价就越大。因此，在每个阶段都进行严格的评审，以便尽早发现在软件开发过程中所犯的错误，这是一条必须遵循的重要原则。

3. 实行严格的产品控制

在软件开发过程中不应随意改变需求，因为改变一项需求往往需要付出较大的代价。但是，在软件开发过程中改变需求又是难免的，由于外部环境的变化，相应地改变用户需求是一种客观需要，显然不能硬性禁止客户提出改变需求的要求，而只能依靠科学的产品控制技术来顺应这种要求。也就是说，当改变需求时，为了保持软件各个配置成分的一致性，必

须实行严格的产品控制，其中主要是实行基准配置管理。所谓基准配置，又称基线配置，它们是经过阶段评审后的软件配置成分（各个阶段产生的文档或程序代码）。基准配置管理也称变动控制，是指一切有关修改软件的建议，特别是涉及对基准配置的修改建议，都必须按照严格的规程进行评审，获得批准以后才能实施修改。绝对不能谁想修改软件（包括尚在开发的软件），就随意进行修改。

4. 采用现代化的程序设计技术

从提出软件工程的概念开始，人们一直把主要精力用于研究各种新的程序设计技术。20 世纪 60 年代末提出的结构程序设计技术，已经成为绝大多数人公认的先进的程序设计技术。以后又进一步发展出各种结构化分析（SA）与结构化设计（SD）技术。实践表明，采用先进的技术既可提高软件开发的效率，又可提高软件维护的效率。

5. 结果应能清楚地审查

软件产品不同于一般的物理产品，它是看不见摸不着的逻辑产品。软件开发人员（或开发小组）的工作进展情况可见性差，难以准确度量，从而使得软件产品的开发过程比一般产品的开发过程更难于评价和管理。为了提高软件开发过程的可见性，更好地进行管理，应该根据软件开发项目的总目标及完成期限，规定开发组织的责任和产品标准，从而使得所得到的结果能够清楚地审查。

6. 开发小组的人员应该少而精

这条基本原理的含义是，软件开发小组的组成人员的素质应该好，而人数则不宜过多。开发小组人员的素质和数量是影响软件产品质量和开发效率的重要因素。素质高的人员的开发效率比素质低的人员的开发效率可能高几倍至几十倍，而且素质高的人员所开发的软件中的错误明显少于素质低的人员所开发的软件中的错误。此外，随着开发小组人员数量的增加，因为交流情况、讨论问题而造成的通信开销也急剧增加。当开发小组人员数量为 N 时，可能的通信路径有 $N/2$ 条。可见，随着人员数量 N 的增加，通信开销将急剧增加。因此，组成少而精的开发小组是软件工程的一条基本原理。

7. 承认不断进行软件工程实践的必要性

遵循上述六条基本原理，就能够按照当代软件工程基本原理实现软件的工程化生产。但是，仅有上述六条原理并不能保证软件开发与维护的过程能赶上时代前进的步伐，能跟上技术的不断进步。因此，Boehm 提出应把承认不断进行软件工程实践的必要性作为软件工程的第七条基本原理。按照这条原理，不仅要积极主动地采纳新的软件技术，而且要注意不断总结经验，例如，收集进度和资源耗费数据、收集出错类型和问题报告数据等。这些数据不仅可以用来评价新的软件技术的效果，而且可以用来指明必须着重开发的软件工具和应该优先研究的技术。

1.3.3 软件开发方法简介

简单介绍三种常用的软件开发方法：结构化方法、快速原型法和面向对象法。

1. 结构化方法

结构化方法是指以“结构化”的思想、方法和工具进行软件开发。

“结构化”是指用一组标准的准则和工具从事某项工作，它最早出自结构化程序设计。

结构化程序设计的基本思想是：只使用顺序、选择、循环三种基本结构来编写程序，它们都是单入口单出口的；使用自顶向下逐步求精的程序设计方法，即利用三种基本结构实现程

序结构的连续分解，产生较低层次的结构，直到设计下降到能使用低层伪代码或高级语言中的三种基本语句表达为止。

结构化程序设计是成功的程序设计方法，但不能解决系统的结构问题，更不能解决系统总体模型表达方面的问题。

结构化系统设计原则如下：

(1) 一个系统由层次化的程序模块构成；

(2) 每个模块只有一个入口和一个出口；

(3) 每个模块归其上级模块调用；

(4) 应当构造内部联系紧密的模块，降低模块间的联系；

(5) 使用系统结构图等图形工具表达系统结构；

(6) 结构化设计采用自顶向下的模块化设计方法。

结构化设计并不能对系统分析有帮助。当问题比较复杂，软件规模较大时，系统分析是必不可少的阶段。不论从用户角度还是从系统结构设计角度，都需要有一个逻辑模型定义系统的逻辑功能。

