第一章

软件的概念和特点

概念:计算机软件是由专业人员开发并长期维护的软件产品。完整的软件产品包括了在各种不同容量和体系结构计算机上的可执行的程序，运行过程中产生的各种结果，以及以硬复制和电子表格等多种方式存在的软件文档。

特点:1.具有抽象性 2.无明显的制造过程 3.存在退化问题 4.对计算机系统有着不同程度的依赖性 5.尚未完全摆脱人工的开发方式 6.软件本身是复杂的 7.成本相当昂贵 8.相当多的软件工作设计社会因素

软件的分类

基于功能不同:系统软件，支撑软件，应用软件

根据服务对象的不同:通用软件，定制软件

按照软件产品的规模的不同:小型软件，中型软件，大型软件

根据工作的方式不同:实时软件，分时软件，交互式软件，批处理软件

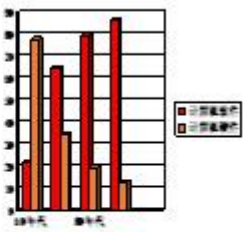
软件规模

软件危机的表现与原因

在软件开发的过程中，会经常出现一些不能按时完成任务、产品质量得不到保证、工作效率低下和开发经费严重超支等现象。计算机软件的开发、维护和应用过程中普遍出现的这一些严重的问题便是软件危机

举例:

- 80年代欧洲亚丽安娜火箭的发射失败，原因是软件错误。
- Windows vista，介于win xp和win7的操作系统，导致系统不稳定，存在大量的bug。
- 日本第5代机因为软件问题在投入50亿美元后于1993年下马。



主要表现:

1.产品的功能或特性与需求不符

2.相比硬件，软件代价过高

3.质量难以保证，难以发挥硬件潜能

4.难以准确估计开发，维护的费用和开发周期

5.难以控制开发风险，开发速度赶不上市场变化

6.软件产品修改,维护困难

7.软件文档不完备，存在内容与产品不符的情况

本质原因:人们对软件产品的认识的不足以及软件开发的内在规律理解的偏差

具体原因:

1.忽视开发前期的需求分析

2.开发过程缺乏统一,规范化的方法论指导

3.文档资料不齐全或不准确

4.忽视与用户之间，开发组成员之间的交流

5.忽视测试的重要性

6.不重视维护，或维护工作困难

7.对产业认识不充分，缺乏经验

8.没有完善的质量保证体系

软件工程概念

IEEE 对软件工程的定义为:

1. 将系统化，严格约束的，可量化的方法应用于软件的开发，运行和维护，即将工程化应用于软件
2. 对系统化，严格约束的，可量化的方法进行研究

具体来说：软件工程是以借鉴传统工程的原则,方法，以提高质量，降低成本为目的指导计算机软件开发和维护的工程学科，它是一种层次化的技术

目标:关注质量

三要素：过程，方法和工具

软件工程目标和原则

6 条基本目标:

1. 达到要求的软件功能
2. 取得较好的软件性能
3. 开发出高质量的软件
4. 付出较低的开发成本
5. 需要较低的维护费用
6. 能按时完成开发工作，及时交互使用

7 条基本原则:

1. 用分阶段的生命周期计划进行严格的管理
2. 检查进行阶段评审
3. 实行严格的版本控制
4. 采用现代程序设计技术
5. 结果应能清楚的审查
6. 开发小组的人员应该少而精
7. 承认不断改进软件工程实践的必要性

软件开发的方法

常见软件开发方法：

1. 结构化方法
2. 面向数据结构方法
3. 面向对象方法
4. 形式化方法
5. 问题分析法
6. 可视化开发方法等

软件工程工具三种分类标准

1. 按照功能划分
2. 按照支持的过程划分
3. 按照支持的范围划分

常识：

在需求分析与系统设计阶段，常用的 CASE（计算机辅助软件工程）工具有面向通用软件设计的 Microsoft Visio、用于面向对象软件设计的 Rational Rose、用于数据库设计的 Power Designer，除此之外近几年还出现了更加集成化的工具，如 Enterprise Architect、Rational Software Architect 和 StarUML 等。这些工具通过简化 UML 图的绘制工作，以及强大的模型转换功能（诸如正向工程、反向工程、数据库模型转化等），大大简化了设计以及从设计向编码转化的工作。

软件开发人员应该注意的标准

1. 保密
2. 能力
3. 知识产权
4. 计算机滥用

第二章

软件过程概述

软件的诞生和生命周期是一个过程，我们总体上称这个过程为软件过程。通常，使用生命周期模型简洁地描述软件过程。生命周期模型规定了把生命周期划分为哪些阶段及各个阶段的执行顺序，因此也称为过程模型。

软件生命周期

软件生命周期的概念

软件产品的生命周期是指从设计该产品的构想开始，到软件需求的确定、软件设计、软件实现、产品测试与验收、投入使用以及产品版本的不断更新，到最终该产品被市场淘汰的全过程。

软件生命周期这个概念从时间的角度将软件的开发和维护的复杂过程分解为了若干个阶段，每个阶段都完成特定的相对独立的任务。

传统软件生命周期的各个阶段

在传统的软件工程中，软件产品的生命周期一般可以划分为 6 个阶段。

1．可行性研究

它首先要确定待开发的软件产品所要解决的问题。同时需确定开发策略与方式，估计资金、时间和资源，并从技术、经济、操作等方面进行可行性分析，制订初步开发计划，完成可行性分析报告。

2．需求分析

需求是指为了解决用户提出的问题，目标系统需要做什么。需求分析直接关系软件开发的成败。开发人员需通过各种途径获取需求，与用户充分沟通，然后从功能、性能、界面和接口等方面详细描述，最终形成软件需求规格说明书。

3．软件设计

软件设计在需求分析基础上，将功能可操作化，分概要设计和详细设计两阶段。概要设计建立系统总体结构（结构、接口、全局数据结构）；详细设计关注模块内部实现细节，为编码提供依据。

4．编码

在编码阶段，开发人员根据设计阶段制订出的设计方案，编写程序代码。

5．软件测试

软件测试是保证软件质量的关键步骤。软件测试的目的是发现软件产品中存在的软件缺陷，进而保证软件产品的质量。

6．软件维护

软件交付后生命周期仍在继续。用户会不断发现隐藏错误，且随需求变化和市场环境变化，软件需不断更新升级。因此开发人员需持续维护产品，保证正常运行。

此外，从另一个角度来看，软件生命周期还可以划分为以下 8 个阶段，每个阶段涉及不同的人员角色：

1. 问题定义：要解决的问题是什么？（项目经理，客户经理，用户）
2. 可行性研究：有行得通的解决办法吗？（项目经理，系统架构师）
3. 需求分析：为了解决这个问题，目标系统必须做什么？（需求分析师）
4. 总体设计：概况地说，应该怎样实现目标系统？（系统架构师）
5. 详细设计：应该怎样具体地实现各个系统？（系统架构师，高级程序员）
6. 编码和单元测试：程序模块（程序员）
7. 综合测试：集成测试，验收测试（测试工程师，用户）
8. 软件维护：改正性维护，适应性维护，完善性维护，预防性维护（维护工程师）

软件过程模型

在软件工程中，人们通过建立抽象的软件过程模型，把软件生命周期中的各个活动或步骤安排到一个框架中，将软件开发的全过程清晰且直观地表达出来。

瀑布模型

瀑布模型是一种线性的开发模型，具有**不可回溯性**。开发人员必须等前一阶段的任务完成后，才能开始进行后一阶段的工作，并且前一阶段的输出往往就是后一阶段的输入。由于其不可回溯性，如果在软件生命周期的后期发现并要改正前期的错误，那么需要付出很高的代价。传统的瀑布模型是**文档驱动**的。

瀑布模型的优点是过程模型简单，执行容易；缺点是无法适应变更。瀑布模型适应于具有以下特征的软件开发项目：

1. 在软件开发的过程中，需求不发生或发生很少变化，并且开发人员可以一次性获取到全部需求。否则，由于瀑布模型较差的可回溯性，在后续阶段中需求经常性的变更需要付出高昂的代价。

2. 软件开发人员具有丰富的经验，对软件应用领域很熟悉。
3. 软件项目的风险较低。瀑布模型不具有完善的风险控制机制。

快速原型模型

快速原型的基本思想是快速建立一个能反映用户主要需求的原型系统，让用户在计算机上试用它，通过实践来了解目标系统的概貌。通常，用户试用原型系统之后会提出许多修改意见，开发人员按照用户的意见快速地修改原型系统，然后再次请用户试用……反反复复地改进，直到原型系统满足用户的要求。

快速原型模型适用于具有以下特征的软件开发项目：

1. 已有产品或产品的原型（样品），只需客户化的工程项目
2. 简单而熟悉的行业或领域
3. 有快速原型开发工具
4. 进行产品移植或升级

增量模型

增量模型将软件系统模块化，每个模块作为增量组件，分批次进行分析、设计、编码和测试。开发人员可分批次提交产品，而非一次性提交。

增量模型的最大特点就是将待开发的软件系统模块化和组件化。基于这个特点，增量模型具有以下优点：

1. 模块化后可分批提交，用户及时了解项目进展。
2. 以组件为单位开发，降低风险，单个周期错误不影响整个系统。
3. 开发顺序灵活，可按优先级排序，先完成核心组件，随时调整。

