第3章 需求分析

为了开发出真正满足用户需求的软件产品，首要任务是深入探究并精准把握用户的实际需求。对软件需求的细致理解，不仅是软件开发项目得以顺利推进的基础，更是软件开发工作获得成功的关键所在。

尽管在可行性研究阶段已经初步了解了用户的需求，甚至还提出了一些可行的解决方案。但是，可行性研究的主要目的在于以较低的成本和较短的时间来评估项目是否具备实施的可能性，在可行性研究阶段往往会忽略很多细节和特定场景下的用户行为。所以，可行性研究并不能代替需求分析，因为它并不能全面、详尽地回答“系统必须做什么”这个问题。

需求分析是软件定义时期的最后一个阶段，它的基本任务是准确地回答“系统必须做什么”这个问题。

3.1 需求分析概述

软件需求分析作为软件工程的核心环节，承担着将业务诉求转化为技术规范的关键任务。其理论体系由需求分析原则、任务与方法构成三维框架，共同指导分析人员系统化地完成需求工程活动。需求分析原则确立了信息域建模、功能定义、行为描述与模型分解四大支柱，通过结构化思维将模糊需求转化为精确模型。信息域建模构建业务数据的流动全景，功能定义实现业务流程的算法映射，行为描述刻画系统动态响应机制，模型分解则通过分层解耦策略降低复杂度，四者形成有机整体，确保需求描述的完整性与可行性。

在实施层面，需求分析任务系统化展开为需求识别、逻辑建模与文档化三大阶段。需求识别需构建功能、性能、环境、安全与界面需求的多维矩阵，通过影响关系图揭示需求间的协同关系与制约因素；逻辑建模运用数据流图、实体-联系图与状态转换图建立可视化表达体系，实现业务逻辑向技术方案的平滑过渡；文档化过程通过规格说明书、原型设计与变更记录形成知识资产，构建可验证、可追溯的需求基线。全过程强调业务规则的形式化表达与工程化验证，通过多级评审机制确保需求质量。

方法论层面，结构化分析与面向对象分析形成互补范式。结构化方法以数据流为核心，通过分层分解构建严谨的功能模型，适用于流程固化的数据处理系统；面向对象方法以对象抽象为基础，通过类与关系的建模应对复杂交互场景，擅长处理动态演化的业务需求。现代实践中，二者常融合应用：利用结构化方法规范核心数据处理流程，借助面向对象技术构建灵活交互模型，形成兼顾严谨性与扩展性的混合分析框架。这种多维方法体系有效解决了传统单一方法存在的局限性，既能确保关键业务逻辑的精确实现，又能适应需求迭代与技

术创新，为数字化转型时代的软件工程提供了强有力的方法论支撑。

3.1.1 需求分析原则

需求分析原则是软件工程中指导需求获取与定义的核心准则，其本质在于通过系统化方法将模糊的业务诉求转化为可执行的开发方案。这一过程要求分析人员具备跨领域理解能力，既要深入业务运作机理，又要掌握软件构建规律，最终形成完整的需求描述体系。需求分析并非简单的需求记录，而是需要运用工程化思维对用户诉求进行结构化梳理、逻辑化重组和标准化表达，确保技术实现与业务目标的高度统一。在这个过程中，信息域建模、功能定义、行为描述和模型分解四大原则构成了相互支撑的分析框架，共同确保需求描述的完整性、准确性和可实施性。

在需求分析的初始阶段，信息域建模占据基础性地位。信息域作为业务系统的数字镜像，完整映射了组织运作中的数据流动规律和加工处理机制。分析人员需要采用系统工程的视角，将业务场景中的信息要素抽象为可分析的模型结构。这个过程包含三个递进层次：首先是信息流的捕捉，需要绘制出数据在业务环节间的传递路径，识别关键的数据输入输出节点，明确信息交换的频率和模式；其次是信息内容的解析，需要定义每个数据单元的结构属性，包括数据类型、取值范围、约束条件等元数据特征；最后是信息结构的构建，需要揭示数据实体间的关联关系，建立符合业务逻辑的数据组织框架。通过这三个层次的建模，能够将离散的业务数据转化为具有明确语义的信息网络，为后续功能设计奠定数据基础。这种建模过程需要特别关注信息的生命周期管理，从数据产生、流转、加工到归档的完整轨迹，都需要在模型中体现，确保数据处理的完整性和可追溯性。

在明确信息域的基础上，功能定义成为需求分析的核心任务。软件功能本质上是信息加工处理过程的算法化表达，需要将业务需求转化为具体的服务能力。这一过程需要建立多维度功能模型：在横向维度上，需要划分功能模块的边界，明确每个模块的职责范围；在纵向维度上，需要构建功能层次结构，从宏观业务功能逐级分解到可执行的原子功能单元。功能定义的关键在于保持功能粒度的适中性，既要有足够的抽象度体现业务价值，又要具备足够的精细度指导技术实现。在此过程中，需要特别注意功能间的协同关系，包括功能调用顺序、数据依赖关系、并发控制机制等。同时，功能模型还需要体现非功能性需求对功能实现的影响，例如性能指标对算法选择的要求，安全需求对功能访问控制的设计等。通过建立完整的功能体系，确保软件服务与业务需求形成精准映射。

行为描述是需求分析中动态维度的补充，着重刻画系统对外部刺激的响应机制。行为描述需要覆盖正常操作流程和异常处理场景两个层面：在正常流程层面，需要定义用户操作、设备输入、定时触发等各类事件的处理逻辑，建立从事件捕获到系统响应的完整链路；在异常处理层面，需要预设各类容错机制，包括输入校验、故障检测、恢复策略等保障措施。行为描述需要特别关注系统状态的演变过程，通过状态迁移图描述系统在不同条件下的行为模式。在这个过程中，需要明确状态转换的触发条件、前置校验、动作执行和后置处理等细节要素。同时，行为描述还需要考虑并发事件的处理策略，定义事件优先级、冲突解决机制等复杂场景下的系统响应规则。通过建立精确的行为模型，可以有效预防需求盲区，确保系统在各种运行环境下都能保持预期行为。

模型分解作为需求分析的整合手段，承担着降低系统复杂度的关键作用。面对现代软

件系统的复杂性特征，分层解耦的分解策略成为必然选择。在分解过程中需要遵循三个基本原则：首先是模块化原则，将系统划分为高内聚、低耦合的功能模块；其次是抽象层次原则，建立从业务概念到技术实现的分层模型体系；最后是关注点分离原则，将不同性质的需求要素分配到适当的模型维度。模型分解需要保持各子模型间的协调一致性，通过建立模型间的映射关系和约束规则，确保分解后的模型片段能够无缝整合。在这个过程中，需要特别注意接口规范的制定，明确模块间的交互协议和数据格式，为后续的系统集成奠定基础。通过科学的模型分解，不仅能够提高需求分析的可管理性，还能为后续的架构设计提供清晰的指导框架。

这些原则在实践应用中呈现出显著的协同效应：信息域建模为功能定义提供数据支撑，功能模型为行为描述确立操作主体，行为模型则通过动态视角完善系统需求，而模型分解贯穿始终确保分析的可行性。这种多维度的需求分析方法，能够有效克服传统需求分析中常见的片面性和局限性。在具体实施过程中，需要建立迭代优化的机制，通过多轮次的模型验证和需求确认，逐步逼近真实需求本质。同时，要注重需求可验证性的构建，为每个需求项建立明确的验收标准，确保开发成果的可度量性。

现代需求分析方法的发展趋势表明，优秀的需求工程实践正在向模型驱动的方向演进。建立形式化的模型体系，不仅能够提高需求描述的精确度，还能实现需求模型到设计模型的自动化转换。这种模型化的需求表达方式，为需求追踪、变更管理和质量保证提供了有力支撑。特别是在复杂系统开发中，模型化的需求分析能够有效管理需求要素间的复杂关联，预防需求蔓延和范围失控的风险。

在需求分析的质量控制方面，需要建立多层次的验证机制。首先是逻辑一致性验证，确保各模型间不存在矛盾冲突；其次是完整性验证，检查是否覆盖所有已知需求场景；最后是可行性验证，评估技术实现的可能性。这些验证活动需要贯穿需求分析的全过程，通过建立检查清单、评审流程和验证用例等手段，持续提升需求规格的质量水平。同时，要注重需求文档的可维护性设计，建立模块化的文档结构，便于后续的需求变更和版本管理。

需求分析作为软件工程的基石，其质量直接决定项目的成败。遵循上述原则建立的需求模型体系，不仅为开发团队提供了明确的技术蓝图，也为项目干系人搭建了沟通的桥梁。通过标准化的模型语言，能够有效消除业务人员与技术人员之间的认知鸿沟，确保需求理解的统一性。这种模型化的需求表达方式，使得软件需求从抽象概念转化为可执行的开发指令，为高质量软件产品的构建奠定了坚实基础。随着软件开发方法的不断演进，需求分析原则也在持续发展完善，但其核心目标始终不变：在复杂的需求丛林中开辟出清晰的技术路径，将用户愿景转化为可靠的软件解决方案。

3.1.2 需求分析任务

需求分析任务是软件工程中承前启后的核心环节，其核心目标在于将模糊的业务诉求转化为结构化、可验证的系统需求集合。这一过程不仅需要精确捕捉用户显性需求，还需通过系统性分析挖掘潜在需求，最终形成完整的需求基线。作为软件生命周期的基石，需求分析的质量直接影响后续开发活动的效率和产物的质量，其任务实施需遵循严格的工程化方法，确保需求描述的全面性、一致性和可追溯性。

1. 明确目标系统的具体需求

需求分析的首要任务是系统化识别并定义目标系统的具体需求。这一过程需要构建多维度的需求识别框架，覆盖功能、性能、环境、安全及用户界面等核心维度，并通过关联分析确保各需求类型的协调统一。

1）功能需求

功能需求是系统能力的核心载体，定义了系统必须提供的服务集合。在需求分析中，需通过业务场景分解与重构，将用户操作流程转化为功能模块的交互逻辑。功能需求的提炼需遵循原子性原则，将复杂业务流程拆解为可独立验证的功能单元，同时保持功能间的逻辑连贯性。在此过程中，需求优先级评估机制不可或缺，需结合业务价值、实现复杂度等因素建立功能实施的优先级矩阵。功能需求描述需包含触发条件、输入输出定义、处理逻辑及异常处理机制，确保开发团队能够准确理解每个功能的执行边界和预期效果。

2）性能需求

性能需求是系统质量的重要保障，需建立可量化的指标体系。分析人员需从响应时效、吞吐量、资源利用率、容量边界等维度定义性能基准，并区分正常负载与峰值负载场景下的性能要求。对于实时系统，需明确端到端处理延迟的约束条件；对于高并发场景，需规定最大用户承载量及性能降级策略。性能需求需与功能需求形成映射关系，以确保每个功能的执行效率符合业务预期。此外，需考虑性能监控与调优的可扩展性，为系统优化预留空间。

3）环境需求

环境需求定义了系统运行的物理与逻辑支撑框架。硬件需求需明确服务器配置、存储容量、网络带宽等基础设施参数；软件需求需规定操作系统版本、数据库类型、中间件版本等依赖环境；网络需求需描述通信协议、数据传输安全机制及分布式架构要求。分析人员需制定兼容性矩阵，确保系统在不同环境下的适配性，同时为未来技术演进预留弹性扩展能力。对于云原生系统，还需明确容器化部署、微服务架构及动态扩缩容策略。

4）安全需求

安全需求构建了系统的防御体系，需覆盖数据保护、访问控制与风险应对三个层面。数据安全需定义加密算法、传输安全协议及存储隔离机制；身份认证需支持多因素验证与单点登录；权限管理需基于角色或属性建立细粒度访问控制策略。同时需规划安全审计功能，记录关键操作日志并支持溯源分析。安全需求需与业务流程深度融合，例如在支付流程中嵌入风险检测机制，在数据导出时触发审批流程，形成纵深防御体系。

5）用户界面需求

用户界面需求平衡了交互效率与体验品质，需建立统一的界面设计规范。交互设计需定义操作流线、导航逻辑及反馈机制，确保用户操作的直观性与一致性；视觉设计需规定色彩体系、字体规范及布局原则，兼顾美观性与可访问性。对于多终端系统，需制订响应式设计策略，确保不同设备上的界面适配效果。此外，需考虑无障碍设计需求，例如为视觉障碍用户提供语音导航支持，为操作效率要求高的场景设计快捷键体系。

在需求分析过程中，需通过需求影响关系图揭示各需求类型的协同与制约关系。例如，安全需求的加密算法选择可能影响系统性能，界面设计的动态效果可能增加资源消耗。这种关联性分析有助于优化需求实现方案，确保系统整体效能的最优化。

2. 建立目标系统的逻辑模型

逻辑模型是连接需求与设计的桥梁，通过可视化建模方法将抽象需求转化为结构化表达。分析人员需根据系统特征选择适当的建模工具，构建多视角模型体系。

数据流模型：通过分层递进的方式展现信息的产生、加工与消费过程。顶层数据流图描述系统与外部实体的交互边界，底层细节图揭示内部处理逻辑。数据字典需同步定义数据元素的属性、格式及约束条件，确保数据语义的精确性。

实体-联系模型：聚焦业务领域的核心概念及其关联规则，通过实体及其属性定义和联系约束建立业务数据的结构化表达。模型需遵循范式化原则，消除数据冗余并保证完整性。

状态转换模型：刻画系统行为的动态特征，明确定义状态变迁的触发条件、前置校验及响应动作。对于复杂状态机，需采用层次化状态转换图分解状态空间，降低模型复杂度。

模型构建需遵循逐步求精原则，从全局概览到细节描述逐层深化。模型验证机制需通过形式化检查、场景推演等方法确认模型的完备性与一致性。例如，通过状态转换图与数据流图的交叉验证，确保业务流程与数据处理逻辑的匹配。模型间的关联映射需建立明确的对应规则，例如将数据流图中的处理节点映射为功能需求条目，将实体-联系模型中的属性映射为界面输入字段。

3. 编写文档

需求文档化是需求分析成果的结晶过程，其质量直接影响后续开发活动的有效性。文档体系需兼顾全面性与可维护性，从而形成层次分明的知识资产体系。

1）需求规格说明书

作为核心交付物，需求规格说明书需采用模块化结构设计。引言部分界定系统边界与术语体系，避免概念歧义；功能需求部分采用结构化表达，每个功能条目包含输入预处理、核心逻辑、输出生成及异常处理描述；性能需求部分分为基准场景和极限场景，定义度量方法与验收阈值；安全需求部分形成策略、机制、实施的三层描述结构；用户界面需求部分附带交互原型图例及设计规范说明；验收标准部分需明确功能覆盖度、性能达标率及缺陷容忍度等量化指标。文档需嵌入需求追踪矩阵，关联原始业务诉求与技术实现方案。