结构化分析是定义系统逻辑模型的一种方法学。结构化方法把软件开发过程分成六个阶段：调查、分析、设计、编码、测试和维护。在分析、设计和编码阶段均采用结构化的思想和工具进行软件的开发，如分析阶段的数据流图、数据字典、实体联系图和状态转移图等，设计阶段的软件结构图、层次图等，编码阶段的结构化编程等。

结构化方法的优点是：简单易学，易交流。

结构化方法的缺点是：它试图在系统建立之前对用户需求进行严格定义或预先加以明确说明，这通常是不切实际的；用户只参加软件开发的软件定义阶段的工作，不易发现开发中的问题，导致维护代价增加；静态的建模工具(文字和图形)缺乏直观的感性认识。

2. 快速原型法

原型是系统的早期版本，是系统的物理模型，只实现了系统的一些最基本的功能，反映系统的行为特性，但不一定满足全部需求。快速原型法是在结构化生命周期的编码阶段之前插入一个建立系统原型的阶段。

建立原型分四步进行：

第一步，确定用户的基本需求，而不是全部需求；

第二步，建立一个工作原型；

第三步，试用原型；

第四步，修改和补充原型。

原型要求快速建立，通常只有几周时间，所以称这种方法为快速原型法。快速原型法的优点如下：

(1) 容易理解和沟通，有一个可以“运行”的物理模型；

(2) 通过与原型交互，用户可以及早发现需求中的问题；

(3) 开发人员可以检查设计的可行性，可以在详细设计目标系统之前较容易地改正原型设计的问题。

总之，快速原型法缩短了软件开发周期，降低了开发和维护费用，也提高了用户的满意度。

快速原型法需要有快速地建立系统原型的工具，其中包括超高级语言。

3. 面向对象法

结构化方法和快速原型法使用的核心技术是结构分析与设计技术。结构化方法和快速原型法存在的缺陷是：软件的稳定性、可修改性和可复用性比较差。此缺陷产生的原因是：

(1) 结构分析与设计技术的本质是功能分解。

开发人员是围绕实现处理功能的“过程”来构造系统的，而用户需求的改动大部分是针对功能的，这必然引起软件结构的变化。

(2) 严格地定义了目标系统的边界。

很难把这样的系统扩展到新的边界，系统较难修改和扩充。

(3) 把处理分解成子处理的过程有些任意性。

不同的开发人员开发相同的系统时，可能经分解而得出不同的软件结构。

(4) 开发出的软件复用性较差，或不能实现真正意义上的软件复用。

基于上述种种因素，诞生了一种新的软件开发方法——面向对象法(object oriented，OO)。

面向对象法是指尽可能模拟人类习惯的思维方式，使软件开发方法与过程尽可能地接近人类认识世界解决问题的方法与过程。其最主要的特征之一是整个生命周期使用相同的概念、表示法和策略，即每一件事都围绕对象进行。

面向对象法有以下几种：

1) 面向对象分析(object oriented analysis，OOA)

OOA 是软件开发过程中的问题定义阶段。它从对问题的初始陈述开始，运用应用领域知识来识别该领域中的物理实体和概念，提取出对象，分析对象之间存在的相互关系，最后建立系统模型。

系统模型描述了系统的对象结构，是对问题论域精确、清晰的定义。

2) 面向对象设计(object oriented design，OOD)

OOD 决定如何将系统组织成子系统，每个子系统分成更小的子系统。较低层的子系统称为模块。一个遵循对话独立性原则的交互软件系统常分成用户界面和应用功能核心两个子系统。面向对象系统的控制结构方式可以是过程驱动、事件驱动或共行方式。

3) 面向对象程序设计(object oriented programming，OOP)

OOP 将 OOD 的结果用一种程序设计语言实现。通常总是选择一种面向对象的程序设计语言。

1.4 软件工程环境

软件开发手段经历了从手工编码到使用支撑软件产品的自动化软件工具的变迁。现在，从软件的开发、运行到维护，各阶段都有软件工具，这些工具形成了现代化软件工程环境的基础。软件工具是指可以用来帮助开发、测试、分析、维护其他计算机程序的程序以及文档资料的集合，它可以实现软件生产过程自动化，提高软件的生产率、可靠性，降低软件的生产成本。软件工具在各种状况下都能被简单、方便地使用，能给软件的开发带来极大的方便。大型软件生产所使用的软件工具是一种自动化系统，包括需求分析工具、设计工具、编码工具、确认工具、维护工具等。