增量模型的缺点是要求待开发的软件系统可以被模块化。如果待开发的软件系统很难被模块化，那么将会给增量开发带来很多麻烦。

增量模型适用于具有以下特征的软件开发项目：

1. 软件产品可以分批次地进行交付
2. 待开发的软件系统能够被模块化
3. 软件开发人员对应用领域不熟悉，难以一次性地进行系统开发
4. 项目管理人员把握全局的水平较高

螺旋模型

螺旋模型是一种用于风险较大的大型软件项目开发的过程模型。该模型将瀑布模型与快速原型模型结合起来，并且加入了这两种模型忽略了的风险分析。它把开发过程分为**制定计划、风险分析、实施工程和客户评估** 4 种活动。

螺旋模型适应于风险较大的大型软件项目的开发。它的优点是将风险分析扩展到各个阶段中，大幅度降低了软件开发的风险。但是这种模型的控制和管理较为复杂，可操作性不强，对项目管理人员的要求较高。

喷泉模型

喷泉模型是一种过程模型，同时也支持面向对象开发。“喷泉”一词体现了面向对象方法的迭代和无间隙性。迭代是指各阶段需要多次重复，例如，分析和设计阶段常常需要多次、重复进行，以更好的实现需求。无间隙性是指各个阶段之间没有明显的界限，并常常在时间上互相交叉，并行进行。

喷泉模型主要用于面向对象的软件项目，软件的某个部分通常被重复多次，相关对象在每次迭代中随之加入渐进的软件成分。

基于组件的开发模型

基于组件的开发模型使用现有组件和系统框架进行开发。确定需求后，从组件库筛选合适组件，分析后可修改需求或组件。然后用成熟框架复用组件，无法利用的地方单独开发，最后集成测试。新组件经考验后也可加入组件库。

基于组件的开发模型充分的体现了软件复用的思想，降低了开发成本和风险，并加快了产品开发。

统一软件开发过程模型

统一软件开发过程（Rational Unified Process，RUP）模型是基于 UML（统一建模语言）的一种面向对象软件开发模型。它解决了螺旋模型的可操作性问题，采用迭代和增量递进的开发策略，并以用例驱动为特点，集中了多个软件开发模型的优点。RUP 模型是迭代模型的一种。

RUP 基于迭代思想，系统由软件构件建造而成，构件定义明确接口相互连接。RUP 采用架构优先策略，架构包含系统最重要的静态和动态特征，体现总体设计。

敏捷过程与极限编程

随着技术发展和市场竞争加剧，软件需求常变，传统方法在时效上面临挑战，敏捷开发方法因此流行。敏捷方法是一种轻量级方法，强调变化的必然性，通过团队交流和合理机制有效响应变化。

"敏捷联盟"制定的 12 条原则:

1. 通过尽早和持续交付有价值的软件来让客户满意
2. 需求变更可以发生在整个软件的开发过程中

3. 经常交付可工作的软件
4. 业务人员和开发人员应该天天在一起工作
5. 围绕受激励的个人构建项目
6. 在团队的内部，最有效果和效率的信息传递方法是面对面交谈
7. 可工作的软件是进度的首要度量标准
8. 敏捷过程提倡可持续的开发速度
9. 不断地关注优秀的技能和良好的设计会增强敏捷能力
10. 尽量使工作简单化
11. 好的架构、需求和设计来源于自身组织团队
12. 每隔一定时间，团队应该反省如何才能有效地工作，并相应地调整自己的行为

敏捷模型包括多种实践方法，比如：

1. 极限编程（eXtreme Programming，XP）
2. 自适应软件开发（Adaptive Software Development，ASD）
3. 动态系统开发方法（Dynamic System Development Method，DSDM）
4. Scrum
5. Crystal
6. 特征驱动开发（Feature Driven Development，FDD）

极限编程（XP）强调用户需求和团队工作，即使在开发末期也能快速响应需求变化。适用于需求模糊易变、团队少于 10 人、地点集中的场合。

代码的质量在极限编程项目中非常重要。为了保证代码的质量，可以采用结对编程以及在编码之前构造测试用例等措施。

几种模型之间的关系

1．瀑布模型与统一软件开发过程 RUP 模型之间的关系

在宏观上，瀑布模型是静态模型，RUP 模型是动态模型。RUP 模型的每一次迭代，实际上都需要执行一次瀑布模型，都要经历先启、细化、构建、产品化这 4 个阶段，完成瀑布模型的整个过程。瀑布模型中有 RUP 模型，反过来，RUP 模型中也有瀑布模型。

2．瀑布模型与增量模型之间的关系

增量模型中每开发一个模块通常采用瀑布模型。即宏观上是增量模型，微观上是瀑布模型。每增加一个模块就迭代一次，所以增量模型本质上是迭代的。

3．瀑布模型与快速原型模型之间的关系

快速原型的基本思想是快速建立一个能反映用户主要需求的原型系统，在此基础上之后的每一次迭代，都可能会用到瀑布模型。快速原型模型中不但包含了迭代模型的思想，而且包含了瀑布模型的思想。

4．瀑布模型与螺旋模型之间的关系

螺旋模型是瀑布模型和快速原型模型的结合，快速原型模型是原型模型的简化，原型模型又是迭代模型和瀑布模型的组合，这些模型之间是相互依存的、彼此有关的。螺旋模型每一次顺时针方向旋转，相当于顺时针方向迭代一次，都是走完一次瀑布模型，这就是瀑布模型与螺旋模型之间的关系。实际上，瀑布模型与喷泉模型也有关系。

选择软件过程模型

各种模型反映了生命周期的多样性，不同阶段可采用不同模型。开发时可选某种模型，配合相应开发方法和工具。

在选择软件过程模型时需要考虑以下几点：

1. 符合软件自身的特性，如规模、成本和复杂性等
2. 满足软件开发进度的要求
3. 对软件开发的风险进行预防和控制
4. 具有计算机辅助工具的支持
5. 与用户和软件开发人员的知识和技能相匹配
6. 有利于软件开发的管理和控制

一般来说，结构化方法和面向数据结构方法可采用瀑布模型或增量模型进行软件开发；而面向对象方法可采用快速原型模型、喷泉模型或 RUP 模型进行软件开发。

不同组织可能选择不同模型，但基本目标一致：满足用户的基本功能需求。

本章小结

本章介绍了软件过程的基本概念、软件生命周期的各个阶段以及各种常见的软件过程模型。下表对各种软件过程模型的优点、缺点和适用范围进行了总结对比：

模型	优点	缺点	适用范围
瀑布模型	有利于大型软件开发过程中的组织管理	开发过程一般不能逆转，很难严格按该模型进行	需求非常清楚全面，且没有或很少变化
快速原型	可以得到比较良好的需求定义，容易适应需求的变化	客户与开发者对原型理解不同；原型设计比较困难	对所开发的领域比较熟悉而且有快速的原型开发工具
增量模型	人员分配灵活，不用投入大量人力资源	可能遇到不能集成的风险，软件必须具备开放式的体系结构	进行已有产品升级或新版本开发，对所开发的领域比较熟悉

模型	优点	缺点	适用范围
螺旋模型	设计上的灵活性，可以在项目的各个阶段进行变更	需要具有相当丰富的风险评估经验和专门知识，过多的迭代次数会增加开发成本	大规模的软件项目
敏捷模型	很快可以看到一个基线架构版的产品，注重市场快速反应	注重人员的沟通，忽略文档的重要性，给维护带来不少难度	中小型项目和小型项目组
喷泉模型	各个阶段没有明显的界限，开发人员可以同步进行开发	各阶段重叠，需要大量的开发人员，因此不利于项目管理。此外这种模型要求严格管理文档	面向对象的软件开发

思考题

假设你被任命为一家软件公司的项目负责人，你的工作是管理该公司已被广泛应用的字处理软件的新版本开发。由于市场竞争激烈，公司规定了严格的完成期限并且已对外公布。你打算采用哪种软件生命周期模型？为什么？

参考答案：答：应采用增量模型。原因：（1）字处理软件是成熟产品，可按功能模块化（如编辑、排版、打印、拼写检查等）；（2）市场竞争激烈且期限严格，增量模型可分批交付核心功能，抢占市场；（3）已有产品的经验积累可降低开发风险；（4）增量模型允许优先开发最关键的功能模块，灵活调整开发顺序。

第四章

项目立项概述

任何一个完整的软件工程项目都是从项目立项开始的。项目立项包括项目发起、项目论证、项目审核和项目立项四个过程。

在发起一个项目时，项目发起人或单位为寻求他人的支持，要以书面材料的形式递交给项目的支持者和领导，使其明白项目的必要性和可行性。

项目论证过程，也就是可行性研究过程。可行性研究就是指在项目进行开发之前，根据项目发起文件和实际情况，对该项目是否能在特定的资源、时间等制约条件下完成做出评估，并且确定它是否值得去开发。

项目经过可行性研究并且认为可行后，还需要报告主管领导或单位，以获得项目的进一步审核，并得到他们的支持。

项目通过可行性研究和主管部门的批准后，将其列入项目计划的过程，叫做项目立项。

经过项目发起、项目论证、项目审核和项目立项四个过程后，一个软件工程项目就正式启动了。

可行性研究

可行性研究的任务

可行性研究需要从多个方面进行评估，主要包括:

1. 战略可行性
2. 操作可行性
3. 计划可行性
4. 技术可行性
5. 社会可行性
6. 市场可行性
7. 经济可行性
8. 风险可行性

技术可行性

技术可行性主要研究待开发的系统的功能、性能和限制条件，确定现有技术能否实现有关的解决方案，在现有的资源条件下实现新系统的技术风险有多大。

在评估技术可行性时，需要考虑以下情况：了解当前最先进的技术，分析相关技术的发展是否支持新系统；确定资源的有效性，如新系统的软硬件资源是否具备，开发项目的人员在技术和时间上是否可行等；分析项目的开发的技术风险，即能在给定的资源和时间等条件下，设计并实现系统的功能和性能等。

操作可行性

操作可行性衡量系统在给定工作环境中能否运行及运行好坏程度。需考虑政治、法律、道德、组织机构等因素。操作可行性最容易被忽视或低估。

经济可行性

成本-效益分析是可行性研究的重要内容，它用于评估基于项目的经济合理性，给出项目开发的成本论证，并将估算的成本与预期的利润进行对比。

项目成本由 4 部分组成：软硬件设备购置费、项目开发费、安装运行维护费、人员培训费。分析设计阶段为估算成本，交付后统计为实际成本。

项目开发效益包括经济效益和社会效益两部分。经济效益是指所使用系统为用户增加的收入，可以通过直接的或统计的方法估算；社会效益只能用定性的方法估算。

1．成本估算

代码行技术：将功能成本与源代码行数联系，用每行代码的平均成本乘以行数确定成本。平均成本取决于软件复杂度和人员薪资。

任务分解技术。首先将开发项目分解为若干个相对独立的任务，再分别估算每个任务单独开发的成本，最后累加起来就可得出开发项目的总成本。

2．成本-效益分析

1. 开发成本：使用代码行技术或任务分解技术进行估算
2. 运行费用：取决于系统操作的费用（涉及操作人员、工作时间和消耗物资等），以及维护费用
3. 经济效益：因使用新系统而增加的收入加上使用新系统可以节省的运行费用

3．货币的时间价值

通常使用利率的形式表示货币的时间价值。假设年利率为 i，如果现在存入 P，则 n 年后可以得到的价值为：**F=P(1+i)^n**。F 就是 P 在 n 年后的价值。反之，如果 n 年后能收入 F，那么这些货币的现在价值就是：**P=F/(1+i)^n**。

4．投资回收期

投资回收期是衡量一个项目价值的常用方法。投资回收期就是使**累计的经济效益等于最初投资所需要的时间**。很明显，投资回收期越短，所获得的利润就越快，因此该项目就

越值得开发。

5．纯收入

纯收入是衡量一个项目价值的另一项经济指标。纯收入就是在软件生命周期中软件系统的**累计经济效益（折合成现在值）与投资之差**。这相当于比较投资开发一个软件系统和将钱存在银行中（或贷给其他企业）这两种方案的优劣。

可行性研究的步骤

进行可行性研究的步骤不是固化的，而是根据项目的性质、特点以及开发团队的能力有所区别。一个典型的可行性研究的步骤可以归结为以下几步。

1．明确系统的目标

访问相关人员，分析材料，确认问题实质，明确系统目标及所需资源。

2．分析研究现行系统

现行系统是新系统的重要信息来源。可从 3 方面分析：组织结构（组织结构图）、处理流程（系统流程图）、数据流（数据流图和数据字典）。

3．设计新系统的高层逻辑模型

这一步从较高层次设想新系统的逻辑模型，概括地描述开发人员对新系统的理解和设想。

4．获得并比较可行的方案

开发人员可根据新系统的高层逻辑模型提出实现此模型的不同方案。在设计方案的过程中要从技术、经济等角度考虑各方面的可行性。然后，从多个方案中选择出最合适的方案。

5．撰写可行性研究报告

可行性研究的最后一步就是撰写可行性研究报告。此报告包括项目介绍、可行性分析过程和结论等内容。

可行性研究的结论一般有以下三种:

1. 可以按计划进行软件项目的开发
2. 需要解决某些存在的问题（如资金短缺、设备陈旧和开发人员短缺等）或者需要对现有的解决方案进行一些调整或改善后才能进行软件项目的开发
3. 待开发的软件项目不具有可行性，立即停止该软件项目

制定项目开发计划

可行性研究后若项目值得开发，则应制订项目开发计划。计划涉及项目各环节，其合理

性关系项目成败。计划应考虑周全（含未知和不确定因素）并尽量准确。

软件开发计划是一种管理文档，对人员、进度、费用及环境配置进行说明规划，是管理人员控制费用、进度和资源的依据。

项目开发计划的主要内容如下：

1. 项目概述：说明项目的各项主要工作；说明软件的功能和性能；为完成项目应具备的条件；甲方和乙方应承担的工作、完成期限和其他限制条件；应交付的软件名称，所使用的开发语言及存储形式；应交付的文档等。
2. 实施计划：说明任务的划分，各项任务的责任人；说明项目开发进度，按阶段应完成的任务，用图表说明每项任务的开始时间和完成时间；说明项目的预算，各阶段的费用支出预算等。
3. 人员组织及分工：说明开发该项目所需人员的类型、组成结构和数量等。
4. 交付期限：说明项目应交付的日期等。

第五章

需求分析

需求分析概述

需求分析是指开发人员要准确理解用户的需求，将用户的需求用软件工程开发语言表达出来，产生需求规格说明书的过程。需求分析是软件生命周期中重要的一步，也是最关键的一步。

需求分析的基本任务是要准确地定义新系统的目标，回答系统必须"做什么"的问题。

需求获取

需求获取是需求分析的第一步，目标是通过各种途径获取用户的需求。常用的需求获取方法包括：

1. 访谈：开发人员与用户面对面交流，了解用户的需求和期望
2. 问卷调查：通过发放问卷收集大量用户的意见和需求
3. 观察：开发人员到用户的工作现场，观察用户的实际工作流程
4. 文档分析：分析现有的文档、报表、表单等，了解系统的输入输出

需求建模

需求建模是需求分析的第二步，目的是将获取的需求用模型来表示，便于分析和验证。需求建模需要建立 3 种模型：

1. 功能模型：描述系统"做什么"，用数据流图表示
2. 数据模型：描述系统中的数据对象及其关系，用实体-关系图表示
3. 行为模型：描述系统响应外部事件的行为，用状态转换图表示

需求验证

需求验证是需求分析的第四步，目的是验证需求分析成果的正确性，确保需求的一致性、完整性、现实性和有效性。

需求评审是将需求规约文档发布给利益相关者进行检查，发现需求规约中存在缺陷的过程。评审分为正式技术评审（同行评审）和非正式技术评审两类。对于任何重要的工作产品，都应该至少执行一次正式技术评审。

需求管理

需求管理是一种用于查找、记录、组织和跟踪系统需求变更的系统化方法。需求管理实际上是项目管理的一个部分，它涉及以下 3 个主要问题：

1. 识别、分类、组织需求，并为需求建立文档
2. 需求变化：建立对需求不可避免的变化如何提出、协商、验证以及形成文档的过程
3. 需求的可追踪性：带有维护需求之间以及与系统的其他制品之间依赖关系的过程

需求分析的常用方法

需求分析的方法有多种，下面简单介绍 4 种常用方法。

（1）功能分解方法将一个系统看成由若干功能模块组成，每个功能又可分解为若干子功能及接口，子功能再继续分解。功能、子功能和功能接口是功能分解方法的 3 个要素，采用自顶向下、逐步求精的理念。

（2）结构化分析方法一种从问题空间到某种表示的映射方法，其逻辑模型由数据流图和数据词典构成。它是一种面向数据流的需求分析方法，主要适用于数据处理领域问题。

（3）信息建模方法常用的基本工具是 E-R 图，基本要素由实体、属性和关系构成。核心概念是实体和关系，基本策略是从现实中找出实体，再用属性对其进行描述。

（4）面向对象的分析方法关键是识别问题域内的对象，分析它们之间的关系，并建立 3 类模型：描述系统静态结构的对象模型、描述系统控制结构的动态模型、描述系统计算结构的功能模型。

结构化分析概述

结构化分析方法（Structured Analysis，简称 SA 法）是 70 年代由 Yourdon、Constaintine 及 DeMarco 等人提出和发展的，是一种面向数据流的需求分析方法。它基于"分解"和"抽象"的基本思想，逐步建立目标系统的逻辑模型，进而描绘出满足用户要求的软件系统。