2）原型设计文档

原型设计文档需平衡保真度与迭代效率。低保真原型聚焦信息架构与流程验证，通过线框图展现界面布局与导航逻辑；高保真原型细化视觉设计与交互细节，采用动态演示模拟用户操作流。文档需记录设计决策依据，例如色彩选择的情感传达意图，布局优化的认知负荷分析。同时需建立用户反馈闭环机制，将测试结果转化为界面优化需求。

3）需求变更记录

变更管理需构建全生命周期管控流程。变更申请需说明业务背景、影响范围及替代方案；影响分析需评估对进度、成本及架构的连锁反应；审批决策需基于变更优先级与资源约束；版本更新需同步修改需求文档与关联模型。变更记录需采用版本树可视化呈现需求演化路径，并为每个变更项标记追溯标识，确保历史可审计。

文档质量保障需通过多级评审机制实现。技术评审聚焦需求可实现性，业务评审确认需求符合性，合规评审检查安全与隐私要求。文档版本控制需支持基线管理，通过差异对比工具追踪内容变更，确保团队成员始终基于最新版本开展工作。

需求分析任务的复杂性源于业务与技术视角的融合挑战。分析人员需扮演“翻译者”角

色，将业务语言转化为技术规范，同时维护需求的可验证性与可追踪性。系统化的需求工程实践，能够有效降低开发过程中的返工风险，为高质量软件交付奠定坚实基础。

3.1.3 需求分析方法

需求分析方法作为软件工程的核心技术之一，旨在通过系统化的方法论将模糊的用户需求转化为结构化、可执行的技术规范。常用的方法包括结构化分析方法与面向对象分析方法，二者分别基于不同的哲学视角与技术框架，适用于不同类型的系统开发场景。

1. 结构化分析方法

结构化分析方法（Structured Analysis，SA）是一种基于数据流驱动的传统分析范式，其核心逻辑是通过分解与抽象机制，将复杂系统转化为可管理的功能模块集合。该方法强调系统的功能行为与数据流动的显式建模，尤其适用于流程明确、功能边界清晰的系统。

1）基本思想

结构化分析的核心在于层次化分解与数据流控制。通过自顶向下的逐层细化策略，系统被分解为多个功能层次，每一层次仅关注当前抽象级别的功能实现与数据交互。数据字典作为全流程的核心枢纽，统一管理系统中所有数据元素的定义、格式及约束条件，确保数据模型的一致性。分析过程中同步构建数据模型（描述数据实体及其联系）、功能模型（描述数据处理逻辑）与行为模型（描述系统状态变迁），形成三位一体的需求描述体系。

2）主要步骤

（1）理解当前系统

通过用户访谈、流程观察与文档分析，全面梳理现有业务系统的运行机制。重点关注输入输出数据的类型、处理规则及关键业务节点，识别冗余流程与优化空间。此阶段需建立业务流程图与用例场景库，为后续建模提供基线参考。

（2）定义数据流图

数据流图通过分层建模技术，以图形化方式展现数据在系统内的流动路径与处理逻辑。顶层上下文图界定系统与外部实体的交互边界；下层图逐级展开功能细节，直至原子级处理过程。每个处理节点需明确定义输入输出关系，数据存储节点需标注持久化机制，数据流箭头需关联具体数据项。

（3）建立数据字典

数据字典采用标准化模板定义所有数据元素的元数据信息，包括数据名称、结构类型（基本类型或复合类型）、取值范围、数据来源及使用约束。对于复杂数据结构（如嵌套表或树状结构），需定义其层级关系与访问路径。数据字典与数据流图形成双向映射关系，确保数据语义的全局一致性。

（4）构建功能模型

功能模型通过结构化语言（如判定表、判定树或过程描述语言）详细描述每个处理节点的逻辑规则。对于条件分支逻辑，需覆盖所有可能的输入组合与异常场景；对于计算逻辑，需明确算法公式或处理步骤；对于数据转换逻辑，需定义源数据与目标数据的映射规则。

（5）建立行为模型

行为模型采用状态转换图描述系统的动态响应机制。每个状态需定义进入条件、持续

条件及退出条件；状态转换需标注触发事件与响应动作；并发状态需通过并行状态机建模。此模型需与数据流图的功能节点关联，确保功能逻辑与状态转换的同步性。

3）特点

系统性：通过层次化分解策略，将复杂问题转化为可管理的子问题集合，确保分析过程的逻辑连贯性。

数据驱动：以数据流为核心纽带，显式揭示功能模块间的依赖关系与信息交换机制。

模块独立性：遵循高内聚、低耦合的模块划分原则，降低系统复杂度并提高可维护性。

形式化验证：数据字典与模型的标准化定义支持自动化一致性检查，减少人工审查误差。

2. 面向对象分析方法

面向对象分析方法（Object-Oriented Analysis，OOA）以现实世界的事物抽象为核心，通过对象、类与关系的建模技术，构建贴近业务本质的需求模型。该方法擅长处理需求动态变化、交互复杂的系统，尤其适用于用户界面密集或业务规则灵活的场景。

1）基本思想

面向对象分析的核心在于对象抽象与关系建模。系统被视为相互协作的对象集合，每个对象封装特定属性和行为，并通过消息传递实现交互。类作为对象的抽象模板，通过继承机制实现层次化分类，通过聚合与关联表达对象间的结构关系。分析过程聚焦于静态结构（如类图）、动态行为（如交互图）与功能约束（如用例图）的三维建模。

2）主要步骤

（1）识别对象与类

通过领域术语提取、用例分析及场景模拟，识别系统中的关键实体对象。每个对象需定义其核心属性（如数据类型与可见性）与方法（如操作逻辑与调用权限）。具有共性特征的对象被抽象为类，类层次结构通过泛化-特化关系组织。

（2）定义对象关系

继承关系：建立父类与子类间的层次化分类体系，实现属性与方法的复用与扩展。

聚合关系：描述整体与部分的包容关系（如“汽车包含发动机”），支持部分对象的独立生命周期管理。

关联关系：定义对象间的业务连接（如“学生选课”），通过多重性约束（如一对多）量化关联强度。

依赖关系：标识临时性交互依赖（如“订单生成报表”），确保模型反映动态协作需求。

（3）构建动态模型

顺序图：按时间序列展示对象间的消息传递过程，标注同步/异步调用、返回结果及异常处理路径。

状态转换图：描述单个对象在其生命周期内的状态转换逻辑，包括事件触发条件、监护条件及动作执行序列。

活动图：模拟业务流程中的并行任务与决策分支，通过泳道划分明确不同对象的职责边界。

（4）定义消息传递机制

消息作为对象间交互的载体，需明确其类型（同步调用、异步信号或数据流）、参数结构

及异常处理策略。对于复杂交互场景，需设计消息队列机制与超时重试策略，确保系统鲁棒性。

(5) 完善约束规则

通过对象约束语言或自然语言描述业务规则，包括数据验证逻辑（如“订单金额必须大于零”）、状态约束（如“已支付订单不可取消”）及权限控制（如“仅管理员可修改配置参数”）。

3) 特点

现实映射性：对象模型直接反映业务领域的实体与关系，降低需求理解的认知偏差。

封装隔离：通过信息隐藏机制，隔离对象的内部实现细节与外部接口，提升系统的模块化程度。

扩展灵活性：继承与多态机制支持功能的增量式扩展，适应需求迭代演化。

行为可视化：动态模型直观展示系统运行时的交互逻辑，便于早期发现设计缺陷。

结构化分析更适用于数据处理密集、流程固化的系统（如工业控制系统、传统金融交易系统），其形式化模型便于实现自动化代码生成与严格的功能验证。面向对象分析则更适合交互复杂、需求多变的系统（如电子商务平台、社交应用），其抽象机制能够有效应对业务规则的动态调整与用户需求的快速迭代。

在实际项目中，二者并非互斥。现代需求工程常采用混合策略：利用结构化方法梳理核心数据流，结合面向对象技术构建交互模型，形成多维互补的分析框架。这种融合方法既能保证数据处理逻辑的严谨性，又能满足用户体验与业务灵活性的双重需求，可为复杂系统的需求建模提供全面支持。

需求分析中的严谨治学与工匠精神

在软件需求分析的教学中，培养学生严谨的专业态度与追求卓越的工匠精神是提升技术能力的重要维度。需求分析不仅是技术方案的设计起点，更是锻造工程思维、培育职业素养的关键环节。教师通过系统化的建模方法与规范化的工程实践，引导学生理解技术工作的科学性与艺术性，在精确性与创新性之间找到平衡点。

1. 精准建模中的科学态度

信息域建模要求对数据流动规律进行毫厘不差的捕捉，这映射着技术工作者的科学精神。教学中通过数据字典的构建训练，学生体会结构化定义的价值：每个数据元素的类型、约束必须明确定义，如同实验室中的精密仪器校准。例如，在金融系统的交易流程建模中，小数点后四位的精度误差可能导致资金结算的重大偏差。这种训练让学生理解，技术方案的可信度建立在每个细节的精准刻画之上，唯有严谨的工程思维才能构筑可靠的技术基座。

2. 功能定义中的系统思维

功能分解过程体现着化繁为简的智慧，需要将宏观需求转化为可验证的原子单元。教师通过功能优先级矩阵的构建，引导学生建立层次化思考模式：既要保持业务逻辑的完整性，又要确保技术实现的可行性。在智能家居系统的功能设计中，需将“环境自适应”这种抽象需求拆解为温度监测、设备联动、异常预警等具体模块，每个模块的输入输出需经多轮验证。这种系统化的分解能力，培养的是对复杂问题的驾驭能力，以及对技术方案的敬畏之心。

3. 文档规范中的匠心追求

需求规格说明书的质量直接反映技术工作的专业水准。通过模块化文档结构的训练，学生领悟规范表达的重要性：功能描述需同时具备可读性与可验证性，性能指标需量化到可测量的维度。例如，在医疗系统的需求文档中，“快速响应”必须转化为“90%请求在200ms内处理完成”的具体阈值。文档的版本控制与变更记录管理，则培养学生对技术成果的珍视态度，使学生明白每一次修改都应如同匠人雕琢作品般审慎细致。

4. 持续验证中的优化意识

多级评审机制与模型验证流程，诠释着技术工作永无止境的优化追求。教师通过状态转换图的场景推演，引导学生建立“零容忍缺陷”的思维习惯：需穷举所有可能的系统状态变迁路径，预设容错机制。在物流系统的异常处理模型中，需模拟网络中断、数据冲突等极端场景，设计自动重试与人工介入的双重保障。这种反复推敲的过程，让学生理解优秀的技术方案往往诞生于千百次的迭代优化。

我们通过将严谨治学的科学态度与精益求精的工匠精神融入技术实践，塑造既具备扎实专业功底、又拥有持久创新动力的软件人才，使其在技术演进中持续创造价值，推动行业向更高标准迈进。

3.2 结构化方法

结构化方法是一种传统的、基于数据流的需求分析技术，它主要用于大型、复杂系统的需求获取和定义。该方法强调自顶向下、逐步求精的分析过程，旨在将系统需求分解为更小、更具体、更易于管理的部分。在初始阶段，系统流程图被用于描述系统的物理交互框架，通过图形化符号明确系统与外部实体（用户、硬件或其他系统）之间的物理边界和数据传递方式，为后续逻辑分析奠定基础。这一工具帮助分析人员从宏观视角梳理系统运行的物理环境和交互媒介，例如纸质表单、数据库文件或硬件接口。

结构化分析方法的基本思想是将软件系统看作一个由数据流、数据处理、数据存储和数据源等元素组成的有机整体。通过分析数据流在系统中的流动和处理过程，确定系统的功能需求、数据需求和性能需求。在此阶段，数据流图成为核心建模工具，它以分层结构展现系统的逻辑处理流程：顶层上下文图定义系统与外部实体的交互关系；逐层分解的子图通过加工节点、数据存储和数据流箭头细化内部功能模块。同时，数据字典作为元数据管理工具，对数据流图中的所有数据元素进行标准化定义，包括数据流名称、数据结构、数据项类型及取值范围，确保数据语义的精确性和一致性。数据存储的复杂关系则通过实体-联系图进行建模，以实体、属性和联系三元组描述数据间的逻辑约束，为数据库设计提供概念模型支持。

该方法强调从系统的整体结构出发，通过逐层分解和细化，逐步揭示系统的内部结构和功能细节。在分解过程中，遵循“高内聚、低耦合”的原则，确保每个部分都具有相对独立的功能和完整的数据结构。系统流程图、数据流图与实体-联系图形成多维度建模体系：流程图刻画物理交互框架，数据流图聚焦功能逻辑分解，实体-联系图则锚定数据存储结构。数据字典作为贯穿始终的语义枢纽，将所有图形元素转化为可追溯的标准化的定义。最终，这些工具协同生成的需求规格文档覆盖功能需求（通过数据流图层级描述）、数据需求（通过实

体-联系图和数据字典定义)及性能需求(如数据流速率、存储容量约束),形成完整的结构化分析成果,为后续系统设计提供严格的输入依据。

3.2.1 系统流程图

在进行可行性研究时需要了解和分析现有的系统,并以概括的形式表达对现有系统的认识;进入设计阶段以后应该把设想的新系统的逻辑模型转变成物理模型,因此需要描绘未来的物理系统的概貌。

系统流程图是概括地描绘物理系统的传统工具。它的基本思想是用图形符号以黑盒子形式描绘组成系统的每个部件。系统流程图表达的是数据在系统各部件之间流动的情况,而不是对数据进行加工处理的控制过程。

系统流程图的图形符号如表 3-1 所示。

表 3-1 系统流程图的图形符号

符 号	名 称	说 明
	处理	能改变数据值或数据位置的加工或部件
	输入输出	表示输入或输出,是一个广义的不指明具体设备的符号
	连接	指出转到图的另一部分或从图的另一部分转来,通常在同一页
	换页连接	指出转到另一页的图或由另一页的图转来
	数据流	连接其他符号,指明数据流动方向
	文档	通常表示打印输出,也可表示用打印终端输入数据
	联机存储	任何种类的联机存储,包括磁盘、磁鼓和海量存储器件等
	磁盘	磁盘输入输出,也可表示存储在磁盘上的文件或数据库
	显示	CRT 终端或类似的显示部件,可用于输入或输出,也可既输入又输出
	人工输入	人工输入数据的脱机处理
	人工操作	人工完成的处理
	辅助操作	使用设备进行的脱机操作
	通信链路	通过远程通信线路或链路传送数据

【案例 3-1】 某库房的零件库存管理系统。各种零件数量以及每种零件的库存量临界值等数据记录在库存文件中。当库房中零件数量有变化时,需要修改库存文件。若某种零件的库存量少于其库存量临界值时,则报告采购部门订货,系统规定每天向采购部门发送一