需求分析工具能够辅助系统分析员把用户所提出的含糊的用户说明，经过分析及一致性、完备性检查后，快速生成指导系统设计用的需求规格说明书及其相应的文档资料。

设计工具能够依据输入的需求规格说明，自动设计出一系列软件设计文档，如软件结构说明、模块接口说明等。

编码工具的主要功能是输入设计阶段产生的文档，自动生成特定语言编制的程序，如各种应用程序生成器等。

尽管软件工具种类繁多，形式多样，但都只是用于软件生存周期中的某一个阶段或某一个环节，而不能对整个生命周期有效。为了能够对软件整个生命周期提供支持，于是出现了软件工程环境的新课题。软件工程环境(software engineering environment，SEE)是指用以支持需求定义、程序生成，以及软件维护等整个软件生命周期全部活动的，并把方法、规模和计算机程序集成在一起的整个体系。软件工程环境又称为软件开发环境、软件支撑环境、自动开发环境等。

软件工程环境的全部需求可以概括为：

(1) 应该是集成化的系统；

(2) 应该是通用的系统；

(3) 应该是既可剪裁又可扩充的系统；

(4) 应该是实用的、经济合算的系统。

近几年，软件工程领域出现了一种新趋势，即将软件工程方法、工具与环境方面的新技术同形式化语义理论有机地结合起来，形成高水平的计算机辅助软件工程(computer aided software engineering，CASE)系统，标志着软件开发技术的发展进入一个新阶段。CASE 系统可以帮助开发人员执行许多和软件开发有关的艰苦工作，包括对各种计划、合同、规格说明、设计、源代码和管理信息之类的文档进行组织。可以说，CASE 系统可以对软件生产过程的每一步提供辅助手段。

习　　题

1-1　什么是软件危机？软件危机的解决方法是什么？

1-2　什么是软件工程？

1-3　软件生命周期划分成哪些阶段？

1-4　传统的“瀑布模型”的主要缺陷是什么？试说明改进的方法。

1-5　什么是软件复用？软件复用包括哪几个层次？

1-6　什么是软件工程环境？软件工程环境的需求有哪些？

第2章 需求分析工程

需求分析是软件开发过程中最重要的阶段，如果不清楚系统“做什么”，也就谈不上“怎么做”。把需求当作一项工程，足见需求分析的重要。

2.1 需求分析工程概述

1. 问题的引出

软件危机引起人们对需求分析的重视。以下五个事实说明了这一点：

(1) 在软件生命周期中，一个错误发现越晚，修复错误的费用越高。

(2) 许多错误是潜伏的，且在错误产生后的很长一段时间才被检测出。

(3) 需求分析中会产生大量的错误。

(4) 需求分析中的错误多为疏忽、不一致和二义性。

(5) 需求错误是可以被检测出来的。

因此，有必要将需求过程上升为需求分析工程。

2. 需求分析工程的概念

需求是一个待开发软件中各个有意义陈述的集合，它必须是清晰的、简洁的、一致的和无二义性的。

需求分析工程是指应用已证实有效的原理、方法，通过合适的工具和记号，系统地描述待开发软件及其行为和相关约束。

需求分析工程的最终目标是得到待开发软件的系统模型，它必须是清晰的、易于理解的、一致的和无二义性的。

模型是对现实系统的描述，是现实系统的抽象和简化。原型是系统的早期版本，是系统的物理模型，只实现了系统的一些最基本的功能，反映系统的行为特性，但不一定满足全部需求。