"分解"是指对于一个复杂的系统，为了将复杂性降低到可以掌握的程度，可以把大问题分解为若干个小问题，然后再分别解决。分解可以分层进行：

1. 最顶层描述了整个目标系统
2. 中间层将目标系统划分为若干个模块，每个模块完成一定的功能
3. 最底层是对每个模块实现方法的细节性描述

结构化分析方法

功能建模

功能建模的思想是用抽象模型的概念，按照软件内部数据传递和变换的关系，自顶向下逐层分解，直到找到满足功能要求的可实现的软件为止。功能模型用数据流图来描述。

数据流图（简称 DFD 图）采用图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具。

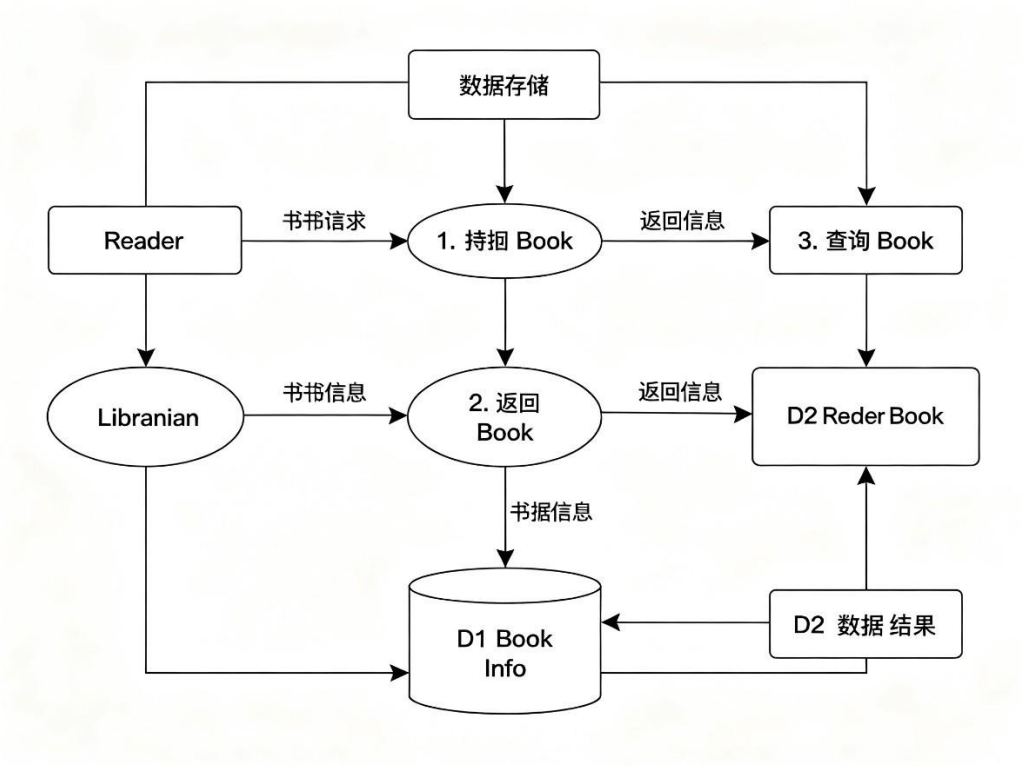


图 5-1 数据流图（DFD）示例——图书管理系统

在数据流图中，存在 4 种表示符号：

1. 外部实体：表示数据的源点或终点，是系统之外的实体，可以是人、物或其他系统
2. 数据流：表示数据的流动方向，可以从加工流向加工、从加工流向文件、从文件流向加工
3. 数据变换（加工）：表示对数据进行加工或处理
4. 数据存储：表示输入或输出文件，可以是计算机系统中的外部或内部文件

根据"自顶向下，由外到内，逐层分解"的思想，开发人员先画出系统顶层的数据流图，然后再逐层画出低层的数据流图。顶层的数据流图定义系统范围，描述系统与外界的数据联系。

环境图（也称系统顶层数据流图或 0 层数据流图）仅包括一个数据处理过程，即要开发的目标系统。其作用是确定系统在其环境中的位置，通过确定系统的输入和输出与外部实体的关系确定其边界。

数据建模

数据建模的思想是在较高的抽象层次（概念层）上对数据库结构进行建模。数据模型用实体-关系图来描述。

E-R 图以实体、关系和属性 3 个基本概念概括数据的基本结构：

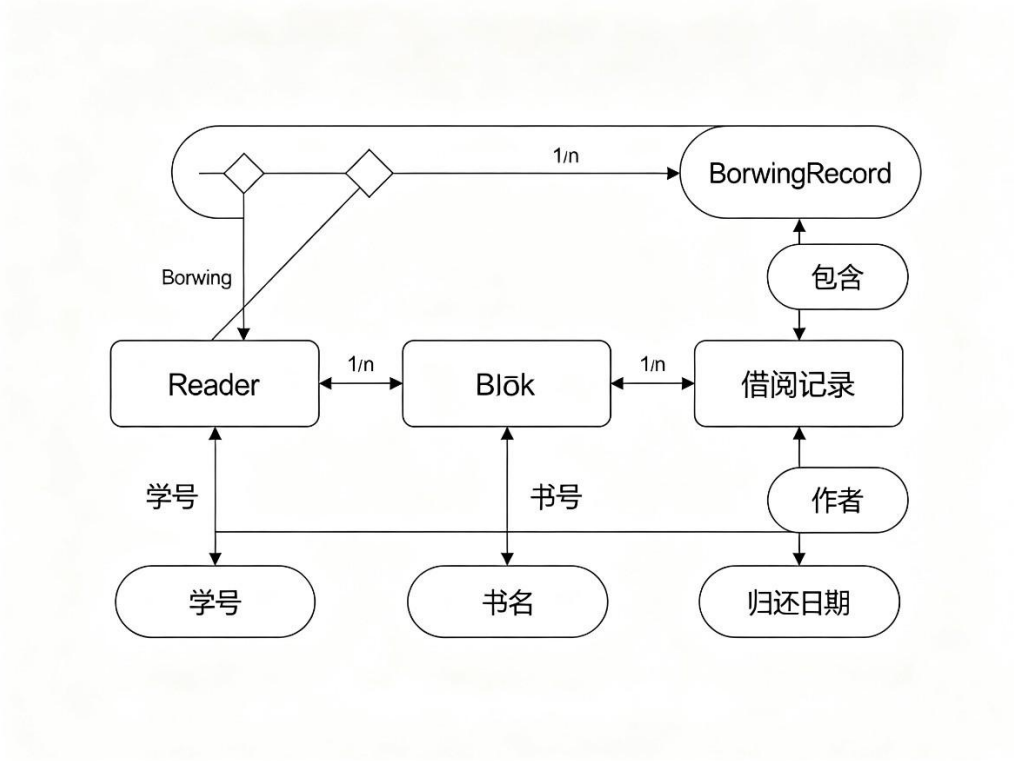


图 5-2 实体-关系图（E-R 图）示例——图书管理系统

1. 实体：现实世界中的事物，用矩形框表示，框内含实体名称
2. 属性：用椭圆形表示，用无向边与相应实体联系，实体由若干属性组成
3. 关系：用菱形表示，用无向边与有关实体连接

实体之间存在 3 种关系类型：一对一（1:1）、一对多（1:n）和多对多（m:n）。

行为建模

状态转换图是一种描述系统对内部或外部事件响应的行为模型。它描述系统状态和事件，事件引发系统在状态间的转换，而不是描述系统中数据的流动。这种模型尤其适合用来描述实时系统。

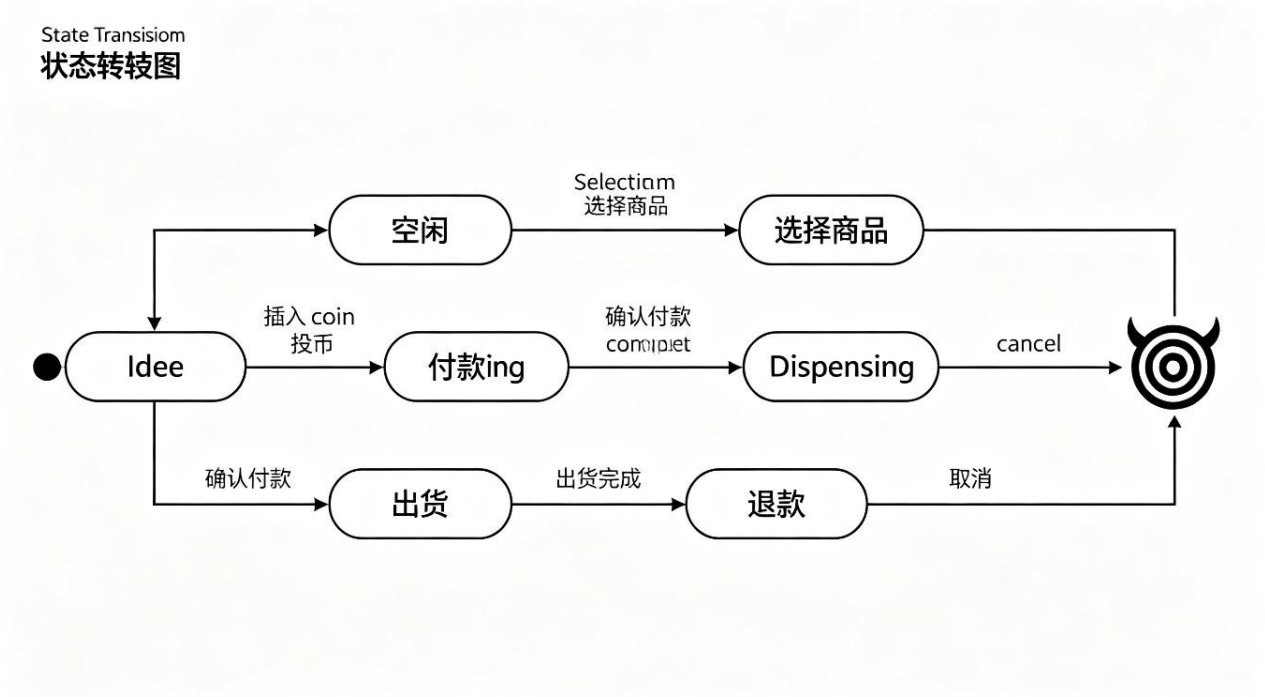


图 5-3 状态转换图示例——自动售货机

使用状态转换图具有以下优点：

1. 状态之间的关系能够直观地被捕捉到
2. 因其单纯性，能够机械地分析许多情况，可容易地建立分析工具
3. 能够很方便地对应状态转换表等工具

状态

状态是任何可以被观察到的系统行为模式，一个状态代表系统的一种行为模式。状态规定了系统对事件的响应方式。在状态转换图中定义的状态主要有：初态、终态和中间状态。一张状态图中只能有一个初态，终态可以没有也可以有多个。