次订货报告。库房使用一台计算机完成更新库存文件和生成订货报告的任务。零件库存量的每一次变化称为一个事务，由库房终端输入计算机。库存清单程序对事务进行处理，更新存储的库存文件，并且把订货信息进行联机存储。最后，每天由报告生成程序读取订货信息并且生成订货报告。该零件库存管理系统的系统流程图如图 3-1 所示。

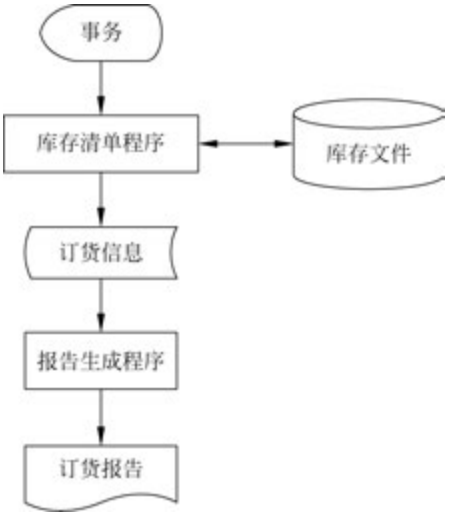


图 3-1　零件库存管理系统的系统流程图

【案例 3-2】 银行存取款系统。如果储户要存款，储户填写存款单后由银行业务员输入系统，系统确认储户信息后让储户输入密码，系统将记录存款人存款信息，并打印输出存款单让储户签字。如果储户要取款，储户填写取款单后由银行业务员输入系统，系统确认储户信息后让储户输入密码，系统计算利息打印输出利息清单并让储户签字。该系统的存款流程图如图 3-2 所示，取款流程图如图 3-3 所示。

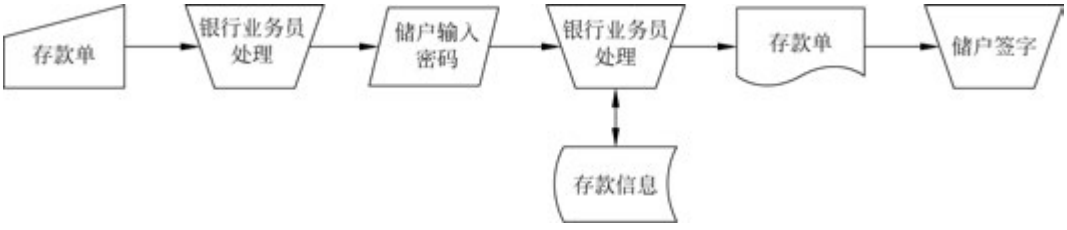


图 3-2　银行存取款系统的存款流程图

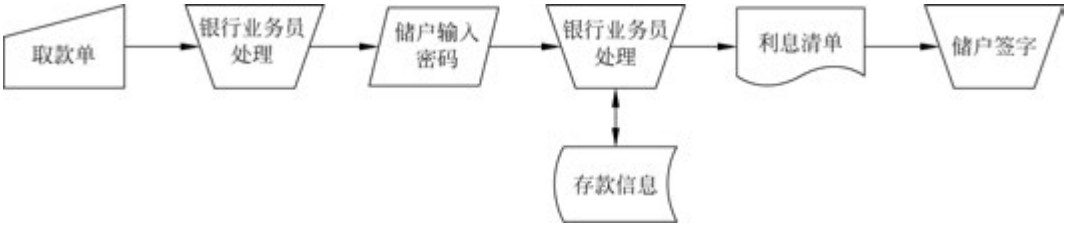


图 3-3　银行存取款系统的取款流程图

3.2.2 数据流图

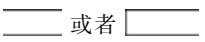
数据流图(Data Flow Diagram,DFD)是一种图形化技术,它能够直观地展现信息在系统中的流动和处理过程。通过数据流图,可以清晰地看到数据如何从系统的输入端流入,经过一系列的转换和处理,最终到达输出端。在数据流图中没有任何具体的物理部件,它只是展示数据在软件中的逻辑流动和处理过程。数据流图是系统逻辑功能的图形表示,非计算机专业人员也很容易理解它,因此它是分析员与用户之间极好的通信工具,也是分析员与用户之间沟通的桥梁。

设计数据流图时只需考虑系统必须完成的基本逻辑功能,而无须考虑具体的实现细节。通过数据流图,开发团队可以明确系统的功能和数据处理流程,为后续的软件设计提供清晰的指导。

1. 数据流图的基本元素

数据流图有 4 种基本元素:数据源点和终点、数据处理、数据存储和数据流。数据流图的基本元素如表 3-2 所示。

表 3-2 数据流图的基本元素

元 素	符 号	说 明
数据源点和终点		数据的源点或终点。用于展示系统中数据的来源和去处
数据处理		对数据进行处理的单元,它接收一定的数据输入,对其进行处理,并产生输出。用于描述数据在系统中所经历的处理或变换
数据存储		静态存储,可以代表文件、文件的一部分、数据库的元素等。用于为数据处理提供所需的输入数据或为处理后的数据提供存储仓库
数据流		数据在系统内流动的路径。箭头方向表示数据流动的方向,数据流名称标注在数据流线上面。用于展示数据从一处流向另一处的路径

2. 画数据流图

采用数据流图来描绘系统中某一特定层级的数据处理流程,是一种直观且高效的方法。然而,当面对一个错综复杂的系统或问题时,试图通过单一的数据流图来全面呈现其数据处理的全貌,是比较困难的。要想有效表达这类复杂问题的数据处理流程,需要根据问题的内在层次结构进行逐步分解,通过构建一套分层的数据流图体系来反映这种多层次、多维度的结构关系。

1) 画系统的输入输出

画系统的输入输出,就是确定数据源点和终点。把系统视为一个整体,看这个整体与外界的联系,分析哪些内容是要通过外界获取的,即系统的输入;哪些内容是要向外界提供服务的,即系统的输出。为了直观地表达这一过程,首先绘制顶层数据流图。

顶层数据流图是数据流图中的一个关键层次,它代表了系统的最高层次抽象,主要用于展示系统的整体概览和与外部实体之间的数据流动关系,它仅包含一个处理,用以代表整个

被开发的系统。绘制顶层数据流图时考虑该系统有哪些输入数据，这些输入数据从哪里来，有哪些输出数据，输出到哪里去，这样就定义了系统的输入输出数据流。顶层数据流图的作用在于表明被开发系统的范围以及它与周围环境的数据交换关系。顶层数据流图只有一张。

2）画系统内部

数据流图主要用于详尽地描述系统内部的数据处理流程。数据流图通过分层的方式，将复杂的处理过程逐步细化为更易于理解和管理的部分。绘制数据流图时，通常遵循自顶向下、逐层细化的原则，从最高层次的顶层数据流图开始，逐步深入更具体的下层数据流图。

当发现某个子系统内部的处理流程较为复杂时，需要进一步绘制该子系统的下层数据流图。在这一阶段，将子系统中的每一个处理视为潜在的分解点，根据数据流的实际流向和逻辑需求，继续将处理过程分解为更小的、更具体的处理单元。这一过程重复进行，直至每个处理都足够简单，无须进一步分解为止。

【案例 3-3】 案例 3-2 中银行存取款系统的数据流图如图 3-4 所示。

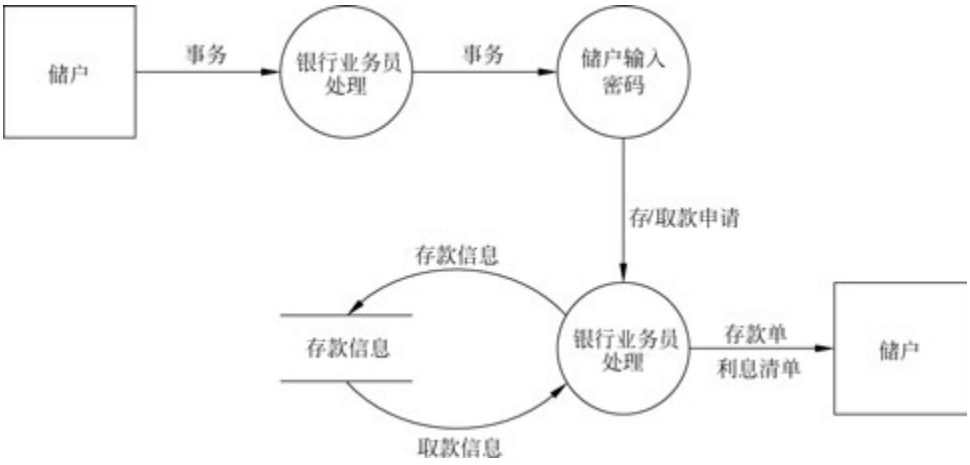


图 3-4　案例 3-2 中银行存取款系统的数据流图

【案例 3-4】 工厂的采购部每天需要一张订货报表，表中列出所有需要再次订货的零件。零件入库或出库称为事务，仓库管理员通过放在仓库中的终端把事务报告给订货系统。当某种零件的库存数量少于库存量临界值时就应该再次订货。订货系统的顶层数据流图如图 3-5 所示，订货系统的一层数据流图如图 3-6 所示，订货系统的二层数据流图如图 3-7 所示。

图 3-5　订货系统的顶层数据流图

3.2.3　数据字典

数据字典（Data Dictionary，DD）是关于数据的信息的集合，是对数据流图中包含的所有元素的定义的集合。它提供了对数据元素的详细描述和定义，以及数据的结构、关系和属

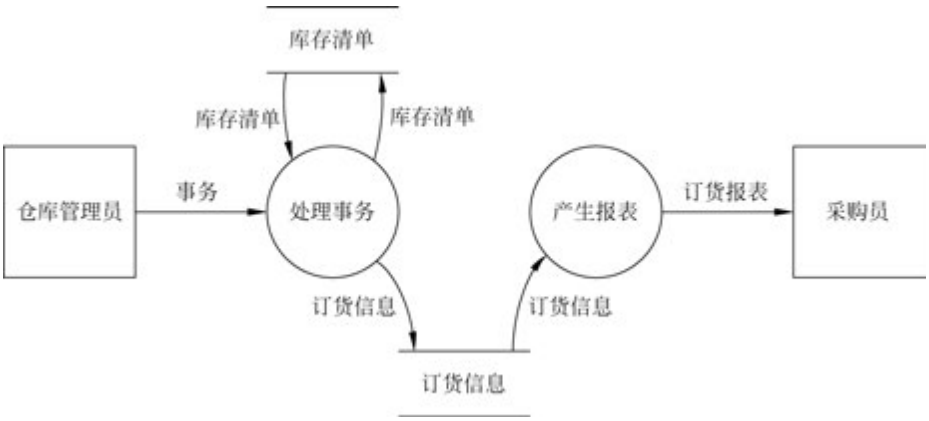


图 3-6　订货系统的一层数据流图

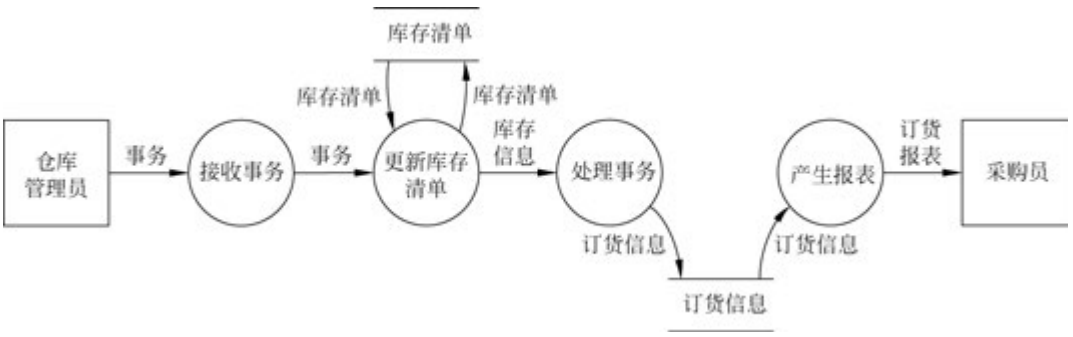


图 3-7　订货系统的二层数据流图

性等信息。它通过对数据元素的详细定义和描述，帮助用户更好地理解和使用数据，提高数据的一致性、可靠性和可用性。

为了完整地描述系统，只用数据流图是不够的，需要借助数据字典对图中的数据和处理给出解释。

数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程 5 个部分。

1. 数据项

数据项是不可再分的数据单位。数据项的描述通常包括以下内容：

数据项描述＝{数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系，数据项之间的联系}

2. 数据结构

数据结构反映了数据之间的组合关系。一个数据结构可以由若干数据项组成，也可以由若干数据结构组成，还可以由若干数据项和数据结构混合组成。数据结构的描述通常包括以下内容：

数据结构描述＝{数据结构名，数据结构含义说明，组成：{数据项或者数据结构}}

3. 数据流

数据流是数据结构在系统内传输的路径。数据流的描述通常包括以下内容：

数据流描述＝{数据流名，说明，数据流来源，数据流去向，组成：{数据结构}，平均流量，高峰期流量}

4. 数据存储

数据存储是数据结构停留或保存的地方，也是数据流来源和数据流去向之一。数据存储的描述通常包括以下内容：

数据存储描述＝{数据存储名，说明，编号，输入的数据流，输出的数据流，组成：{数据结构}，数据量，存取频度，存取方式}

5. 处理过程

处理过程的具体处理逻辑一般用判定表或者判定树进行描述。在数据字典中只需要描述处理过程的说明性信息。处理过程的描述通常包括以下内容：

处理过程描述＝{处理过程名，说明，输入：{数据流}，输出：{数据流}，处理：{简要说明}}

3.2.4 实体-联系图

实体-联系图(Entity-Relationship Diagram，E-R 图)是用于描述现实世界的概念模型的一种图形表示方法。为了把用户的数据要求清楚、准确地描述出来，系统分析员通常建立一个概念性的数据模型。概念性数据模型是按照用户的观点对数据建立的模型，它描述了从用户角度看到的数据，反映了用户的现实环境，有助于设计者更直观地理解业务需求。

E-R 图包含三个基本要素：实体、属性和联系。

实体(Entity)：现实世界中可区分、可识别的对象或事物。在 E-R 图中，实体通常用矩形表示，并在框内注明实体名称。例如，人、学生、课程等都可以作为实体。

属性(Attribute)：实体所具有的某一特性。在 E-R 图中，属性通常用椭圆表示，并将属性名记入框中。例如，学生的属性可能包括姓名、学号、年龄等。通常带有下画线的属性表示主键。

联系(Relationship)：实体之间的关联或相互作用。在 E-R 图中，联系通常用菱形表示，并在框内注明联系名称。实体之间的联系可以是一对一(1∶1)、一对多(1∶N)或多对多(M∶N)类型。

