需求规范与UML Requirements Specification and UML

By, Fahad Sabah

SRE_Fall2025_Beijing University of Technology, China

SRE_Fall2025_北京工业大学，中国

Defining System with Use Cases & 第三部分：使用用例和场景定义系统行为

3.1 From Needs to Behavior
3.1 从需求到具体行为

The Bridge / 桥梁:

Use Cases and Scenarios bridge the gap between high-level stakeholder goals and detailed functional requirements.
用例和场景在高层级干系人目标与详细的功能需求之间架起了桥梁。

3.2 What is a Use Case?
3.2 什么是用例?

A list of **actions or event steps**, typically defining the **interactions between a role (an actor) and a system**, to achieve a goal.

一系列动作或事件步骤，通常用于定义角色（参与者）与系统之间的交互，以实现一个目标。

Key Components / 关键组成部分:

Actor / 参与者: A role played by a user or external system that interacts with the system.
与系统交互的用户或外部系统所扮演的角色。

Goal / 目标: The successful outcome for the primary actor.
主要参与者期望的成功结果。

3.3 The Use Case Model
3.3 用例模型

A diagram showing actors, use cases, and their relationships (<<include>>, <<extend>>).
一种图表，用于显示参与者、用例及其关系（<<include>>包含, <<extend>>扩展）。

Actors / 参与者: Customer (顾客), Bank (银行)

Use Cases / 用例: Withdraw Cash (取现), Check Balance (查询余额), Deposit Funds (存款)

3.4 Anatomy of a Detailed Use Case
3.4 详细用例的结构

Use Case Name / 用例名称: "Withdraw Cash" ("取现")
Primary Actor / 主要参与者: Customer (顾客)
Preconditions / 前置条件: The ATM is idle; the customer has a valid card. (ATM机处于待机状态；顾客持有有效银行卡。)

Main Success Scenario (Happy Path) / 主成功场景（理想路径）:

1. Customer inserts card. (顾客插入银行卡。)
2. System reads card, prompts for PIN. (系统读取卡片，提示输入PIN码。)
3. Customer enters PIN. (顾客输入PIN码。)
4. System validates PIN. (系统验证PIN码。)
5. .. (Cash dispensed, card returned) ... (... (吐出钞票，退还卡片) ...)

Extensions (Alternative Flows) / 扩展（替代流程）:
4a. Invalid PIN: System prompts to re-enter (with limit). (4a. PIN码无效：系统提示重新输入（有次数限制）。)
4b. Card is stolen: System retains card. (4b. 卡片已挂失：系统吞卡。)

Postconditions (Success) / 后置条件（成功）: Customer account is debited; cash and card dispensed. (客户账户已扣款；现金和卡片已吐出。)

Special Requirements / 特殊需求: "Step 4 must complete verification in under 2 seconds." ("步骤4必须在2秒内完成验证。")

3.5 Scenarios: Instances of a Use Case
3.5 场景：用例的实例

A scenario is a single, specific path through a use case.
场景是贯穿用例的一条具体的路径。

Example / 示例:

Scenario 1 (The "Happy Path"): The "Main Success Scenario" itself.
场景1（理想路径）： "主成功场景"本身就是一条场景。

Scenario 2: "Invalid PIN entered once, then correct PIN entered, then successful withdrawal."
场景2： "第一次输入错误PIN码，然后输入正确PIN码，最终成功取款。"

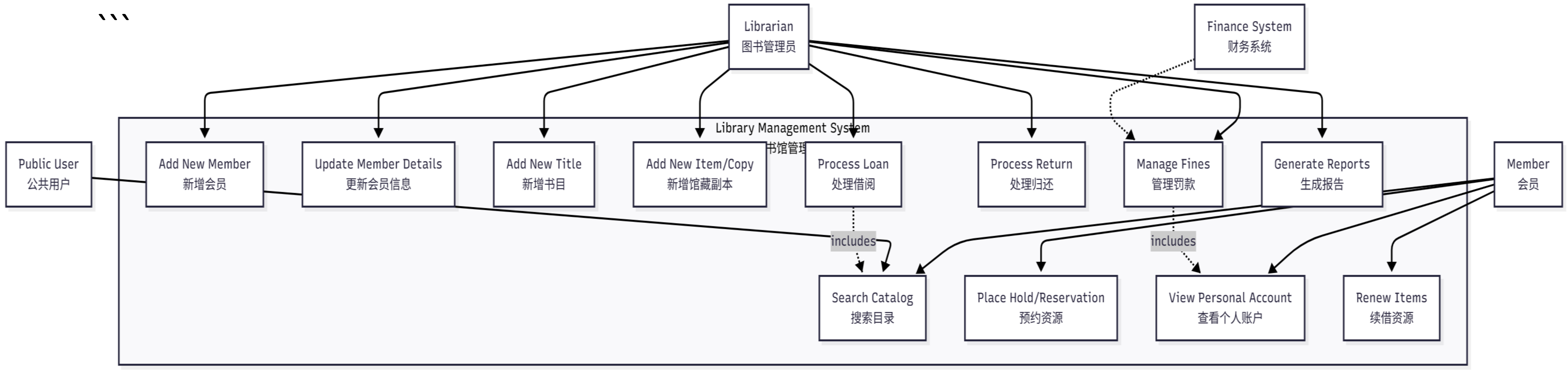
Relationship / 关系:

A Use Case is a collection of related scenarios.
一个用例是一系列相关场景的集合。

Key Use Cases:

- Add New Member (Librarian)
- Add New Title/Item (Librarian)
- Process Loan (Librarian)
- Process Return (Librarian)
- Search Catalog (Member, Public User)
- View My Account (Member)

```


```

Key Use Cases:

- Manage Menu (Manager, Chef)
- Place Dine-In Order (Waiter)
- Process Takeaway/Delivery Order (Receptionist/Customer)
- Manage Table Reservations (Receptionist, Customer)
- Process Payment (Waiter, Cashier)
- Update Table Status (Host, Waiter)
- View Kitchen Order Queue (Chef)
- Generate Sales Report (Manager)
- …..
- Manage Staff Schedule (Manager)
- Apply Discount/Promo (Manager)
- View Customer Feedback (Manager)
- Modify Existing Order (Customer)
- Join Waitlist Remotely (Customer)
- Redeem Loyalty Points (Customer)

Part 4: Formalizing Requirements with Specification Languages
第四部分：使用规格说明语言形式化需求

4.1 The Need for Precision
4.1 精确性的必要

The Problem / 问题:

Natural language is ambiguous.
自然语言是模糊的。

Example: "The system should be fast." vs. "The system shall respond to user input within 200 milliseconds."
示例：“系统应该快” vs. “系统应在200毫秒内响应用户输入”

The Solution / 解决方案:

Specification languages add structure and reduce ambiguity.
规格说明语言通过增加结构来减少模糊性。

4.2 Structured Natural Language
4.2 结构化的自然语言

Using templates and a controlled vocabulary.
使用模板和受控的词汇表。

Template Example / 模板示例:

"The system shall [action] [object] [condition]."
"系统应 [动作] [对象] [条件]。"

Example: "The system shall display the user profile after successful login."
示例: "系统应在成功登录后显示用户配置文件。"

Advantages/优点: Readable by non-technical stakeholders. (对非技术干系人可读)
Disadvantages/缺点: Still prone to ambiguity for complex logic. (对于复杂逻辑仍存在模糊性)

4.3 The Software Requirements Specification (SRS)
4.3 软件需求规格说明书

The official, signed-off contract between developers and customers.
开发人员和客户之间正式的、已签署的合同。

Typical Structure (IEEE Std 830) / 典型结构 (IEEE标准 830):

- ❑ Introduction (引言)
- ❑ Overall Description (总体描述) (Stakeholders, Constraints, Assumptions 干系人、约束、假设)
- ❑ System Features (系统特性) (Use Cases or Functional Requirements 用例或功能需求)
- ❑ External Interface Requirements (外部接口需求)
- ❑ Non-Functional Requirements (非功能需求)
- ❑ Other Requirements (其他需求) (e.g., legal 如法律)

4.4 Model-Based Specification Languages
4.4 基于模型的规格说明语言

Using visual or formal models to represent requirements.
使用可视化或形式化模型来表示需求。

Unified Modeling Language (UML) / 统一建模语言:

Use Case Diagrams / 用例图: For functional scope. (用于描述功能范围)
Activity Diagrams / 活动图: For workflow and business processes. (用于描述工作流和业务流程)
State Machine Diagrams / 状态机图: For systems with state-dependent behavior (e.g., a vending machine). (用于描述状态依赖型系统，如自动售货机)

Formal Methods (Briefly) / 形式化方法 (简介):

Concept / 概念: Using mathematical notation for absolute precision (e.g., Z notation, B-Method).
使用数学符号实现绝对精确 (如 Z notation, B-Method)。

Advantages/优点: Unambiguous, can be proven correct. (无歧义，可被证明正确)
Disadvantages/缺点: Steep learning curve, not accessible to all stakeholders. Used in safety-critical systems (avionics, medical devices). (学习曲线陡峭，非所有干系人都能理解。用于安全关键系统，如航空电子、医疗设备)

Core Domain Classes:

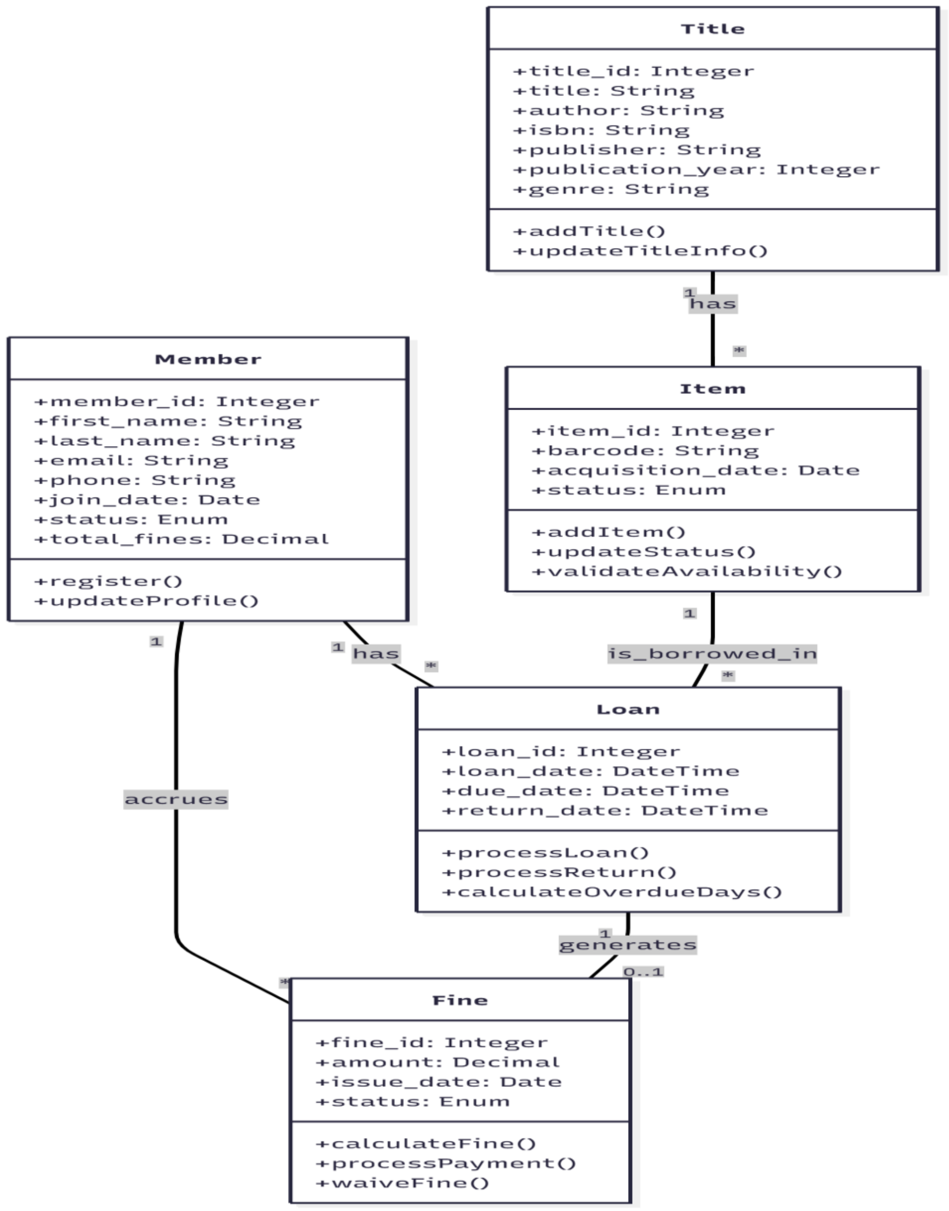
- ❑ **Member (会员)**
- ❑ **Title (书目)**
- ❑ **Item (馆藏)**
- ❑ **Loan (借阅记录)**

Supporting Classes:

- ❑ **Fine (罚款)**

Key Design Patterns Illustrated

- ❑ Association
- ❑ Composition
- ❑ Aggregation
- ❑ Separation of Concerns



Summary, Challenges, and Conclusion
第五部分：总结、挑战与结论

5.1 Key Takeaways
5.1 关键要点

Stakeholder Analysis ensures you are building the right system for the right people.
干系人分析确保你为正确的人构建正确的系统。

Use Cases & Scenarios provide a user-centric, narrative description of what the system must do.
用例和场景提供了关于系统必须做什么的以用户为中心的、叙述性的描述。

Specification Languages (from structured text to models) provide the precision needed to build the system correctly.
规格说明语言（从结构化文本到模型）提供了正确构建系统所需的精确性。

5.2 Common Pitfalls & Limitations
5.2 常见陷阱与局限性

Spending too much time documenting and not enough building.
花费过多时间编写文档，而没有足够时间构建。

Inflexibility to Change / 对变更缺乏灵活性:

The "Big Design Up-Front" can be brittle when requirements evolve.
当需求演进时，“大型前期设计”可能变得脆弱。

The "Myth of Complete Requirements" / "完整需求"的迷思:

It's often impossible to know everything upfront.
通常不可能在前期就了解所有事情。

5.3 Looking Forward & Resources
5.3 展望与资源

Traditional methods are not obsolete; they are a vital tool in the RE toolbox.
传统方法并未过时；它们是需求工程工具箱中的重要工具。

Modern practices often blend traditional rigor with agile adaptability (e.g., "Agile Modeling," creating a "light" SRS).
现代实践通常融合了传统方法的严谨与敏捷的适应性（例如，“敏捷建模”，创建“轻量级”SRS）。

Recommended Reading / 推荐阅读:

❑ Software Requirements by Karl Wiegers and Joy Beatty.
❑ Writing Effective Use Cases by Alistair Cockburn.
❑ IEEE Standard 830-1998 for SRS guidelines.

THANK YOU

2025/10/31