事件

事件是在某个特定时刻发生的事情，它是对引起系统做动作或（和）从一个状态转换到另一个状态的外界事件的抽象。状态变迁通常由事件触发，应在表示状态转换的箭头线上标出触发转换的事件表达式。如果未标明事件，则表示在源状态的内部活动执行完后自动触发转换。

数据字典

分析模型包括功能模型、数据模型和行为模型。数据字典以一种系统化的方式定义在分析模型中出现的数据对象及控制信息的特性，给出它们的准确定义，包括数据流、数据存储、数据项、数据加工，以及数据源点、数据汇点等。数据字典是将分析模型中 3 种模型黏合在一起的"黏合剂"，是分析模型的"核心"。

数据字典中常用的符号：

1. = ： 等价于（或由...组成）
2. + ： 与（连接两个数据元素）
3. [|] ： 或（选择括号内的某一项）
4. { } ： 重复（括号内的内容可重复出现）
5. ( ) ： 可选（括号内的内容可有可无）

本章小结

本章介绍了需求分析的基本概念和常用方法，重点介绍了结构化分析方法。结构化分析方法以"分解"和"抽象"为核心思想，通过功能建模（数据流图）、数据建模（实体-关系图）和行为建模（状态转换图）三种模型来描述系统，数据字典作为核心将三种模型联系起来。

思考题

1. 数据流图有哪几种基本符号？各表示什么含义？

参考答案：答：数据流图有 4 种基本符号：（1）外部实体（矩形）：表示数据的源点或终点，是系统之外的实体；（2）数据流（箭头）：表示数据的流动方向；（3）数据变换/加工（圆形或圆角矩形）：表示对数据进行加工或处理；（4）数据存储（双横线或开口矩形）：表示数据的存储文件。

2. 实体-关系图的基本要素是什么？实体之间的关系有哪几种类型？

参考答案：答：E-R 图的 3 个基本要素是：实体（矩形）、属性（椭圆）和关系（菱形）。实体之间的 3 种关系类型为：一对一（1:1）、一对多（1:n）和多对多（m:n）。

3. 状态转换图中的状态和事件分别是什么含义？

参考答案：答：状态是任何可以被观察到的系统行为模式，一个状态代表系统的一种行为模式，规定了系统对事件的响应方式。状态分为初态、终态和中间状态，一张状态图只能有一个初态，终态可以没有或有多个。事件是在某个特定时刻发生的事情，是引起系统做动作或从一个状态转换到另一个状态的外界事件的抽象。

4. 数据字典的作用是什么？常用的符号有哪些？

参考答案：答：数据字典的作用是以系统化的方式定义分析模型中出现的所有数据对象及控制信息的特性，给出准确定义，是将功能模型、数据模型和行为模型三种模型黏合在一起的"黏合剂"，是分析模型的"核心"。常用符号：=（等价于）、+（与）、[|]（或）、{}（重复）、()（可选）。

第六章

软件设计的基本概念

完成了需求分析，回答了软件系统能"做什么"的问题，软件的生命周期就进入了设计阶段。软件设计是软件开发过程中的核心阶段，开发人员将需求规格说明书中的分析模型转换为可行的设计模型。软件设计的目标就是要回答"怎么做"才能实现软件系统的问题。

软件设计的意义和目标

良好设计的 3 个特征：

1. 设计必须实现所有包含在分析模型中的明确需求，而且必须满足用户期望的所有隐含需求
2. 对于程序员、测试人员和维护人员而言，设计必须是可读的、可理解的指南
3. 设计必须提供软件的全貌，从实现的角度说明数据域、功能域和行为域

软件设计的原则

1．模块化模块化就是把系统或程序划分为独立命名并且可以独立访问的模块，每个模块完成一个特定的子功能。模块集成起来构成一个整体，完成特定功能。在模块化过程中要注意：模块规模适中、提高模块独立性降低耦合、提高内聚程度、加强模块保护性。

2．抽象抽象在软件开发中起着重要作用。一个庞大复杂的系统可以先用宏观概念构造和理解，然后逐层用微观概念去解释上层概念，直到最底层元素。

3．逐步求精按照逐步求精的思想，程序的体系结构按照层次结构，逐步精化过程细节而开发出来。求精就是细化，它与抽象是互补的概念。

4．信息隐藏模块仅仅公开必须让外界知道的信息，而将其他信息隐藏起来，模块的具体实现细节相对于其他不相关模块而言是不可见的。通常通过接口实现，模块通过接口与外部通信，把数据结构、算法等内部信息隐藏起来。

5．复用性设计软件复就是将已有的软件成分用于构造新的软件系统。可被复用的软件成分称作可复用构件。软件复用不仅包括程序复用，还包括开发计划、可行性研究报告、分析模型、设计模型、源程序、测试用例等的复用。

6．灵活性设计灵活性设计是软件面对需求修改时的随机应变能力。主要策略包括：降低耦合并提高内聚、建立抽象（继承和多态）、不要将代码写死（消除常数）、抛出异常、使用并创建可复用代码。

软件设计的分类

从工程管理角度，软件设计分为概要设计和详细设计：

1）概要设计（总体设计）：确定软件的结构及各组成部分之间的相互关系。以需求规格说明书为基础，概要说明软件系统的实现方案，包括目标系统总体架构、每个模块的功能描述和数据接口描述及模块间调用关系、数据库和数据结构等。

2）详细设计：确定模块内部的算法和数据结构，产生描述各模块程序过程的详细文档。涉及过程设计（描述每个模块的实现算法和细节）、数据设计（细化各模块所用数据结构）和接口设计（模块间关系或通信方式以及目标系统与外部系统的联系）。

结构化软件设计概述

结构化软件设计的任务是从软件需求规格说明书出发，设计软件系统的整体结构、确定每个模块的实现算法以及编写具体代码，形成软件的具体设计方案，解决"怎么做"的问题。

在结构化设计中，概要设计阶段将软件需求转化为数据结构和软件系统结构，完成体系结构设计、数据设计及接口设计。详细设计阶段完成过程设计，详细设计每个模块，确定完成每个模块功能所需的算法和数据结构。

结构化设计与结构化分析的关系

要进行结构化设计，必须依据结构化分析的结果。结构化软件设计的具体步骤如下：

1. 从需求分析阶段的数据流图出发，制订几个方案，从中选择合理的方案
2. 采用某种设计方法，将一个复杂的系统按功能划分成模块化的层次结构
3. 确定每个模块的功能、模块间的调用关系，建立与已确定的软件需求的对应关系
4. 系统接口设计，确定模块间的接口信息
5. 数据结构及数据库设计，确定实现软件的数据结构和数据库模式
6. 依据分析模型中的处理规格说明、状态转换图及控制规格说明进行过程设计
7. 制订测试计划
8. 撰写软件设计文档

体系结构设计

结构化软件工程方法关注系统功能，采用自顶向下、逐步求精的设计过程，以模块为中心解决问题。结构化设计方法分为面向数据流的设计方法和面向数据结构的设计方法。

表示软件结构的图形工具

层次图和 HIPO 图：通常使用层次图描绘软件的层次结构。一个矩形框代表一个模块，

框间连线表示调用关系。带编号的层次图称为 HIPO（Hierarchy Input-Process-Output）图。

面向数据流的设计方法

面向数据流的设计方法多在概要设计阶段使用，依据一定的映射规则，将数据流图转换为目标系统的结构描述。数据流分为变换型数据流和事务型数据流两种：

变换流：信息沿输入路径流入系统，经过加工处理后离开系统。变换是把输入的数据处理后转变成另外的输出数据。

事务流：非数据变换的处理，将输入的数据流分散成许多数据流，形成若干个加工，然后选择其中一个路径来执行。

针对变换型数据流的设计步骤：

1. 区分变换型数据流中的输入数据、变换中心和输出数据，在数据流图上用虚线标明分界线
2. 分析得到系统的初始结构图
3. 对系统结构图进行优化

针对事务型数据流的设计步骤：

1. 确定以事务为中心的结构，找出事务中心、接收数据、处理路径三个部分
2. 将数据流图转换为初始的系统结构图
3. 分解和细化接收分支和处理分支

接口设计

软件系统逐步分解到模块后，模块与模块之间必须根据各模块的功能定义对应的接口。接口设计一般包括 3 个方面：

（1）用户接口：说明将向用户提供的命令和它们的语法结构以及软件回答信息。

（2）外部接口：说明本系统同外界的所有接口的安排，包括软件与硬件之间的接口、本系统与各支持软件之间的接口关系。

（3）内部接口：说明本系统之内各个系统元素之间的接口的安排。

用户界面设计

用户界面设计是接口设计的组成部分。对于交互式系统，用户界面设计与数据设计、体系结构设计、过程设计一样重要。用户界面的设计质量直接影响用户对软件产品的评价，从而影响软件产品的竞争力和寿命。

过程设计

程序流程图

程序流程图（也称程序框图）是一种直观、形象地描述过程控制流程的图形工具。它包含 5 种基本的控制结构：

1. 顺序型
2. 选择型
3. 先判定型循环（WHILE-DO）
4. 后判定型循环（DO-WHILE）
5. 多分支选择型

软件设计评审

一旦所有模块的设计文档完成之后，就可以对软件设计进行评审。在评审中，应着重评审软件需求是否得到满足，软件结构的质量、接口说明、数据结构说明、实现和测试的可行性和可维护性等。此外，还应确认该设计是否覆盖了所有已确定的软件需求，软件设计成果的每一组成部分是否可追踪到某项需求，即满足需求的可追踪性。