【案例 3-5】 假设教学管理规定：一名学生可选修多门课程，一门课程可以有多名学生选修；一名教师可教授多门课，一门课可以有多名教师教授。学生的属性有学号、姓名、性别、年龄；教师的属性有教师编号、姓名、性别、年龄、职称；课程的属性有课程号、课程名、课时、学分。根据上述语义画出选课 E-R 图如图 3-8 所示。

3.2.5 应用案例——图书借阅管理系统结构化需求分析

对图书借阅管理系统进行结构化需求分析。

1. 系统概述

图书借阅管理系统是一个用于图书馆日常图书管理和借阅操作的软件工具。图书借阅管理系统通过整合图书馆的借阅流程，极大地提升了图书管理的效率，改善了读者的借阅体验。系统为不同用户角色提供定制化功能，确保了操作的便捷性和数据的安全性。对于图书管理员而言，它简化了日常管理工作，减少了手动处理的烦琐性；对于读者，它提供了快速查询和自助借还服务，提高了借阅的便捷性。此外，系统管理员通过对用户权限的精细管理，保障了系统的稳定运行。该系统不仅促进了图书馆资源的流通，也增强了图书馆服务的吸引力，对于提升图书馆服务质量和运营效率具有重要意义。

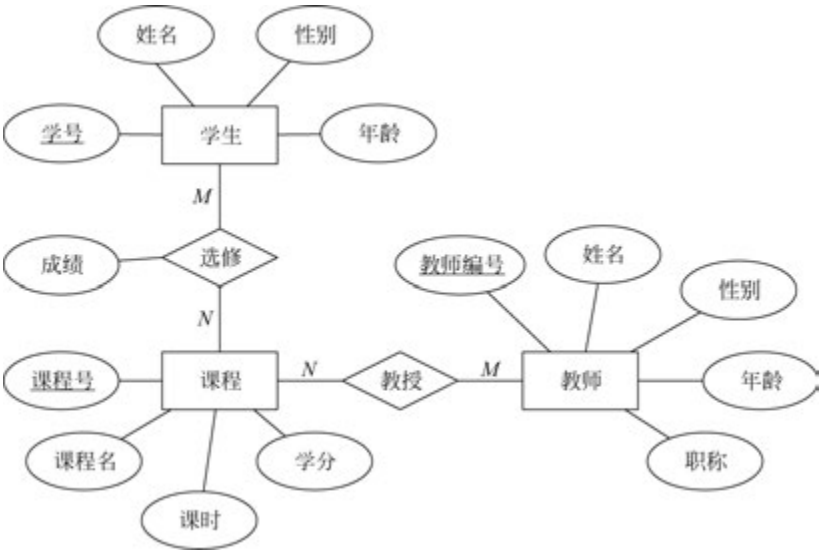


图 3-8　选课 E-R 图

2. 系统用户角色

系统支持三种用户角色：读者、图书管理员和系统管理员。

(1) 读者：图书馆服务的最终用户，可以执行登录、查询图书、借书、还书。

(2) 图书管理员：负责图书馆的日常图书管理工作，可以执行登录、添加新书、删除旧书、借出图书和接收归还的图书。

(3) 系统管理员：负责整个系统的维护和管理工作，可以执行登录、用户管理、权限管理。

3. 系统功能

(1) 登录

支持三种角色的登录，验证用户姓名和密码的正确性；

根据用户角色分配不同的操作权限。

(2) 图书管理(图书管理员使用)

添加新书：输入图书的基本信息(如图书 ID、书名、作者、出版社、出版日期、总库存图书数量、可借阅图书数量)，并保存到数据库；

删除旧书：删除不再需要的图书记录；

借出图书：输入图书信息，记录读者信息和借阅日期；

接收归还的图书：输入归还的图书信息，记录归还日期。

(3) 借阅管理(读者使用)

查询图书：根据图书 ID、书名、作者等条件查询图书信息；

借书：选择需要借阅的图书，系统记录读者的信息，借出书的信息和借阅的日期；

还书：归还借阅的图书。

(4) 成员管理(系统管理员使用)

用户管理：创建、修改、删除用户，用户包括图书管理员和读者；

权限管理：为每种用户分配不同的操作权限，确保系统的安全性。

4. 系统流程图

借书流程图如图 3-9 所示。

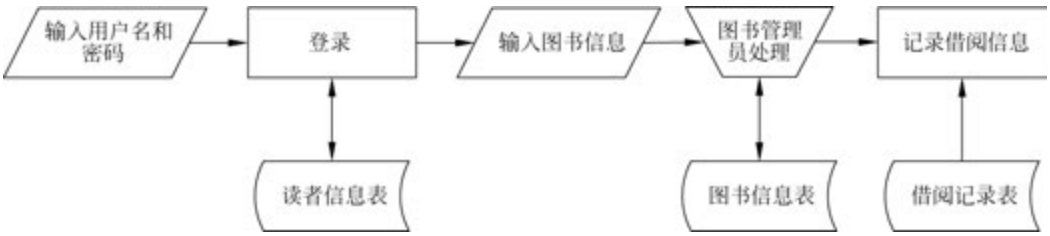


图 3-9　借书流程图

还书流程图如图 3-10 所示。

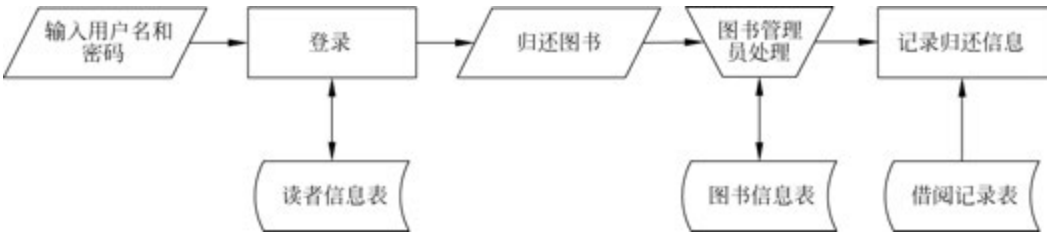


图 3-10　还书流程图

5. 系统数据流图

顶层数据流图如图 3-11 所示。

图 3-11　顶层数据流图

一层数据流图如图 3-12 所示。

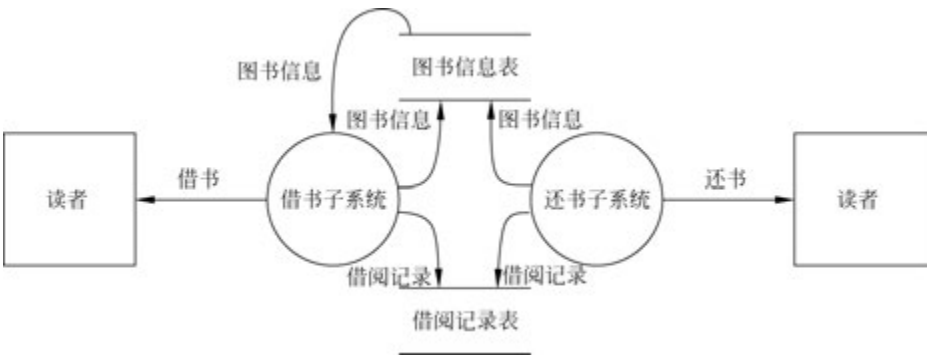


图 3-12　一层数据流图

二层数据流图如图 3-13(借书)、图 3-14(还书)所示。

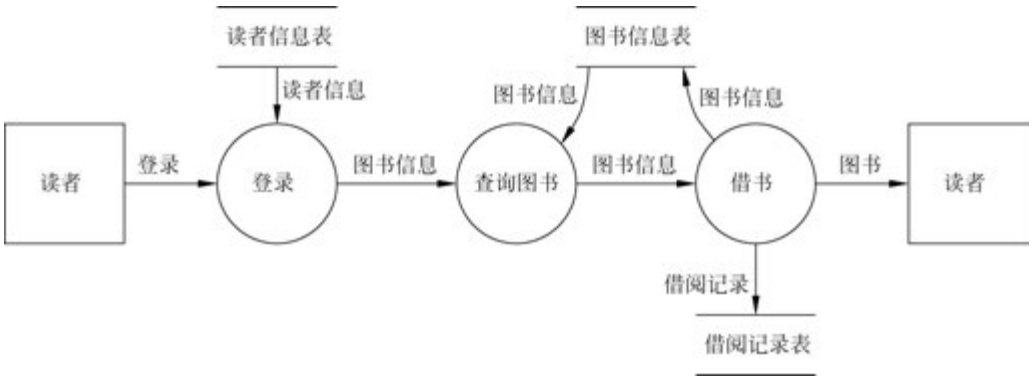


图 3-13　二层数据流图(借书)

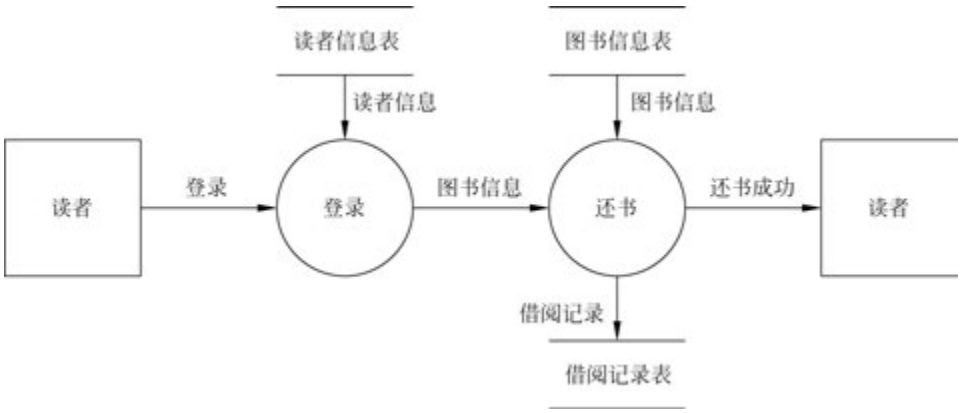


图 3-14　二层数据流图(还书)

6. 系统数据字典

(1) 数据项

系统管理员的数据项如表 3-3 所示,图书信息的数据项如表 3-4 所示,借阅记录的数据项如表 3-5 所示,图书管理员的数据项如表 3-6 所示,读者信息的数据项如表 3-7 所示。

表 3-3　系统管理员的数据项

数据项名	数据项含义	数据类型	长度	是否可空	逻辑关系
系统管理员 ID	系统管理员 ID	int	4	否	主键
姓名	姓名	varchar	10	否	
密码	密码	varchar	10	否	
角色	角色	int	4	否	

表 3-4　图书信息的数据项

数据项名	数据项含义	数据类型	长度	是否可空	逻辑关系
图书 ID	图书 ID	int	4	否	主键
书名	书名	varchar	20	否	
作者	作者	varchar	20	否	
出版社	出版社	varchar	20	否	
出版日期	出版日期	DATE		否	

续表

数据项名	数据项含义	数据类型	长度	是否可空	逻辑关系
库存数量	库存数量	int	4	否	
可借阅数	可借阅数	int	4	否	

表 3-5　借阅记录的数据项

数据项名	数据项含义	数据类型	长度	是否可空	逻辑关系
读者 ID	读者 ID	int	4	否	外键
图书 ID	图书 ID	int	4	否	外键
借阅日期	借阅日期	DATE		否	
归还日期	归还日期	DATE		否	

表 3-6　图书管理员的数据项

数据项名	数据项含义	数据类型	长度	是否可空	逻辑关系
图书管理员 ID	图书管理员 ID	int	4	否	主键
姓名	姓名	varchar	10	否	
密码	密码	varchar	10	否	
角色	角色	int	4	否	

表 3-7　读者信息的数据项

数据项名	数据项含义	数据类型	长度	是否可空	逻辑关系
读者 ID	读者 ID	int	4	否	主键
姓名	姓名	varchar	10	否	
密码	密码	varchar	10	否	
角色	角色	int	4	否	

（2）数据结构

图书借阅管理系统的数据结构如表 3-8 所示。

表 3-8　图书借阅管理系统的数据结构

数据结构名	数据结构含义	组成
系统管理员	系统管理员	ID、姓名、密码、角色
图书信息	图书信息	图书 ID、书名、作者、出版社、出版日期、库存数量、可借阅数
借阅记录	借阅记录	读者 ID、图书 ID、借阅日期、归还日期
图书管理员	图书管理员	ID、姓名、密码、角色
读者信息	读者信息	ID、姓名、密码、角色

（3）数据流

图书借阅管理系统的数据流如表 3-9 所示。

表 3-9　图书借阅管理系统的数据流

数据流名	数据流来源	数据流去向	组成
用户登录信息	用户输入	登录模块	用户名、密码
用户权限验证	登录模块	权限管理模块	用户角色、权限
图书信息录入	图书管理员	图书管理模块	图书基本信息
图书信息删除	图书管理员	图书管理模块	图书 ID

续表

数据流名	数据流来源	数据流去向	组成
图书借阅记录	图书管理员	借阅管理模块	读者信息、图书信息、借阅日期
图书归还记录	图书管理员	借阅管理模块	归还日期
图书查询请求	读者	借阅管理模块	查询条件
图书借阅请求	读者	借阅管理模块	读者信息、借阅图书信息、借阅日期
图书归还请求	读者	借阅管理模块	归还图书信息
用户管理信息	系统管理员	成员管理模块	用户信息
用户权限分配	系统管理员	成员管理模块	用户角色、权限

(4) 数据存储

图书借阅管理系统的数据存储如表 3-10 所示。

表 3-10　图书借阅管理系统的数据存储

数据存储名	输入的数据流	输出的数据流	组成
系统管理员表	系统管理员 ID、姓名、密码、角色	查询、添加、修改、删除系统管理员信息	系统管理员 ID、姓名、密码、角色
图书信息表	图书 ID、书名、作者、出版社、出版日期、库存数量、可借阅数	查询、添加、删除图书信息	图书 ID、书名、作者、出版社、出版日期、库存数量、可借阅数
借阅记录表	读者 ID、图书 ID、借阅日期、归还日期	记录借阅和归还信息	读者 ID、图书 ID、借阅日期、归还日期
图书管理员表	图书管理员 ID、姓名、密码、角色	查询、添加、修改、删除图书管理员信息	图书管理员 ID、姓名、密码、角色
读者信息表	读者 ID、姓名、密码、角色	查询、添加、修改、删除读者信息	读者 ID、姓名、密码、角色

(5) 处理过程

图书借阅管理系统的处理过程如表 3-11 所示。

表 3-11　图书借阅管理系统的处理过程

处理过程名	输入数据流	输出数据流组成
用户登录	用户名、密码	登录成功/失败信息、用户角色
添加新书	图书信息	添加成功/失败信息、更新后的图书列表
删除旧书	待删除图书 ID	删除成功/失败信息、更新后的图书列表
借出图书	读者信息、图书信息	借出成功/失败信息、借阅记录、更新后的图书可借阅数量
接收归还的图书	归还的图书信息	归还成功/失败信息、更新借阅记录、更新后的图书可借阅数量
查询图书信息	查询条件	图书列表
借书	选择的图书信息	借书成功/失败信息、借阅记录、更新后的图书可借阅数量
还书	归还的图书信息	还书成功/失败信息、更新借阅记录、更新后的图书可借阅数量
用户管理	用户信息	管理成功/失败信息、更新后的用户列表
权限管理	用户角色、操作权限设置	设置成功/失败信息、更新后的用户权限