3. 需求分析工程中的角色

需求分析工程会涉及三方面的人员，如图 2-1 所示。

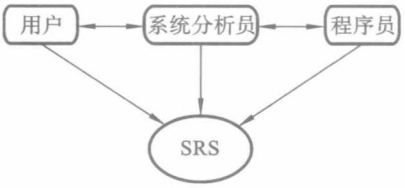


图 2-1 需求分析工程中的角色

(1) 需求方：对软件开发起决定作用的一方，可以是个人或企业等，是需开发软件者，不一定是最终用户。

(2) 系统分析方(系统分析员)：对待开发软件的需求进行详细描述，与开发方不一定是同一个企业(代理、趋势)。

(3) 开发方：构造系统者，如设计员、编程员和项目管理员。

2.2 需求分析工程的步骤

需求分析工程有四个步骤：

(1) 需求获取；

(2) 需求分析；

(3) 编写需求规格说明书(SRS)；

(4)验证。

需求分析工程即包括这四个方面的工作。

1. 需求获取

主要工作：收集信息(功能要求、非功能要求)、理解需求(弄清需求、澄清概念)、归纳整理(保留合理需求，抛弃不可能的需求)。

需求获取的困难之处：误解、交流障碍、缺乏共同语言、需求不完备、需求不稳定、用户意见不统一、错误的要求、认识混淆等，都会影响需求的获取。

解决方法：仔细研究需求，分析资料，深入进行市场调查，多与用户沟通，请教应用领域专家，考察现场等。

2. 需求分析

需求获取后，必须对需求进行分析。其目的是细化、精化软件的作用范围，确定软件的功能、性能、约束、环境等。从两个方面分析用户的需求：功能性需求和非功能性需求。

功能性需求是指系统必须完成的所有功能，而非功能性需求包括四个方面：性能要求、运行要求、未来要求、数据要求。性能要求是指如联机系统的响应时间、系统需要的存储容量以及系统的健壮性和安全性等方面的要求。运行要求是指系统运行所需要的软、硬件环境。未来要求是指系统将来可能的扩充要求、可复用性、可移植性等。数据要求是指系统所要处理的数据以及它们之间的联系。

3. 编写需求规格说明书(SRS)

需求分析工程最重要的结果是需求规格说明书。编写需求规格说明书的指导性原则如下：

(1) 从实现中分离功能，即描述“做什么”，不必描述“怎么做”；

(2) 要求有一个面向处理的系统规格说明语言，以描述系统级的动态行为；

(3) 必须对以该软件为元素的系统进行说明，以描述清楚系统各元素之间的关系；

(4) 必须对系统的运行环境进行说明，以保持系统接口描述的一致性；

(5) 必须是认识的模型而不是实现的模型，即必须以用户能够接受和理解的形式进行描述，将实际规则、条例组合到规格说明中；

(6) 必须是可操作的；

(7) 必须可容忍不完备性和可修改性；

(8) 必须局部化和松散耦合，使信息发生变化时只有唯一的一个片段(理想情况下)需要修改。

4. 验证

验证即是对需求分析工程的结果——SRS 进行评审，纠正错误，弥补缺陷，以保证 SRS 的质量。从以下几个方面评审：正确性、无二义性、完整性、可验证性、一致性、非计算机人员能理解、可修改性、可跟踪性、注释。

正确性是指 SRS 中对需求的描述与用户要求一致。无二义性是指 SRS 中陈述的事情有且仅有一种解释。完整性是指包含软件要做的全部事情，注明系统对有效和无效输入的反应，注明页码、图和表等的编号。可验证性是指存在技术和经济上可行的手段对需求进行验证和确认。一致性是指术语、特性和定时等的一致，不矛盾。非计算机人员能理解是指形式化和非形式化的矛盾。可修改性是指是否方便修改。可跟踪性是指每个需求的来源和流向清晰。注释是指向用户和设计者给出提示。

2.3 需求分析技术

需求分析技术有：①结构化分析技术(SAT)；②结构化分析与设计技术(SADT)；③面向对象技术(OOT)；④时序图；⑤有限状态机(FSM)；⑥Petri 网等。④、⑤、⑥用于(控制)系统动态分析。

2.3.1 时序图

时序图可以描述系统中处理事件的时序与相应的处理时间。图 2-2 中事件 e 被功能 1、功能 2 和功能 3 处理的时间共为 $T_1+T_2+T_3$，功能间的切换时间为 0。

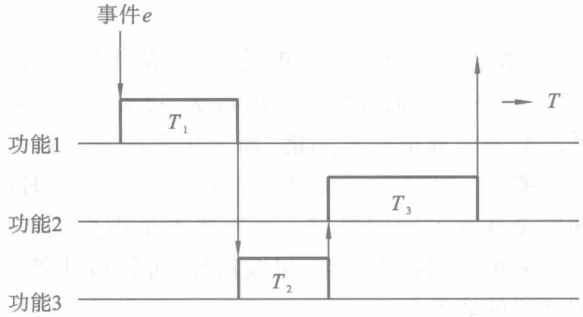


图 2-2 时序图例

图 2-3 采用扩充时序图表示进程间的通信流，可以用于分析几个事件的交错执行。做出如下分析：必须设计成 HOST1 在等待 C_1 的应答 R_1 期间要能够接收从 HOST2 发出的命令 C_2。

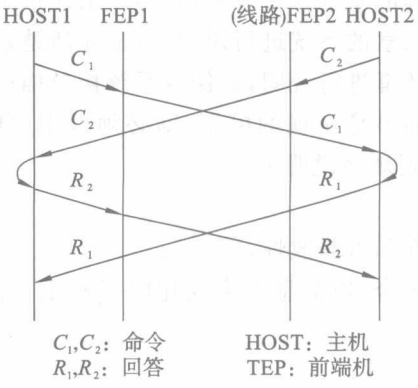


图 2-3 通信流图例