本章小结

本章介绍了结构化设计的基本概念、原则和方法。软件设计分为概要设计和详细设计，遵循模块化、抽象、逐步求精、信息隐藏等原则。结构化设计以数据流图为依据，通过变换分析和事务分析将数据流图映射为软件结构。接口设计包括用户接口、外部接口和内部接口。过程设计使用程序流程图等工具描述模块内部的算法。

思考题

1. 软件设计的原则有哪些？分别简述其含义。

参考答案：答：软件设计的原则包括：（1）模块化：将系统划分为独立命名且可独立访问的模块；（2）抽象：用宏观概念构造系统，逐层用微观概念解释；（3）逐步求精：按层次结构逐步精化过程细节；（4）信息隐藏：模块只公开必要信息，内部细节对外界不可见；（5）复用性设计：将已有软件成分用于构造新系统；（6）灵活性设计：软件面对需求修改时的随机应变能力。

2. 变换型数据流和事务型数据流有什么区别？各自的设计步骤是什么？

参考答案：答：变换流是信息沿输入路径流入系统，经过加工处理后离开系统，强调数据的变换。设计步骤：区分输入/变换中心/输出→得到初始结构图→优化。事务流是将输入数据流分散成若干加工，选择其中一条路径执行，强调选择分发。设计步骤：确定

事务中心/接收数据/处理路径→转换为初始结构图→分解细化接收分支和处理分支。

3. 接口设计包括哪 3 个方面?

参考答案：答：接口设计包括 3 个方面：（1）用户接口：向用户提供的命令及其语法结构和回答信息；（2）外部接口：系统与外界的接口安排，包括软硬件接口和支持软件接口；（3）内部接口：系统内各元素之间的接口安排。

4. 程序流程图有哪 5 种基本控制结构?

参考答案：答：程序流程图的 5 种基本控制结构为：（1）顺序型；（2）选择型；（3）先判定型循环（WHILE-DO）；（4）后判定型循环（DO-WHILE）；（5）多分支选择型。

第七章

面向对象的软件工程方法

面向对象的基本概念

面向对象是按照人们认识客观世界的系统思维方式，采用基于对象的概念建立模型，模拟客观世界分析、设计、实现软件的方法。对象指现实中各种各样的实体，对象的内部状态称为属性，运动规律称为方法或服务。类是指具有相似内部状态和运动规律的实体的集合。

面向对象的软件工程方法的特征与优势

面向对象方法的特征：把数据和操作封装在一起形成对象；把特征相似的对象抽象为类；类之间可以存在继承关系形成层次结构；对象之间通过发送消息进行通信；将对象的私有信息封装起来，外界通过有限的接口访问。

面向对象的实施步骤

面向对象分析：从问题陈述入手，分析和构造问题域的模型，明确必须做什么。步骤包括确定问题域、区分类和对象、区分整体对象及组成部分、定义属性、定义服务、确定附加的系统约束。

面向对象设计：对分析结果进行改进和完善。步骤包括设计交互过程和用户接口、设计任务管理、设计全局资源、对象设计。

面向对象实现：使用面向对象语言实现设计。面向对象测试：包括模型测试、类测试、交互测试、系统测试、验收测试等。

统一建模语言 UML

UML 简述

统一建模语言（UML）是一种通用的可视化建模语言，可以用来描述、可视化、构造和文档化软件密集型系统的各种工件。

UML 的应用范围

在需求分析阶段，通过用例捕获需求，建立用例模型描述系统功能要求。在设计阶段，通过类和对象等建立静态模型，对协作进行动态建模。在开发阶段，将设计模型转化为

代码。在测试阶段，用类图指导单元测试，用构件图和协作图指导集成测试，用用例图指导系统测试。

静态建模机制

用例图

用例图从用户角度描述系统功能，由用例、参与者以及它们的关系连线组成。参与者是与系统交互的外界实体，用例代表一类功能。用例之间可以存在包含、扩展和泛化关系。

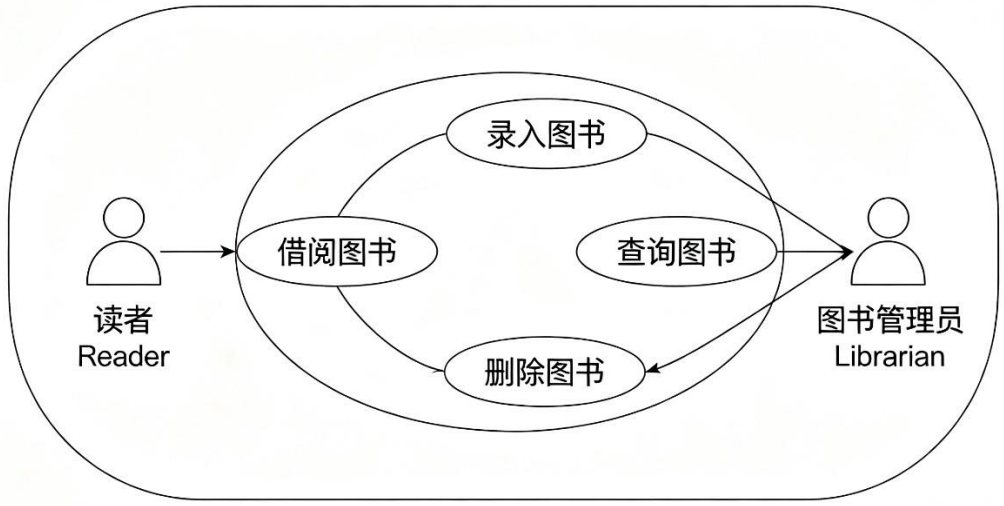


图 7-1 用例图示例——图书管理系统

包含关系：把几个用例的公共部分抽象出来成为新用例，用带<<include>>的虚线箭头表示，基础用例执行后被包含用例一定执行。

扩展关系：在一定条件下把新行为加入到已有用例中，用带<<extend>>的虚线箭头表示，扩展用例只有在满足条件时才执行。

泛化关系：父用例可特化形成多个子用例，子用例继承父用例的所有结构、行为和关系，用带三角形箭头的实线表示。

类图和对象图

类图使用类和对象描述系统的静态结构，展示类与类之间的相互关系。类之间有关联、

依赖、泛化和实现等关系。对象图是类图的实例，展示系统在某一时刻的快照。

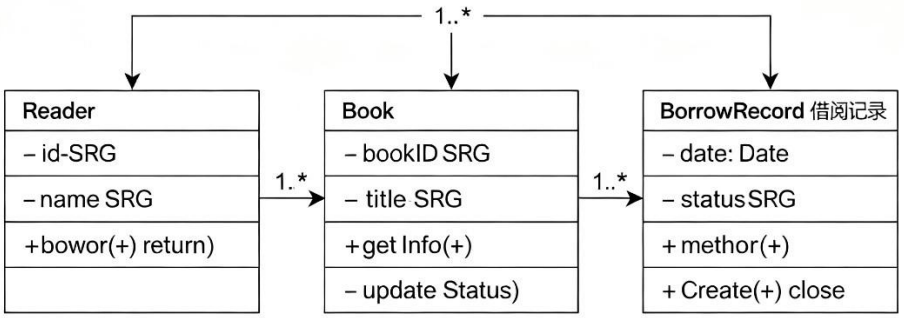


图 7-2 类图示例——图书管理系统

类图用具有 3 个分隔线的矩形表示：顶层是类名，中间是属性，底层是操作。

类与类之间的关系：

1. 关联：表达模型元素间的语义关系，用直线表示，包括二元关联、多元关联、聚合（空心菱形）和组合（实心菱形）等
2. 依赖：一个元素的变化会引起另一个元素的变化，用虚线箭头表示
3. 泛化：描述类的一般-特殊关系，子类继承父类的属性和操作
4. 实现：将一个模型连接到另一个模型（如类与接口），用虚线加三角形表示

动态建模机制

UML 的动态建模机制包括顺序图、协作图、状态图和活动图。

顺序图

顺序图描述一组对象的交互方式，表示完成某项行为的对象和这些对象之间传递消息的时间顺序。由对象、生命线、控制焦点和消息组成。

状态图

状态图描述对象所有可能的状态，以及哪些事件将导致状态的改变。状态图侧重于描述某个对象的动态行为，是对象的生命周期模型。不是所有类都需要画状态图，有明确意义的状态、在不同状态下行为有所不同的类才需要。

描述物理架构的机制

系统架构分为逻辑架构和物理架构。逻辑架构描述系统功能，物理架构描述软件和硬件的分解。

构件图

构件图描述系统的软件组件及其依赖关系。

部署图

部署图显示系统中硬件和软件的物理结构，显示计算机和设备（节点）以及它们之间的互连关系。部署图也显示各组件之间的依赖关系。

本章小结

本章介绍了面向对象的基本概念和特征，以及统一建模语言 UML。UML 的静态建模机制包括用例图、类图和对象图，动态建模机制包括顺序图、协作图、状态图和活动图，物理架构机制包括构件图和部署图。

第八章

面向对象分析概述

面向对象分析 (OOA) 是从问题陈述入手，分析和构造所关心的现实世界问题域的模型，明确系统必须做什么。面向对象分析建立 3 种模型：对象模型（描述系统静态结构）、动态模型（描述系统控制结构）和功能模型（描述系统计算结构）。