7. 系统实体-联系图

系统实体-联系图如图 3-15 所示。

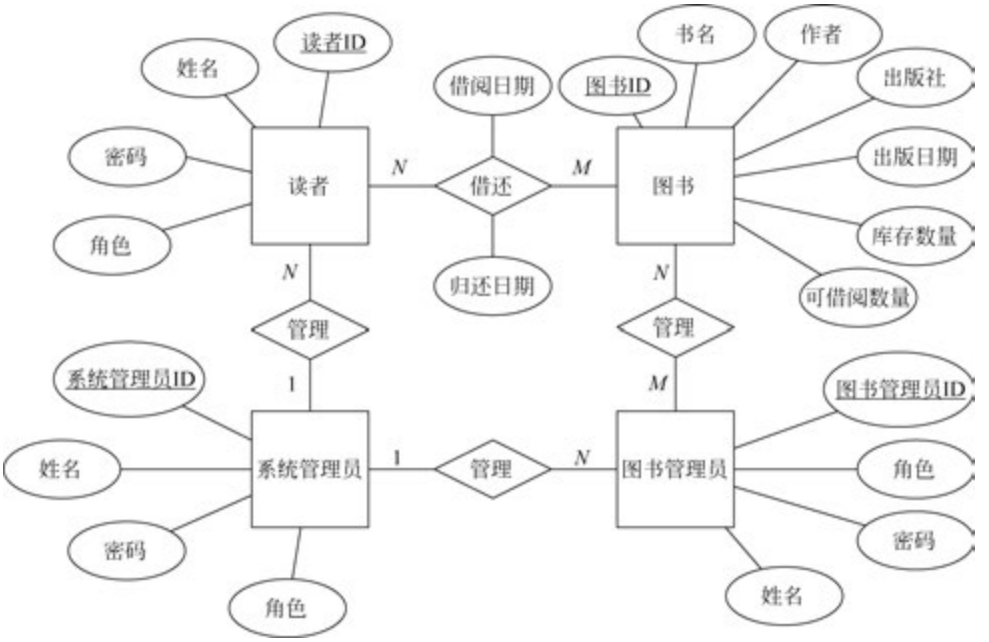


图 3-15　系统实体-联系图

3.3　面向对象方法

面向对象分析(Object-Oriented Analysis, OOA)是一种面向对象的系统分析方法,它运用对象、类、继承、封装、聚合、关联、消息、多态性等面向对象概念进行系统分析。具体来说,面向对象分析将问题域中的事物抽象为系统中的对象,识别对象的特征和各类对象之间的关系,进而建立一个能映射问题域且满足用户需求的面向对象分析模型,即 OOA 模型。

在面向对象的分析中,认为客观世界中的任何事物都可以被看作一个对象,这些对象之间存在一定的关系。用对象的属性来描述事物的数据特征,用对象的操作来描述对象的行为特征。属性和操作结合为一体,形成一个独立的、不可分的实体。对象对外屏蔽其内部细节,只提供有限的接口与外部进行交互,从而提高了数据的安全性和系统的可维护性。通过抽象对事物进行分类,将具有相同属性和操作的对象归为一类,形成类。复杂的对象可以由简单的对象作为其构成部分,同时,特殊类可以继承一般类的属性和操作,从而实现代码的复用和扩展。对象之间通过消息进行通信,以实现对象之间的动态联系。这种通信方式使得系统更加灵活和易于扩展。

面向对象分析的主要任务是建立一个准确的、一致的系统模型用于描述软件需要解决的问题。具体来说,其任务包括以下方面:识别问题域中的对象,通过分析问题域的业务逻辑和数据流程,识别出关键的对象和实体;定义对象的属性和方法,为每个对象定义其属性和方法,以描述对象的静态特征(属性)和动态行为(方法);确定对象之间的关系,分析对象之间的关联、依赖、继承等关系,以构建系统的静态结构模型;构建系统的逻辑模型,在分析

与综合过程中逐步细化软件功能，划分各个子功能，同时对数据域进行分解，确定系统的构成及主要成分，并用图文结合的形式建立起新系统的逻辑模型。

3.3.1 用例图

用例图是 UML 中的一种行为图，用于描述系统的功能和用户（或其他外部实体）与系统之间的交互。用例图通过参与者、用例以及它们之间的关系，来展示系统的功能需求和行为。具体来说，参与者是系统外部的角色、用户或实体，与系统交互执行用例；用例则表示系统的一个功能或一个特定的用户操作，描述了系统对外部参与者提供的一种行为。在用例图中，参与者通常用人形图标表示，用例通常用椭圆形图标表示。创建用例图的步骤如下。

1. 确定参与者

确定系统与外部实体之间的交互参与者，如用户、外部系统或设备等。这些参与者是系统的用户或外部实体，与系统进行交互并对系统执行某些用例。

2. 确定用例

用例是描述系统功能或行为的情景。识别主要的用例，即系统中最重要和关键的功能或任务。每个用例应该表示一个用户或参与者与系统之间的特定目标或功能。

3. 绘制参与者和用例

使用 UML 建模工具，绘制用例图的主要框架。

4. 确定关系

确定参与者和用例之间的关系，以及用例之间的关系。

常见的关系类型包括关联关系、包含关系、扩展关系、泛化关系等。这些关系所对应的符号如表 3-12 所示。

表 3-12 用例图关系对应的符号

关　系	符　号	关　系	符　号
关联关系		扩展关系	
包含关系		泛化关系	

关联关系表示参与者与用例之间的关联。发起用例的参与者是主参与者，主参与者与用例之间用带实心三角形箭头的实线关联。关联关系如图 3-16 所示。

包含关系表示一个用例在执行过程中包含另一个用例的功能，通常用带开叉箭头的虚线表示，虚线上带有《include》标签，箭头指向被包含的用例。包含关系如图 3-17 所示。

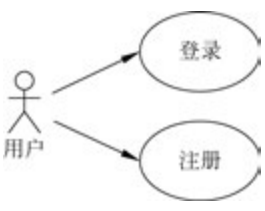


图 3-16 关联关系

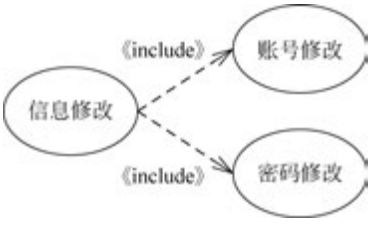


图 3-17 包含关系

扩展关系表示一个用例在某些条件下可以扩展另一个用例的功能，通常用带开叉箭头的虚线表示，虚线上带有《extend》标签，但箭头指向被扩展的用例。扩展关系如图 3-18 所示。

泛化关系表示用例之间的继承关系，用带空心三角形箭头的实线表示，箭头指向父用例。泛化关系如图 3-19 所示。

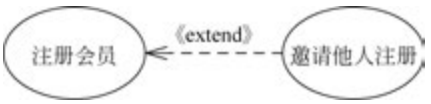


图 3-18　扩展关系

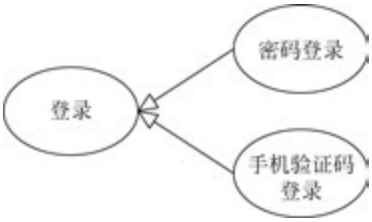


图 3-19　泛化关系

5. 添加注释和细节

根据需要，为参与者和用例添加注释和描述。这些注释和描述可以提供更详细的说明，帮助读者理解用例图的含义和功能。

6. 完善用例图

进一步完善用例图，包括调整参与者和用例的位置和布局，添加连线、箭头和标签以显示关系。确保用例图的清晰易读，能够准确地传达系统的功能和用户需求。

【案例 3-6】 机票预订系统是某航空公司推出的一款网上购票系统。用户登录系统后可以查询航班、购买机票、查看行程、退订机票。系统管理员可以安排系统中的航班信息。此外，该购票系统还与外部的一个信用评价系统有交互。当某用户一个月之内退订两次及以上的机票时，需要降低该用户在信用评价系统中的信用等级。当信用过低时，则不允许该用户再次购买机票。该机票预订系统的用例图如图 3-20 所示。

该用例图展示了一个机票预订系统的核心功能和扩展功能。主要用例包括“查询航班”“购买机票”“查看行程”和“退订机票”。这些用例代表了用户与系统交互的基本操作。其中，“购买机票”用例包含了“检查信用等级”用例，这意味着在用户购买机票时，系统需要先验证其信用等级。

此外，系统还提供了一些扩展功能，如“修改信用等级”和“设定航班信息”。这些用例在特定条件下触发。用户需要通过“登录”用例来访问系统，确保只有授权用户才能使用系统的功能。

3.3.2　类图及其关系

类图是面向对象建模的主要组成部分，用于显示模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其他类的关系等。类图不显示暂时性的信息，而是聚焦于系统的静态视图。它是最常用的 UML 图之一，用于描述系统的结构化设计，是构建其他设计模型的基础。类图不仅用于应用程序的系统分类的一般概念建模，也用于详细建模，将模型转换成编程代码，同时也可用于数据建模。

类图显示了系统中的类、属性和方法，以及类之间的关系。通过类图，可以清晰地表示出类的内部结构以及类与类之间的关系。图 3-21 是一个读者类。

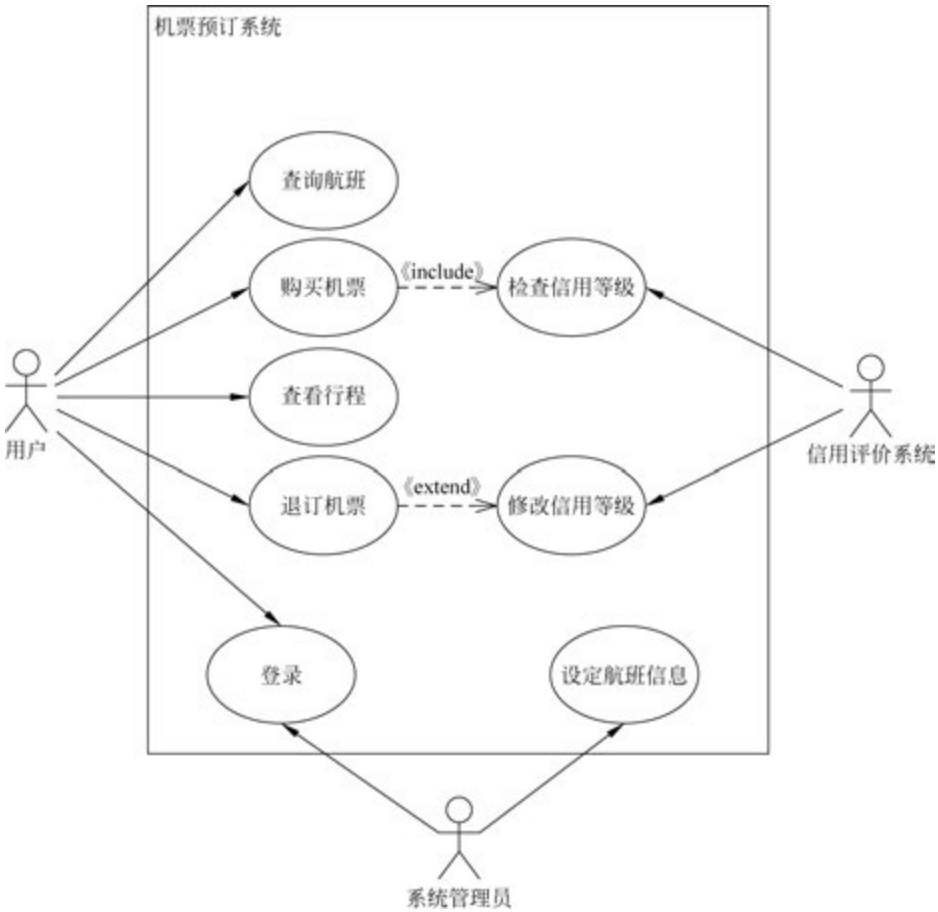


图 3-20　机票预订系统的用例图

读者
-用户ID：int -用户名：String -密码：String -角色：String
+登录 +查询图书 +借阅图书 +归还图书

图 3-21　读者类

读者类展示了读者包含的属性和方法，清晰地描述了一位读者的基本信息和行为。读者是类名。读者的属性包含用户 ID、用户名、密码、角色。读者的行为包含登录、查询图书、借阅图书、归还图书。

类之间的关系主要有继承/泛化关系、实现关系、关联关系、聚合关系、组合关系、依赖关系。这些关系所对应的符号如表 3-13 所示。

表 3-13　类图关系对应的符号

关　系	符　号	关　系	符　号
继承/泛化关系		聚合关系	
实现关系		组合关系	
关联关系		依赖关系	

继承/泛化关系表示一个类是另一个类的特殊类型，通常称为子类（或派生类）与父类（或基类）的关系。在类图中，通过一条带空心三角箭头的实线表示继承/泛化关系，箭头指向父类。继承/泛化关系如图 3-22 所示。

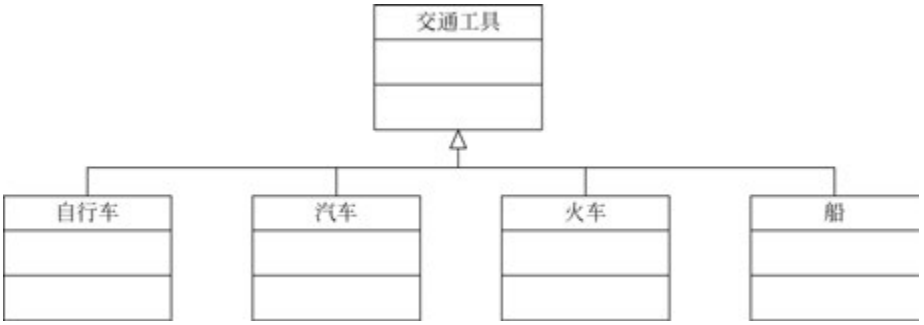


图 3-22　继承/泛化关系

“交通工具”是一个总的概念，代表所有类型的交通工具。下面的“自行车”“汽车”“火车”和“船”都是具体的交通工具，它们都属于交通工具的一种。图中的箭头指向“交通工具”，表示这些具体的交通工具都继承自“交通工具”这个大类。这种关系在面向对象编程中称为继承或泛化，意味着子类会继承父类的一些共同特征，同时可能还有自己独特的属性和功能。

实现关系表示一个类实现了一个接口。在类图中，通过一条带空心三角形箭头的虚线表示实现关系，箭头指向接口。实现关系如图 3-23 所示。

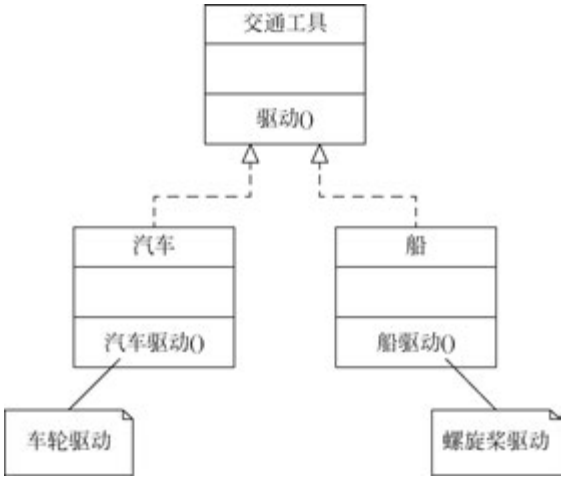


图 3-23　实现关系

“交通工具”这个父类包含了交通工具共有的一些属性和方法。“汽车”和“船”是“交通工具”的子类，它们继承了“交通工具”父类的属性和“驱动”方法。在“汽车”类中，“汽车驱动”这个方法负责具体实现汽车的驱动功能。同理，“船驱动”这个方法负责具体实现船的驱动功能。

关联关系表示两个类之间存在某种联系，这种联系可以是任何类型的关系，如“拥有”“使用”或“熟悉”等。在类图中，通过一条实线表示关联关系，可以带有名称、角色和重数。如果图中未明确标出关联的重数，则默认重数是 1。重数如表 3-14 所示。

表 3-14　重数

重　　数	说　　明
0…1	表示 0 到 1 个对象
0… * 或 *	表示 0 到多个对象
1＋或 1… *	表示 1 到多个对象
1…15	表示 1 到 15 个对象
3	表示 3 个对象

关联关系如图 3-24 所示。该图展示了教师和课程之间的关联关系。表示一名教师可以讲授多门课程，一门课程也可以由多名教师讲授。

聚合关系是“整体-部分”关系的一种弱形式，体现的是一种非强制的拥有关系。例如，项目组和成员之间的关系，项目组是由成员组成的，但成员不一定只属于一个项目组。在类图中，通过一条带空心菱形箭头的实线表示聚合关系，菱形指向整体类。聚合关系如图 3-25 所示。

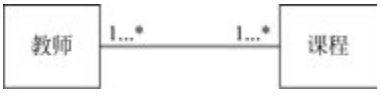


图 3-24　关联关系

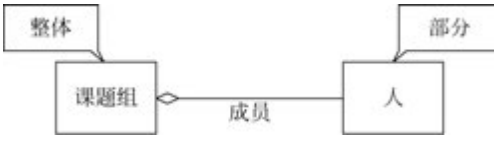


图 3-25　聚合关系

图 3-25 中，“课题组”代表整体，“人”代表部分。连接它们的是一条带空心菱形箭头的实线，菱形指向“课题组”，表明课题组由人组成。这种关系表明课题组包含了人，但这种包含关系并不是强制的或专属的。也就是说，一个人可以属于一个课题组，同时也可以属于其他课题组。聚合关系强调整体与部分之间的关联，同时又能保持部分的独立性。这种关系在描述组织结构、团队组成等场景时非常有用，因为它能够体现出组成部分与整体之间既有联系又相对独立的特点。

组合关系也表示“整体-部分”关系，但比聚合关系语义更强。在组合关系中，部分对象的生命周期由整体对象控制。像树和树枝的关系，树枝是树的一部分，并且随着树的“创建”而“创建”，随着树的“销毁”而“销毁”。在类图中，通过一条带实心菱形箭头的实线表示，菱形指向整体类。组合关系如图 3-26 所示。

图 3-26 展示了学生数据库与学生姓名表和学生成绩表之间的组合关系。学生数据库是主体，学生姓名表和学生成绩表是它不可分割的组成部分。图中的实心菱形箭头指向学生数据库，学生姓名表和学生成绩表这两部分完全依赖于学生数据库。也就是说，学生姓名表和学生成绩表随着学生数据库创建而创建，随着学生数据库删除而删除。

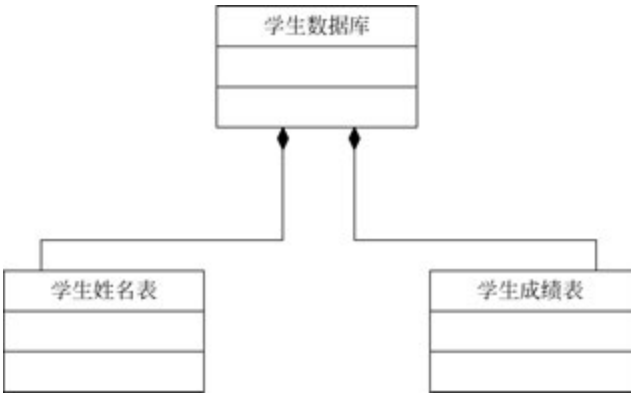


图 3-26　组合关系

依赖关系表示一个类的变化会影响到另一个类。通常是因为一个类使用了另一个类的方法或属性。在类图中，通过一条带开叉箭头的虚线表示，箭头指向被依赖的类。依赖关系如图 3-27 所示。

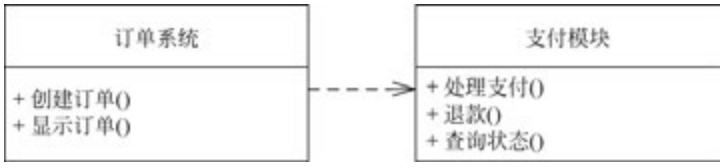


图 3-27　依赖关系

图 3-27 展示了订单系统和支付模块之间的依赖关系。带开叉箭头的虚线从订单系统指向支付模块，表明订单系统依赖于支付模块。订单系统包含创建订单和显示订单的功能。而支付模块则提供处理支付、退款和查询状态的功能。订单系统在处理订单时可能需要调用支付模块的方法，比如在创建订单时可能需要调用处理支付方法，或在显示订单时可能需要查询支付状态。这种依赖关系意味着如果支付模块的接口或实现发生变化，可能会影响到订单系统的功能。例如，如果支付模块修改了处理支付的方法，订单系统可能需要相应地更新其代码。这个例子展示了在软件设计中，不同模块之间如何通过依赖关系进行交互，同时也反映了模块化设计的重要性。

创建类图的步骤如下。

（1）研究分析问题领域确定系统需求：首先需要对所要建模的系统或问题进行全面的了解和分析，明确系统的需求和目标。

（2）确定类：根据系统需求，识别出系统中的类。类的识别需要明确类的含义和职责，以及类的属性和操作。

（3）确定类之间的关系：在确定了类之后，需要进一步分析类之间的关系，包括一般化关系、关联关系（包括聚合关系和合成关系）、依赖关系等。这些关系的确定有助于理解系统的整体结构和各个类之间的相互作用。

（4）绘制类图：根据以上步骤确定的信息，使用 UML 工具绘制类图。在类图中，需要清晰地表示出类的名称、属性、方法以及类之间的关系。

（5）审核和调整：完成类图的绘制后，需要进行审核和调整。这包括检查类图的准确

性、完整性以及是否符合系统的实际需求。如果发现问题或需要改进的地方，应及时进行修改和调整。

【案例 3-7】 在某出版系统中，存在出版社、图书、作者三个实体。一个出版社可能出版多本图书，一本书只能归属于一个出版社；一本图书可能有一个或者多个作者，一个作者可能出版一本或者多本图书。根据语义绘制类图来描述三者之间的关系，该类图如图 3-28 所示。

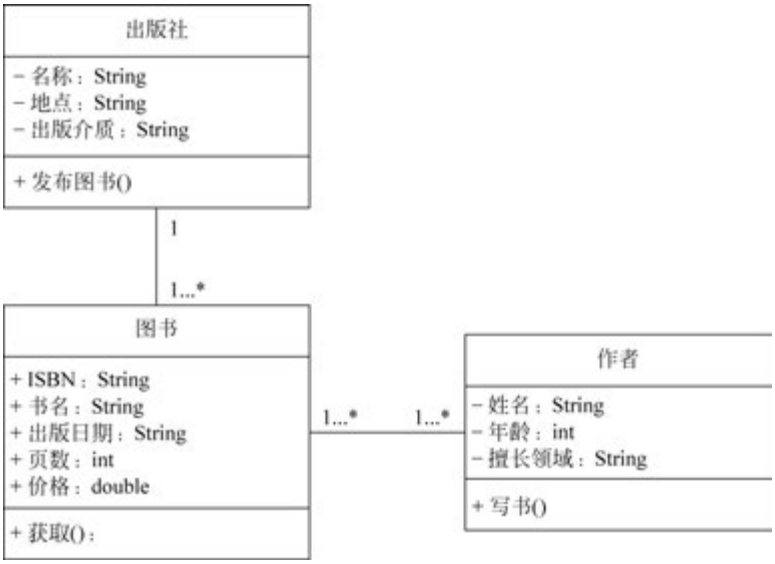


图 3-28　出版社、图书、作者三者的类图

3.3.3　时序图

时序图(Sequence Diagram)是一种 UML 交互图。它通过按时间顺序描述对象之间发送消息的过程来展示多个对象之间的动态协作。时序图可以表示用例的行为顺序，当执行一个用例行为时，其中的每条消息对应一个类操作或状态机中引起转换的触发事件。时序图具有两个坐标轴：纵坐标轴显示时间，横坐标轴显示对象。在时序图中，对象之间的交互通过消息来表示，这些消息按照时间顺序排列，从而清晰地展示了对象之间的协作过程。

时序图的主要元素有对象、生命线、消息、控制焦点。这些元素所对应的符号如表 3-15 所示。

表 3-15　时序图元素对应的符号

元　素	符　号	元　素	符　号
对象		异步消息	
生命线		返回消息	
同步消息		控制焦点	

对象在时序图中表示为矩形框，框内包含对象的名称或类型。这些对象通常是类的实例，它们参与了交互过程。对象的名称或类型通常置于矩形框的顶部，以便清晰标识。对象在时序图中的位置体现了它们在交互中的角色和重要性。

生命线是一条垂直的虚线，从对象的底部向下延伸，表示对象的存在和生命周期。生命线的起点通常与对象的创建相对应，而终点则可能表示对象的销毁或交互的结束。在生命线上，可以通过不同的标记或注释来表示对象状态的变化或重要事件的发生。

消息是时序图中对象之间交互的基本单位，表示一个对象向另一个对象发送的某种请求或动作。消息在时序图中表示为箭头线，箭头指向接收消息的对象。消息上可以标注操作的名称、参数列表等信息，以明确消息的具体内容。消息的类型包括同步消息（带实心三角形箭头的实线）、异步消息（带实心三角形箭头的虚线）和返回消息（带开叉箭头的虚线）。

控制焦点表示在某个特定时间点哪个对象正在执行操作或处理消息。在时序图中，控制焦点通常通过生命线旁的小矩形框（称为“激活框”）来表示。当一个对象接收到消息并开始处理时，激活框会出现在该对象的生命线上，表示该对象当前处于活动状态。激活框的长度和位置反映了对象处理消息的时间段。

创建时序图的步骤通常包括以下几方面。

（1）确定交互过程的上下文：明确时序图需要展示的是哪个用例或业务场景下的交互过程。

（2）识别参与过程的交互对象：列出所有参与交互的对象，包括系统角色、子系统等。

（3）为每个对象设置生命线：在时序图中为每个对象绘制一条生命线，表示其存在的时间。

（4）从初始消息开始，依次画出随后消息：按照时间顺序，从初始消息开始，依次绘制出后续的消息。每个消息都应明确标出发送者和接收者，以及消息的内容和类型。

（5）考虑消息的嵌套和条件分支：如果交互过程中存在消息的嵌套或条件分支，应使用组合片段来表示。

（6）标示消息发生时的时间点：在需要的情况下，可以使用时间戳来标示消息发生时的时间点。

（7）说明时间约束的地点：如果交互过程受到时间约束的影响，应在时序图中进行说明。

（8）审核和调整：完成时序图的绘制后，进行审核和调整，确保时序图的准确性和清晰性。如果需要，可以根据实际情况进行修改和完善。

【案例 3-8】 某银行系统的取款执行顺序是：银行工作人员输入取款单，银行系统进行身份验证，身份验证通过后，数据库将取款数从存款额中扣除，完成扣除后，银行系统打印取款信息。该取款时序图如图 3-29 所示。

3.3.4 应用案例——图书借阅管理系统面向对象需求分析

对图书借阅管理系统进行面向对象需求分析。

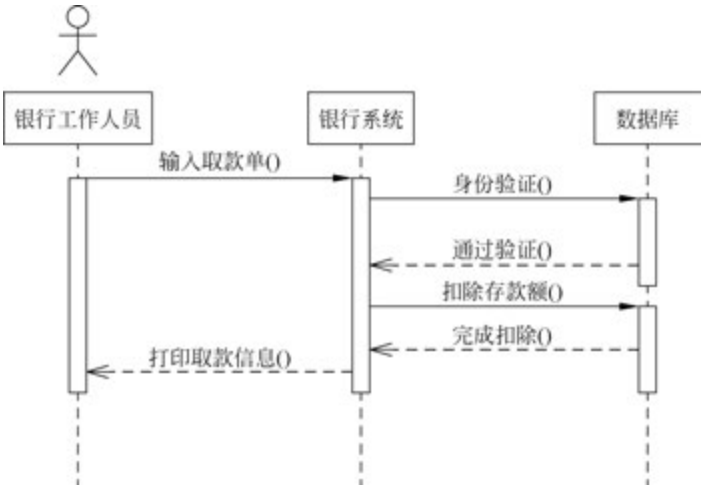


图 3-29　取款时序图

1. 系统用例图

1）确定参与者

首先，明确系统中所有的参与者(用户角色)。在图书借阅管理系统中，有三个主要的参与者：系统管理员，负责系统的整体维护和用户管理；图书管理员，负责图书的日常管理和借还操作；读者，图书馆的服务对象，可以查询和借还图书。

2）确定用例

列出系统中所有参与者可以执行的功能，这些功能即为用例。

系统管理员可以执行：登录、用户管理、权限管理。

图书管理员可以执行：登录、添加新书、删除旧书、借出图书、接收归还的图书。

读者可以执行：登录、查询图书、借书、还书。

3）确定用例之间的关系

使用线条连接参与者和他们对应的用例，表示这些参与者可以执行这些用例，并明确它们之间的关系。读者用例图如图 3-30 所示。图书管理员用例图如图 3-31 所示。系统管理员用例图如图 3-32 所示。