建立对象模型

建立对象模型的主要活动包括：划分主题、找出类与对象、识别结构、定义属性、定义服务。这五项活动没有必要的完成顺序。

划分主题

主题是把一组具有较强联系的类组织在一起而得到的类的集合。主题本身不是一个类，主题内部的类之间往往具有某种内在联系。

确定类与对象

通过区分实体类、边界类和控制类来检查对象模型的完整性。更简单的方法是将需求描述的名词作为候选，然后按以下原则筛选：冗余（选择更合理的）、无关（只保留与本系统密切相关的）、笼统（去除笼统的类）、属性（只有一个属性的可作为另一个类的属性）、操作（正确决定是作为类还是操作）、实现（分析阶段不需保留只和时间有关的候选）。

确定类关系

类与类之间的关系包括关联、依赖、泛化和实现等。确定关系后需要分析关联的多重性。

确定属性

属性的确定既与问题域有关，也与目标系统的任务有关。每个对象至少需包含一个属性，属性取值必须适合对象类的所有实例。

确定服务

对象由属性数据和操作 (方法或服务) 封装构成。需要等建立动态模型和功能模型之后，才能最终确定类中应有的服务。类的服务有 4 种来源：对象模型中的服务 (读写属性值)、从事件导出的服务、自状态动作和活动的服务、与数据流图中处理框对应的操作。

建立动态模型

动态模型描述系统随时间变化的行为。建立动态模型的步骤包括：编写脚本、设计用户界面、画事件跟踪图、画状态图。

建立功能模型

功能模型指明了系统应该做什么，由一组数据流图组成。功能模型中的处理对应于对象模型中的服务。

3 种模型之间的关系

功能模型指明了系统应该做什么，动态模型规定了什么时候做，对象模型定义了做事情的实体。3 种模型相互补充、相互配合。

本章小结

本章介绍了面向对象分析的方法，重点介绍了对象模型的建立过程，包括划分主题、确定类与对象、确定类关系、确定属性和确定服务。3 种模型从不同侧面描述系统，相互补充。

第九章

软件体系结构概述

软件体系结构是系统的高层设计，描述了系统的组织结构、组件及其相互关系。良好的体系结构设计有助于提高系统的可维护性、可扩展性和可复用性。

体系结构风格

管道-过滤器体系结构

每个组件具有一组输入和输出，数据在管道中流动，经过过滤器处理。优点是模块独立性好，缺点是不适合交互式系统。

客户机/服务器体系结构

C/S 体系结构通常有两层或三层。瘦客户机模型中数据管理和应用逻辑都在服务器端执行，客户机只负责表示部分。胖客户机模型中服务器只负责数据管理，客户机实现应用逻辑和交互。

模型-视图-控制器（MVC）

MVC 由 Trygve Reenskaug 博士提出，强调将用户的输入、数据模型和数据表示方式分开设计。由 3 部分组成：

模型：代表应用领域中的业务实体和业务逻辑规则，是整个模型的核心，独立于外在的显示内容和形式。

视图：代表 GUI 对象，以用户需要的格式表现模型信息，是系统与外界的交互接口。

控制器：处理用户输入，给模型发送业务事件，并将模型的更新通知视图，保持视图与模型的一致。

MVC 的优点：将用户接口与面向对象的模型分开，允许同样的模型使用多种界面显示方式，用户接口的变化只需修改控制器。

J2EE 体系结构框架

J2EE 的核心体系结构在 MVC 框架基础上扩展，是分层结构，包含 5 层：客户层、表示层、业务层、集成层和资源层。

软件系统的设计模式

设计模式是解决某一类相似问题的方法论。每种设计模式包含 4 个要素：模式名称、问题、解决方案和效果。

常用的 23 种设计模式分为 3 种类型：创建型模式、结构型模式和行为型模式。

本章小结

本章介绍了软件体系结构的常见风格（管道-过滤器、C/S、MVC、J2EE）以及设计模式的基本概念和分类。

第十章

面向对象设计与结构化设计

与结构化设计相比，面向对象设计更符合复杂的、随机性较强和考虑并发性的系统软件设计。结构化设计从系统功能入手，需求变化容易影响整个系统；而面向对象设计基于类、对象、封装、继承等概念，需求变化对系统的局部影响不容易扩展到全局。

面向对象设计与面向对象分析的关系

面向对象的方法不强调需求分析和软件设计的严格区分。面向对象的需求分析和设计活动是一个反复迭代的过程，从分析到设计的过渡是逐渐扩充、细化和完善分析阶段所得到的各种模型的过程。

面向对象设计的过程与原则

面向对象设计的过程

面向对象设计的过程包括 4 个步骤：

1. 建立软件体系结构环境图：定义与软件交互的外部实体及交互特性
2. 软件体系结构设计：可以自底向上、自顶向下或自中向上下进行
3. 对各个子系统进行设计：大多数系统的面向对象设计模型由 4 大部分组成
4. 对象设计及优化：细化原有的分析对象，确定新的对象，对接口和类进行详细说明

面向对象设计的原则

面向对象设计遵循的原则包括：模块化（一个模块通常为一个类或对象）、抽象化（类是对一组相似特征对象的抽象）、信息隐藏（类的内部信息对外界隐藏，通过接口访问）、低耦合、高内聚和复用性。

系统设计

系统分解

系统分解即建立系统的体系结构，把系统分解成若干个子系统。子系统是类、关联、操作、事件和约束的集合。各子系统之间应具有简单明确的接口。

大多数系统的面向对象设计模型由 4 个子系统组成：

问题域子系统：把面向对象分析模型直接拿来，针对实现要求进行增补和调整。

人机交互子系统：包括有效的人机交互所需的显示和输入。

任务管理子系统：包括任务的定义、通信和协调。

数据管理子系统：包括永久数据的存取，隔离物理的数据管理方法。

对象设计

设计类中的服务

从对象模型中引入服务、从动态模型中确定服务（从状态图提取）、从功能模型中确定服务（从数据流图提取）。设计实现服务的方法包括设计算法、选择数据结构、定义内部类和内部操作。

对象设计优化

常见的对象优化设计方法有提高效率的技术和建立良好的继承结构。

本章小结

本章介绍了面向对象设计与结构化设计的区别、面向对象设计的过程和原则、系统设计（系统分解为 4 个子系统）以及对象设计（设计类中的服务和优化）。

思考题

1. 面向对象设计与结构化设计有什么区别?

参考答案：答：结构化设计从系统功能入手，以函数/过程为中心，需求变化容易影响整个系统。面向对象设计基于类、对象、封装、继承等概念，以对象为中心，需求变化对系统的局部影响不容易扩展到全局。面向对象设计更符合复杂的、随机性较强和考虑并发性的系统。

2. 面向对象设计的原则有哪些?

参考答案：答：面向对象设计遵循的原则包括：模块化（一个模块通常为一个类或对象）、抽象化（类是对一组相似特征对象的抽象）、信息隐藏（类的内部信息对外界隐藏，通过接口访问）、低耦合、高内聚和复用性。

3. 系统分解包括哪 4 个子系统?

参考答案：答：大多数系统的面向对象设计模型由 4 个子系统组成：（1）问题域子系统：把 OOA 模型直接拿来，针对实现要求进行增补和调整；（2）人机交互子系统：包括有效的人机交互所需的显示和输入；（3）任务管理子系统：包括任务的定义、通信和协调；（4）数据管理子系统：包括永久数据的存取，隔离物理的数据管理方法。

4. MVC 模式的 3 个组成部分是什么？

参考答案：答：MVC 模式的 3 个组成部分为：（1）模型：代表应用领域中的业务实体和业务逻辑规则，是整个模型的核心，独立于外在的显示内容和形式；（2）视图：代表 GUI 对象，以用户需要的格式表现模型信息，是系统与外界的交互接口；（3）控制器：处理用户输入，给模型发送业务事件，并将模型的更新通知视图，保持视图与模型的一致。

第七章

面向对象的软件工程方法

面向对象的基本概念

面向对象是按照人们认识客观世界的系统思维方式，采用基于对象的概念建立模型，模拟客观世界分析、设计、实现软件的方法。对象指现实中各种各样的实体，对象的内部状态称为属性，运动规律称为方法或服务。类是指具有相似内部状态和运动规律的实体的集合。

面向对象的软件工程方法的特征与优势

面向对象方法的特征：把数据和操作封装在一起形成对象；把特征相似的对象抽象为类；类之间可以存在继承关系形成层次结构；对象之间通过发送消息进行通信；将对象的私有信息封装起来，外界通过有限的接口访问。

面向对象的实施步骤

面向对象分析：从问题陈述入手，分析和构造问题域的模型，明确必须做什么。步骤包括确定问题域、区分类和对象、区分整体对象及组成部分、定义属性、定义服务、确定附加的系统约束。