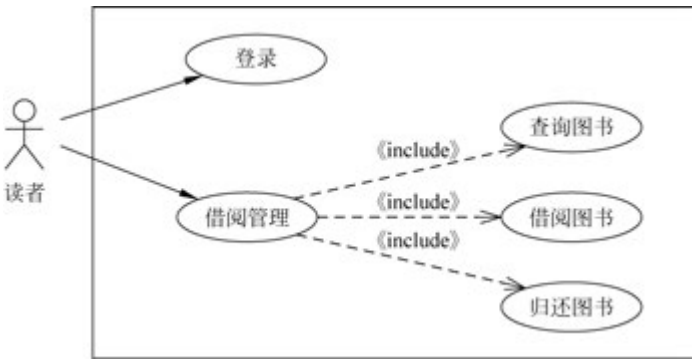


图 3-30　读者用例图

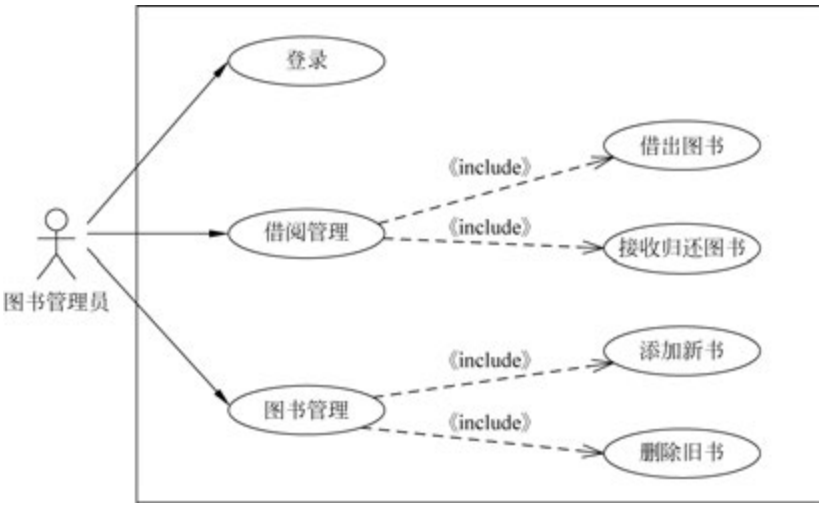


图 3-31　图书管理员用例图

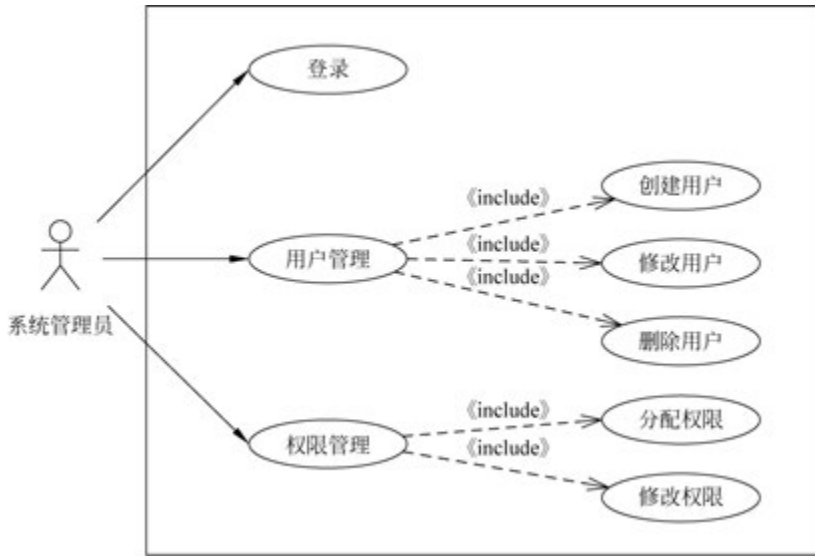


图 3-32　系统管理员用例图

2. 创建类图

根据系统功能描述，确定系统中的主要类有读者类、图书类、图书管理员类、系统管理员类，为每个类定义属性和方法。

1）读者类

属性：用户 ID、用户名、密码、角色。

方法：登录、查询图书、借书、还书。

2）图书类

属性：图书 ID、书名、作者、出版社、出版日期、库存数量、可借阅数量。

方法：添加新书、删除旧书、更新图书信息。

3）图书管理员类

属性：用户 ID、用户名、密码、角色。

方法：登录、添加新书、删除旧书、借出图书、接收归还的图书。

4）系统管理员类

属性：用户 ID、用户名、密码、角色。

方法：登录、创建账户、修改账户、删除账户、分配权限、修改权限。

在图书借阅管理系统中，类之间主要通过关联关系相互联系。图书借阅管理系统类图如图 3-33 所示。

图 3-33　图书借阅管理系统类图

3. 时序图

1）确定交互对象

明确参与该交互的对象。系统中交互对象有读者、图书管理员和系统管理员。它们作为参与者是这个交互过程的发起者。根据发起者的不同，参与这个交互的对象也有所不同。

2）添加消息

在确定参与交互的对象之后，就要在对象之间添加消息的传递了。通过分析系统的交互过程，并且向顺序图中添加消息后，创建出系统的顺序图。读者时序图如图 3-34 所示。图书管理员时序图如图 3-35 所示。系统管理员时序图如图 3-36 所示。

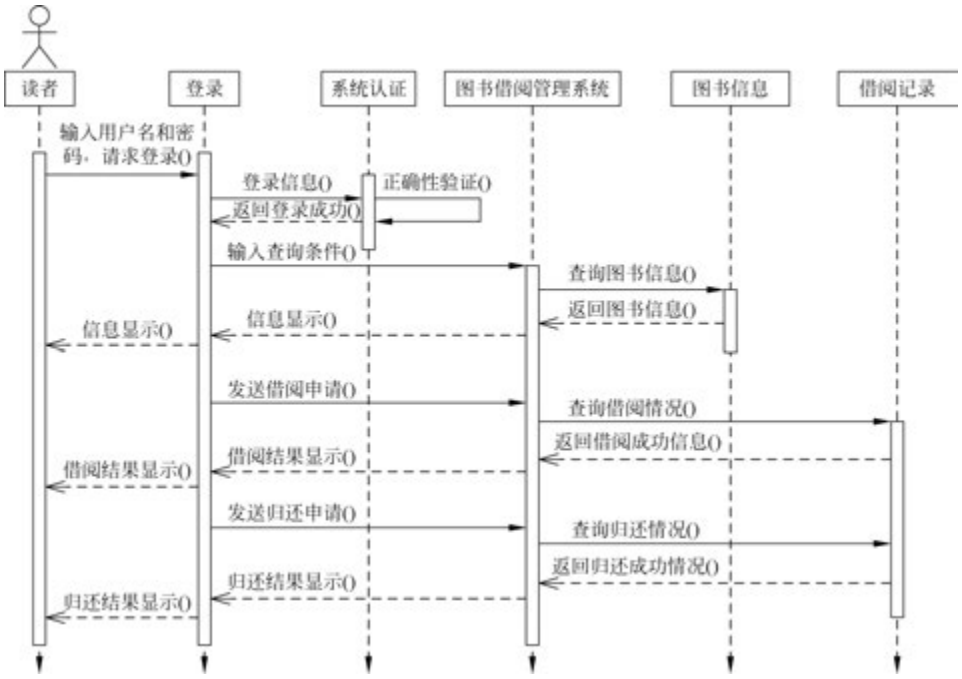


图 3-34　读者时序图

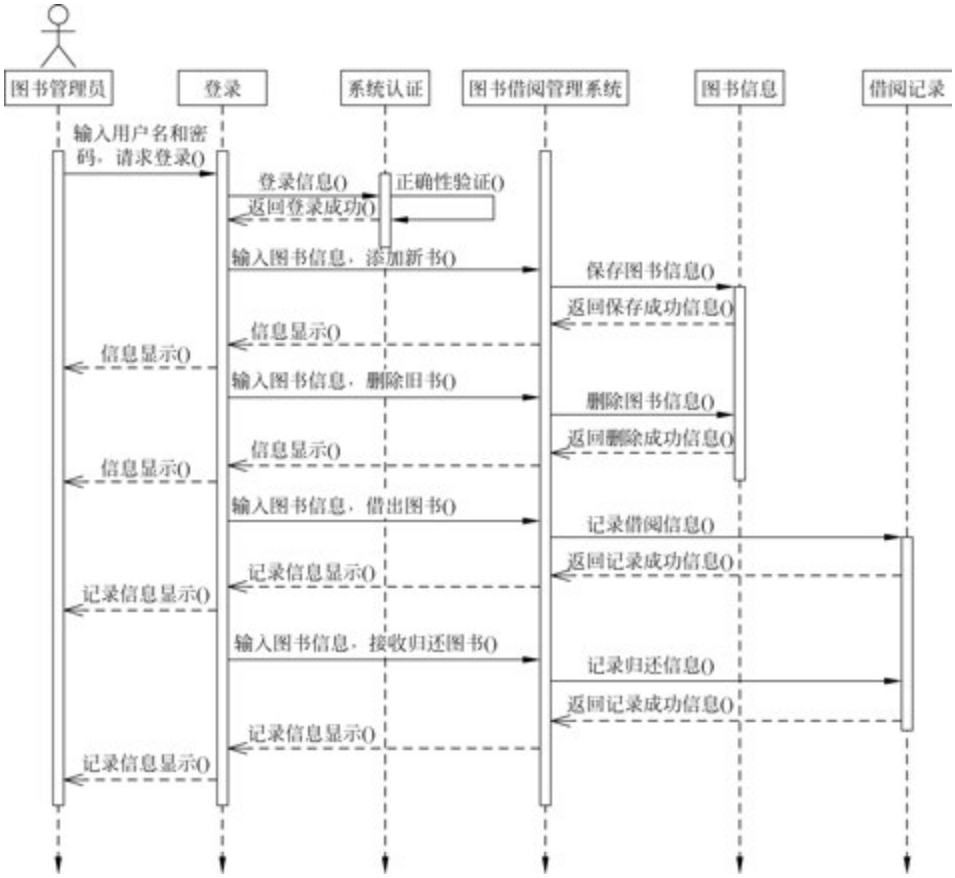


图 3-35　图书管理员时序图

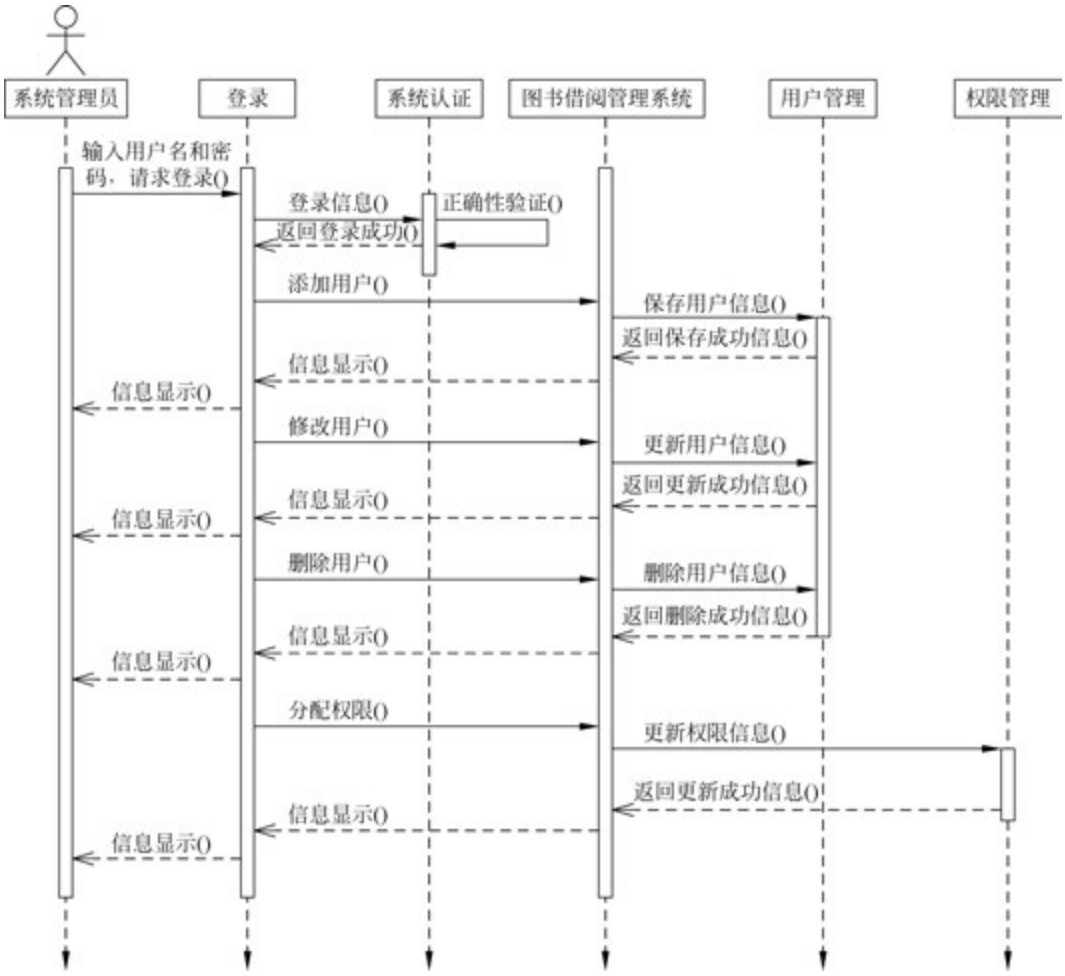


图 3-36　系统管理员时序图

3.4　小　　结

软件需求分析是软件定义时期的最终阶段，其核心任务是准确回答“系统必须做什么”这一根本问题。该过程以可行性研究为基础，通过系统化方法将模糊的业务诉求转化为结构化技术规范，为后续开发活动提供明确依据。需求分析的理论框架由原则体系、任务体系与方法体系构成，形成完整的分析范式。

需求分析原则包含四大核心方面：信息域建模聚焦业务数据的流动规律与加工机制，构建数据输入、处理与输出的全景视图；功能定义将业务需求转化为可执行的算法化服务，建立模块划分与层次分解的双维功能模型；行为描述从动态维度刻画系统对外部事件的响应机制，覆盖正常流程与异常处理场景；模型分解运用分层解耦策略降低系统复杂度，通过模块化、抽象层次与关注点分离原则构建可管理的模型体系。四者形成有机整体，信息域为功能定义提供数据基础，功能模型为行为描述确立操作主体，行为模型完善系统动态特征，模型分解确保分析可行性。

需求分析任务体系包含三个递进阶段：需求识别阶段需构建功能、性能、环境、安全与界面需求的多维矩阵，通过影响关系图揭示需求间的协同与约束；逻辑建模阶段采用数据流图、实体-联系图与状态转换图构建可视化模型，其中数据流图分层展现信息加工流程，实体-联系图描述数据实体间逻辑约束，状态转换图定义系统状态变迁规则；文档化阶段产出需求规格说明书、原型设计文档与变更记录，形成可验证、可追溯的需求基线，通过模块化结构与版本控制机制保障文档可维护性。

方法论层面形成结构化分析与面向对象分析双轨体系。结构化分析方法以数据流为核心驱动要素，通过系统流程图、数据流图、数据字典与实体-联系图构建层次化模型。系统流程图采用标准化符号描述物理交互框架，明确系统与外部实体的介质类型与数据传递方式；数据流图通过上下文图定义系统边界，逐层分解的加工节点展现功能逻辑，数据存储标识持久化数据集，数据流箭头刻画信息传递路径；数据字典对数据流、数据项、数据结构进行元数据管理，确保数据语义的精确性；实体-联系图通过实体、属性与联系三元组建立概念数据模型，为数据库设计提供范式化基础。该方法遵循“高内聚、低耦合”原则，通过自顶向下分解策略确保模块独立性。

面向对象分析方法以对象抽象为基础，运用用例图、类图与时序图构建多维模型。用例图通过参与者与用例的交互关系定义系统功能边界，利用包含关系实现功能复用，借助扩展关系处理条件分支，通过泛化关系建立用例继承体系；类图描述静态结构，类包含属性与方法定义，继承关系实现层次分类，聚合与组合表达整体-部分关系，关联关系描述业务连接，依赖关系标识临时性交互；时序图通过对象生命线与消息传递序列展现动态协作，同步消息表示直接调用，异步消息支持非阻塞交互，返回消息完成操作闭环，控制焦点标识对象激活时段。该方法通过封装、继承与多态机制提升系统扩展性，实现业务概念到技术模型的直接映射。

需求规格文档体系采用结构化表达框架：功能需求按原子性原则拆解为可验证单元，描述触发条件、处理逻辑与异常机制；性能需求建立可量化的时效、吞吐量与容量指标；环境需求定义硬件配置、软件依赖与网络协议；安全需求构建加密认证、访问控制与审计追踪体系；界面需求规范交互流程与视觉要素。文档嵌入需求追踪矩阵，关联原始诉求与技术实现，通过多级评审机制确保质量。变更管理流程涵盖影响分析、版本控制与追溯机制，维护需求基线稳定性。

逻辑建模工具的选择遵循系统特征适配原则：数据流图适用于流程明确的信息处理系统，实体-联系图侧重数据关系结构化表达，状态转换图适合生命周期明确的系统；类图构建领域静态模型，时序图分析交互时序逻辑，用例图定义功能服务边界。模型验证通过形式化检查与场景推演确保一致性，需求的可验证性通过量化的验收标准实现。

结构化分析方法与面向对象分析方法在实践中形成互补。结构化分析方法通过数据流显式控制确保核心处理逻辑的严谨性，面向对象分析方法通过对象抽象提升交互模型的灵活性。现代需求工程趋向于融合应用，在数据处理环节采用结构化建模，在用户交互层面应用面向对象技术，形成兼顾规范性与扩展性的混合框架。这种演进方向既继承传统方法的系统化分解策略，又吸收对象技术的抽象优势，为复杂系统需求分析提供有效支撑。

需求分析的质量保障贯穿全过程：信息域建模需完整映射数据生命周期，功能定义保持适中的抽象粒度，行为描述预设容错机制，模型分解制定清晰的接口规范。文档体系通过

模块化结构提升可维护性，原型设计平衡保真度与迭代效率，变更管理建立版本树实现历史追溯。多维度验证机制覆盖逻辑一致性、场景完整性与技术可行性等方面，为后续开发奠定坚实基础。

最终形成的需求分析成果，既是连接业务目标与技术实现的工程蓝图，也是协调多方认知的沟通媒介。通过系统化的原则指导、结构化的任务执行与科学的方法选择，将用户诉求转化为可执行的开发指令，确保软件产品准确满足预期目标。随着系统复杂度的持续提升，需求分析的理论框架与方法工具将持续演进，但其核心价值始终聚焦于回答“系统必须做什么”这一本质问题，为软件工程提供可靠的逻辑起点。

习　题　3

一、单项选择题

1. 需求分析的基本任务是准确回答什么问题？（　　）
   A. 系统必须做什么　　B. 系统如何实现
   C. 系统需要哪些硬件　　D. 系统开发周期
2. 需求分析原则中的信息域建模不包括以下哪项内容？（　　）
   A. 数据流动路径　　B. 数据取值范围
   C. 功能模块划分　　D. 数据实体关联
3. 功能需求建模的关键在于保持什么特性？（　　）
   A. 高耦合性　　B. 功能粒度适中性
   C. 代码复用性　　D. 界面美观性
4. 系统流程图的主要作用是描述什么？（　　）
   A. 系统逻辑处理流程　　B. 系统与外部实体的物理交互
   C. 数据库设计　　D. 用户界面布局
5. 数据流图的基本元素不包括以下哪项？（　　）
   A. 数据源点　　B. 数据存储
   C. 控制流程　　D. 数据处理
6. 数据字典中对数据流的描述不包括以下哪项？（　　）
   A. 平均流量　　B. 数据来源
   C. 加密算法　　D. 数据结构组成
7. 实体-联系图中表示“一对多”关系的符号是（　　）。
   A. 矩形　　B. 菱形
   C. 椭圆　　D. 箭头
8. 结构化分析方法的核心建模工具是（　　）。
   A. 用例图　　B. 数据流图
   C. 类图　　D. 时序图
9. 面向对象分析中，类图的继承关系使用什么符号表示？（　　）
   A. 带空心菱形箭头的实线　　B. 带空心三角箭头的实线
   C. 带开叉箭头的虚线　　D. 带实心三角箭头的实线

10. 时序图中表示对象生命周期的符号是(　　)。
A. 矩形框　　B. 虚线箭头
C. 垂直虚线　　D. 菱形框
11. 需求分析任务中的逻辑建模工具不包括以下哪项?(　　)
A. 数据流图　　B. 实体-联系图
C. 状态转换图　　D. 甘特图
12. 性能需求中需要定义的指标不包括(　　)。
A. 响应时效　　B. 吞吐量
C. 用户界面色彩　　D. 容量边界
13. 安全需求中"细粒度访问控制"基于什么实现?(　　)
A. 角色或属性　　B. 数据加密
C. 日志审计　　D. 网络协议
14. 需求规格说明书的核心内容是(　　)。
A. 模块化功能描述　　B. 开发团队名单
C. 项目预算　　D. 市场分析
15. 用例图中表示"功能扩展"的关系是(　　)。
A. 包含关系　　B. 扩展关系
C. 泛化关系　　D. 关联关系
16. 类图中聚合关系的符号特征是(　　)。
A. 带实心菱形箭头的实线　　B. 带空心菱形箭头的实线
C. 带空心三角箭头的虚线　　D. 带开叉箭头的虚线
17. 时序图中表示"同步消息"的符号是(　　)。
A. 带实心三角箭头的实线　　B. 带实心三角箭头的虚线
C. 带开叉箭头的虚线　　D. 垂直虚线
18. 模型分解原则不包括以下哪项?(　　)
A. 模块化原则　　B. 抽象层次原则
C. 功能冗余原则　　D. 关注点分离原则
19. 数据流图的分层设计中,顶层图的主要作用是(　　)。
A. 定义系统与外部交互边界　　B. 描述数据库结构
C. 设计用户界面　　D. 编写代码逻辑
20. 面向对象分析中,动态行为建模的工具是(　　)。
A. 类图　　B. 用例图
C. 时序图　　D. 数据字典

二、多项选择题

1. 需求分析的核心任务包括(　　)。
A. 识别功能需求　　B. 建立逻辑模型
C. 编写代码　　D. 性能优化
2. 结构化分析方法的核心思想是(　　)。
A. 自顶向下分解　　B. 数据流驱动
C. 对象抽象　　D. 动态行为建模

3. 数据流图(DFD)的基本元素包括(　　)。

A. 数据源点和终点　　B. 数据处理

C. 数据存储　　D. 数据流

4. 系统流程图的主要作用是(　　)。

A. 描述物理系统的交互框架　　B. 展示数据逻辑处理过程

C. 定义类的属性和方法　　D. 记录用户操作流程

5. 实体-联系图(E-R 图)的组成要素包括(　　)。

A. 实体　　B. 属性

C. 联系　　D. 消息

6. 数据字典的内容包括(　　)。

A. 数据项　　B. 数据结构

C. 数据流　　D. 处理逻辑

7. 面向对象分析的三大模型是(　　)。

A. 静态结构模型　　B. 动态行为模型

C. 功能约束模型　　D. 数据流模型

8. 用例图的主要组成元素是(　　)。

A. 参与者　　B. 用例

C. 关系　　D. 状态转换

9. 类图中“聚合关系”的特征包括(　　)。

A. 整体与部分弱关联　　B. 部分可独立存在

C. 生命周期由整体控制　　D. 用实心菱形箭头表示

10. 时序图的核心元素包括(　　)。

A. 对象　　B. 生命线

C. 消息　　D. 数据存储

11. 需求分析阶段需明确的非功能需求包括(　　)。

A. 性能指标　　B. 安全需求

C. 用户界面规范　　D. 算法复杂度

12. 逻辑模型的构建工具包括(　　)。

A. 数据流模型　　B. 实体-联系模型

C. 状态转换模型　　D. 用例图

13. 需求文档化过程中的核心交付物包括(　　)。

A. 需求规格说明书　　B. 原型设计文档

C. 变更记录　　D. 测试用例

14. 结构化分析方法的优势包括(　　)。

A. 系统性　　B. 数据驱动

C. 模块独立性　　D. 支持多态性

15. 类图中“依赖关系”的特征是(　　)。

A. 临时性关联　　B. 用虚线箭头表示

C. 生命周期控制　　D. 表示继承关系

16. 用例关系的类型包括(　　)。

A. 包含关系　　B. 扩展关系

C. 泛化关系　　D. 聚合关系

17. 需求分析任务的验证机制包括(　　)。

A. 逻辑一致性验证　　B. 完整性验证

C. 可行性验证　　D. 性能压力测试

18. 数据流图的分层原则要求(　　)。

A. 顶层定义系统边界　　B. 逐级细化处理逻辑

C. 包含物理实现细节　　D. 忽略数据存储

19. 面向对象分析的“封装”特性体现为(　　)。

A. 隐藏内部实现　　B. 提供访问接口

C. 支持继承机制　　D. 定义消息传递

20. 时序图中“控制焦点”的作用是(　　)。

A. 表示对象活动时段　　B. 描述消息触发顺序

C. 定义类属性　　D. 标识数据存储

三、判断题

1. 需求分析的核心任务是明确系统必须做什么。(　　)

2. 可行性研究阶段能够全面覆盖所有用户需求的细节。(　　)

3. 需求分析原则包括信息域建模、功能定义、行为描述和模型分解。(　　)

4. 逻辑建模阶段仅需使用数据流图进行需求描述。(　　)

5. 结构化分析方法的核心思想是自顶向下逐层分解。(　　)

6. 数据流图(DFD)的基本元素包括数据源点、数据处理、数据存储和数据流。(　　)

7. 系统流程图主要用于描述系统的逻辑处理流程。(　　)

8. 实体-联系图(E-R 图)的三大要素是实体、属性和联系。(　　)

9. 数据字典仅需定义数据项和数据结构,无须描述处理逻辑。(　　)

10. 面向对象分析方法的核心是对象抽象与关系建模。(　　)

11. 用例图通过参与者、用例及它们之间的关系展示系统功能需求。(　　)

12. 泛化关系在类图中用带空心菱形箭头的实线表示。(　　)

13. 聚合关系中部分对象的生命周期由整体对象控制。(　　)

14. 时序图的核心元素包括对象、生命线、消息和控制焦点。(　　)

15. 需求文档化的核心交付物仅包含需求规格说明书。(　　)

16. 性能需求需要定义可量化的指标(如响应时间、吞吐量)。(　　)

17. 安全需求仅需关注数据加密,无须考虑访问控制。(　　)

18. 状态转换模型用于描述系统行为的动态特征。(　　)

19. 类图中的依赖关系表示对象间的强生命周期关联。(　　)

20. 需求变更管理需通过版本树记录需求演化路径。(　　)

四、填空题

1. 需求分析是软件定义时期的最后一个阶段,其基本任务是准确回答“_____”。

2. 可行性研究的主要目的是以较低成本评估项目的_____。

3. 信息域建模的三个层次包括信息流捕捉、信息内容解析和_____。
4. 功能建模需在横向维度划分功能模块边界，在纵向维度构建_____。
5. 行为建模需要覆盖正常操作流程和_____场景。
6. 模型分解需遵循模块化、抽象层次和_____原则。
7. 系统流程图通过_____符号描述物理交互框架。
8. 数据流图的四种基本元素是数据源点和终点、数据处理、数据存储和_____。
9. 数据字典中对处理过程的描述通常使用_____或判定树。
10. 实体-联系图的三要素是实体、属性和_____。
11. 用例图中参与者与用例之间的主要关系是_____。
12. 类图中表示“整体-部分”弱关系的是_____。
13. 时序图中对象的生命周期通过_____表示。
14. 需求分析任务的三个阶段是需求识别、逻辑建模和_____。
15. 安全需求需覆盖数据保护、访问控制和_____三个层面。
16. 数据流图的分层设计遵循_____原则。
17. 实体-联系图中，带有下画线的属性表示_____。
18. 类图中依赖关系表示一个类的_____会影响另一个类。
19. 时序图中控制焦点通过_____标识对象的活动状态。
20. 需求变更记录需采用_____可视化呈现需求演化路径。

五、简答题

1. 简述需求分析原则中的四大支柱。
2. 需求识别阶段需构建哪些维度的需求矩阵？
3. 说明逻辑建模阶段的主要工具及其作用。
4. 结构化分析方法的核心思想是什么？
5. 数据字典在结构化分析中的作用是什么？
6. 实体-联系图如何支持数据库设计？
7. 面向对象分析中类图的作用是什么？
8. 用例图中包含关系与扩展关系的区别是什么？
9. 时序图中同步消息与异步消息的区别是什么？
10. 需求规格说明书需包含哪些核心内容？
11. 原型设计文档中低保真与高保真原型的区别是什么？
12. 需求变更管理的关键流程包括哪些步骤？
13. 模型分解原则中的“关注点分离”指什么？
14. 行为建模中如何定义系统状态变迁？
15. 数据流图的父图与子图需满足什么平衡原则？
16. 类图中聚合与组合关系的区别是什么？
17. 如何通过用例图识别系统功能边界？
18. 时序图中控制焦点的作用是什么？
19. 需求分析中如何保障数据字典的全局一致性？
20. 结构化分析与面向对象分析在实践中的融合策略是什么？