面向对象设计：对分析结果进行改进和完善。步骤包括设计交互过程和用户接口、设计任务管理、设计全局资源、对象设计。

面向对象实现：使用面向对象语言实现设计。面向对象测试：包括模型测试、类测试、交互测试、系统测试、验收测试等。

统一建模语言 UML

UML 简述

统一建模语言（UML）是一种通用的可视化建模语言，可以用来描述、可视化、构造和文档化软件密集型系统的各种工件。

UML 的应用范围

在需求分析阶段，通过用例捕获需求，建立用例模型描述系统功能要求。在设计阶段，通过类和对象等建立静态模型，对协作进行动态建模。在开发阶段，将设计模型转化为

代码。在测试阶段，用类图指导单元测试，用构件图和协作图指导集成测试，用用例图指导系统测试。

静态建模机制

用例图

用例图从用户角度描述系统功能，由用例、参与者以及它们的关系连线组成。参与者是与系统交互的外界实体，用例代表一类功能。用例之间可以存在包含、扩展和泛化关系。

包含关系：把几个用例的公共部分抽象出来成为新用例，用带<<include>>的虚线箭头表示，基础用例执行后被包含用例一定执行。

扩展关系：在一定条件下把新行为加入到已有用例中，用带<<extend>>的虚线箭头表示，扩展用例只有在满足条件时才执行。

泛化关系：父用例可特化形成多个子用例，子用例继承父用例的所有结构、行为和关系，用带三角形箭头的实线表示。

类图和对象图

类图使用类和对象描述系统的静态结构，展示类与类之间的相互关系。类之间有关联、依赖、泛化和实现等关系。对象图是类图的实例，展示系统在某一时刻的快照。

类图用具有 3 个分隔线的矩形表示：顶层是类名，中间是属性，底层是操作。

类与类之间的关系：

5. 关联：表达模型元素间的语义关系，用直线表示，包括二元关联、多元关联、聚合（空心菱形）和组合（实心菱形）等
6. 依赖：一个元素的变化会引起另一个元素的变化，用虚线箭头表示
7. 泛化：描述类的一般-特殊关系，子类继承父类的属性和操作
8. 实现：将一个模型连接到另一个模型（如类与接口），用虚线加三角形表示

动态建模机制

UML 的动态建模机制包括顺序图、协作图、状态图和活动图。

顺序图

顺序图描述一组对象的交互方式，表示完成某项行为的对象和这些对象之间传递消息的时间顺序。由对象、生命线、控制焦点和消息组成。

状态图

状态图描述对象所有可能的状态，以及哪些事件将导致状态的改变。状态图侧重于描述某个对象的动态行为，是对象的生命周期模型。不是所有类都需要画状态图，有明确意义的状态、在不同状态下行为有所不同的类才需要。

描述物理架构的机制

系统架构分为逻辑架构和物理架构。逻辑架构描述系统功能，物理架构描述软件和硬件的分解。

构件图

构件图描述系统的软件组件及其依赖关系。

部署图

部署图显示系统中硬件和软件的物理结构，显示计算机和设备（节点）以及它们之间的互连关系。部署图也显示各组件之间的依赖关系。

本章小结

本章介绍了面向对象的基本概念和特征，以及统一建模语言 UML。UML 的静态建模机制包括用例图、类图和对象图，动态建模机制包括顺序图、协作图、状态图和活动图，物理架构机制包括构件图和部署图。

第八章

面向对象分析概述

面向对象分析 (OOA) 是从问题陈述入手，分析和构造所关心的现实世界问题域的模型，明确系统必须做什么。面向对象分析建立 3 种模型：对象模型（描述系统静态结构）、动态模型（描述系统控制结构）和功能模型（描述系统计算结构）。

建立对象模型

建立对象模型的主要活动包括：划分主题、找出类与对象、识别结构、定义属性、定义服务。这五项活动没有必要的完成顺序。

划分主题

主题是把一组具有较强联系的类组织在一起而得到的类的集合。主题本身不是一个类，主题内部的类之间往往具有某种内在联系。

确定类与对象

通过区分实体类、边界类和控制类来检查对象模型的完整性。更简单的方法是将需求描述的名词作为候选，然后按以下原则筛选：冗余（选择更合理的）、无关（只保留与本系统密切相关的）、笼统（去除笼统的类）、属性（只有一个属性的可作为另一个类的属性）、操作（正确决定是作为类还是操作）、实现（分析阶段不需保留只和时间有关的候选）。

确定类关系

类与类之间的关系包括关联、依赖、泛化和实现等。确定关系后需要分析关联的多重性。

确定属性

属性的确定既与问题域有关，也与目标系统的任务有关。每个对象至少需包含一个属性，属性取值必须适合对象类的所有实例。

确定服务

对象由属性数据和操作 (方法或服务) 封装构成。需要等建立动态模型和功能模型之后，才能最终确定类中应有的服务。类的服务有 4 种来源：对象模型中的服务 (读写属性值)、从事件导出的服务、自状态动作和活动的服务、与数据流图中处理框对应的操作。

建立动态模型

动态模型描述系统随时间变化的行为。建立动态模型的步骤包括：编写脚本、设计用户界面、画事件跟踪图、画状态图。

建立功能模型

功能模型指明了系统应该做什么，由一组数据流图组成。功能模型中的处理对应于对象模型中的服务。

3 种模型之间的关系

功能模型指明了系统应该做什么，动态模型规定了什么时候做，对象模型定义了做事情的实体。3 种模型相互补充、相互配合。

本章小结

本章介绍了面向对象分析的方法，重点介绍了对象模型的建立过程，包括划分主题、确定类与对象、确定类关系、确定属性和确定服务。3 种模型从不同侧面描述系统，相互补充。

第九章

软件体系结构概述

软件体系结构是系统的高层设计，描述了系统的组织结构、组件及其相互关系。良好的体系结构设计有助于提高系统的可维护性、可扩展性和可复用性。

体系结构风格

管道-过滤器体系结构

每个组件具有一组输入和输出，数据在管道中流动，经过过滤器处理。优点是模块独立性好，缺点是不适合交互式系统。

客户机/服务器体系结构

C/S 体系结构通常有两层或三层。瘦客户机模型中数据管理和应用逻辑都在服务器端执行，客户机只负责表示部分。胖客户机模型中服务器只负责数据管理，客户机实现应用逻辑和交互。

模型-视图-控制器（MVC）

MVC 由 Trygve Reenskaug 博士提出，强调将用户的输入、数据模型和数据表示方式分开设计。由 3 部分组成：

模型：代表应用领域中的业务实体和业务逻辑规则，是整个模型的核心，独立于外在的显示内容和形式。

视图：代表 GUI 对象，以用户需要的格式表现模型信息，是系统与外界的交互接口。

控制器：处理用户输入，给模型发送业务事件，并将模型的更新通知视图，保持视图与模型的一致。

MVC 的优点：将用户接口与面向对象的模型分开，允许同样的模型使用多种界面显示方式，用户接口的变化只需修改控制器。

J2EE 体系结构框架

J2EE 的核心体系结构在 MVC 框架基础上扩展，是分层结构，包含 5 层：客户层、表示层、业务层、集成层和资源层。

软件系统的设计模式

设计模式是解决某一类相似问题的方法论。每种设计模式包含 4 个要素：模式名称、问题、解决方案和效果。

常用的 23 种设计模式分为 3 种类型：创建型模式、结构型模式和行为型模式。

本章小结

本章介绍了软件体系结构的常见风格（管道-过滤器、C/S、MVC、J2EE）以及设计模式的基本概念和分类。

第十章

面向对象设计与结构化设计

与结构化设计相比，面向对象设计更符合复杂的、随机性较强和考虑并发性的系统软件设计。结构化设计从系统功能入手，需求变化容易影响整个系统；而面向对象设计基于类、对象、封装、继承等概念，需求变化对系统的局部影响不容易扩展到全局。

面向对象设计与面向对象分析的关系

面向对象的方法不强调需求分析和软件设计的严格区分。面向对象的需求分析和设计活动是一个反复迭代的过程，从分析到设计的过渡是逐渐扩充、细化和完善分析阶段所得到的各种模型的过程。

面向对象设计的过程与原则

面向对象设计的过程

面向对象设计的过程包括 4 个步骤：

1. 建立软件体系结构环境图：定义与软件交互的外部实体及交互特性
2. 软件体系结构设计：可以自底向上、自顶向下或自中向上下进行
3. 对各个子系统进行设计：大多数系统的面向对象设计模型由 4 大部分组成
4. 对象设计及优化：细化原有的分析对象，确定新的对象，对接口和类进行详细说明

面向对象设计的原则

面向对象设计遵循的原则包括：模块化（一个模块通常为一个类或对象）、抽象化（类是对一组相似特征对象的抽象）、信息隐藏（类的内部信息对外界隐藏，通过接口访问）、低耦合、高内聚和复用性。

系统设计

系统分解

系统分解即建立系统的体系结构，把系统分解成若干个子系统。子系统是类、关联、操作、事件和约束的集合。各子系统之间应具有简单明确的接口。

大多数系统的面向对象设计模型由 4 个子系统组成：

问题域子系统：把面向对象分析模型直接拿来，针对实现要求进行增补和调整。

人机交互子系统：包括有效的人机交互所需的显示和输入。

任务管理子系统：包括任务的定义、通信和协调。

数据管理子系统：包括永久数据的存取，隔离物理的数据管理方法。

对象设计

设计类中的服务

从对象模型中引入服务、从动态模型中确定服务（从状态图提取）、从功能模型中确定服务（从数据流图提取）。设计实现服务的方法包括设计算法、选择数据结构、定义内部类和内部操作。

对象设计优化

常见的对象优化设计方法有提高效率的技术和建立良好的继承结构。

本章小结

本章介绍了面向对象设计与结构化设计的区别、面向对象设计的过程和原则、系统设计（系统分解为 4 个子系统）以及对象设计（设计类中的服务和优化）。

思考题

1. 面向对象设计与结构化设计有什么区别?

2. 面向对象设计的原则有哪些?

3. 系统分解包括哪 4 个子系统?

4. MVC 模式的 3 个组成部分是什么?