需求工程：蓝图与基石
Requirements Engineering: Outline and Cornerstone

By, Fahad Sabah

TFSE_Fall2025_Beijing University of Technology, China

TFSE_Fall2025_北京工业大学，中国

CONTENTS

需求工程是什么与为何重要

What is and why requirements engineering is important

01

需求工程的定义 (Definition of requirements engineering)

需求工程是定义、文档化和维护需求的系统过程。它确保软件开发有明确的方向和目标，避免项目偏离预期。

Requirements engineering is the systematic **process of defining, documenting, and maintaining** requirements. It ensures that software development has a clear direction and goals, avoiding projects deviating from expectations.

02

类比说明 (Analogy illustration)

就像建造房屋需要蓝图一样，需求工程是软件开发的蓝图。没有它，软件开发就像在黑暗中摸索，容易出错。

Just like building a house requires a blueprint, requirements engineering is the blueprint for software development. Without it, software development is like fumbling in the dark and prone to errors.

03

重要性 (Importance)

需求工程的重要性在于它为后续的设计、测试和验收提供了基础。前期的共识可以减少后期的返工和成本。

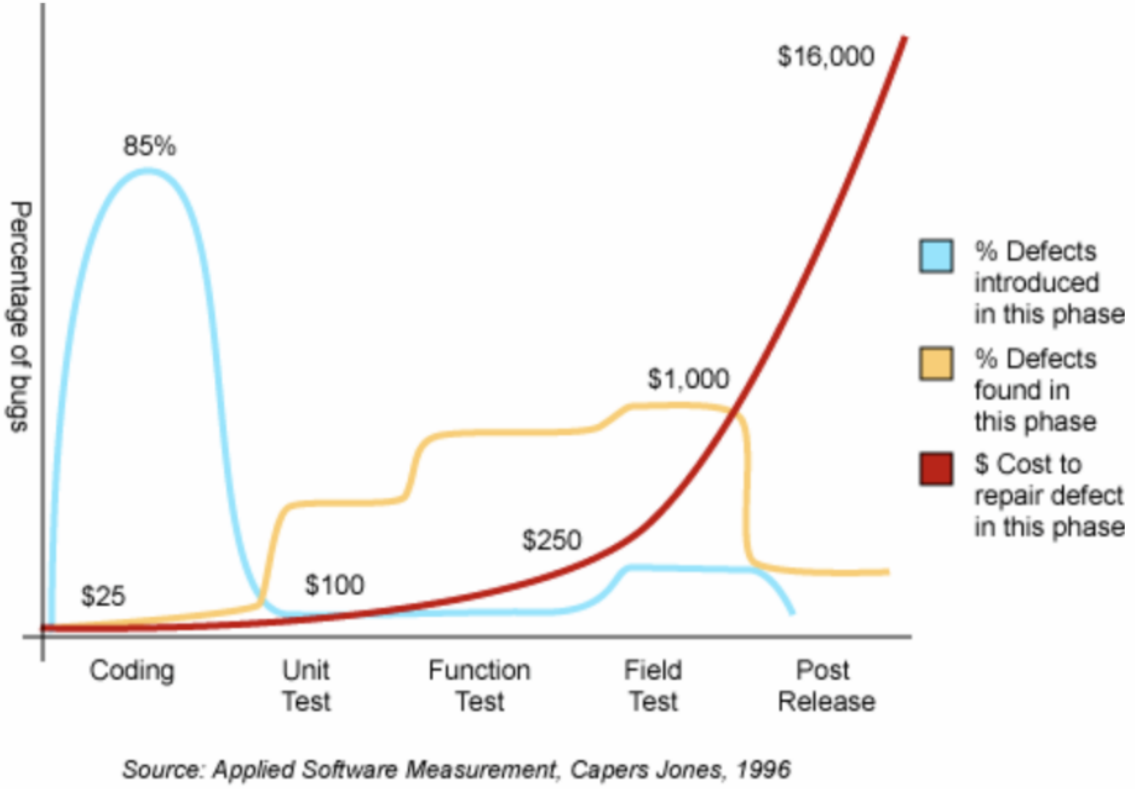
The importance of requirements engineering is that it provides the **basis for subsequent design, testing, and acceptance**. Consensus in the early stages can reduce rework and costs in the later stages.

缺陷发现越晚代价越高
The later the defect is discovered, the more costly it is

成本与时间的关系
The relationship between cost and time

在开发周期中，缺陷发现得越晚，修复成本越高。例如，运维阶段修复需求阶段的缺陷，成本可能放大百倍。

During the development cycle, the later defects are discovered, the higher the cost of fixing. For example, repairing defects in the requirements stage during the O&M phase can increase the cost by a hundredfold.



Source: Applied Software Measurement, Capers Jones, 1996

The Rising Cost of Defects: A Concrete Example
缺陷不断上升的成本：一个具体案例

Scenario: The "User Password Reset" Feature
场景：“用户密码重置”功能

Defect in Requirements Phase (需求阶段的缺陷)

The Flaw / 缺陷描述:
The requirement states: "The system shall email a new temporary password to the user."
需求说明为：“系统应通过电子邮件向用户发送一个新的临时密码。”

What's Missing? / 遗漏了什么?

It fails to specify that the temporary password must be invalid after 15 minutes and forced to change on first use.
它未能明确说明临时密码必须在15分钟后失效，并且用户首次登录时必须强制更改。

Cost to Fix / 修复成本:
A security expert raises this in a review.
一位安全专家在评审会议上指出了这一点。

Action 措施	Cost 成本
Update a few sentences in the requirements document. 修改需求文档中的几句话。	≈ 0.5 person-hours ≈ 0.5 人时

Defect in Coding Phase (编码阶段的缺陷)

The Flaw / 缺陷描述:
The developer implements the feature as specified, creating a permanent temporary password.
开发人员按照规格实现了该功能，创建了一个永久有效的临时密码。

Cost to Fix / 修复成本:
A code reviewer spots the logic error.
代码审查员发现了这个逻辑错误。

Action(s) 措施	Cost 成本
Add a new database field. (添加新的数据库字段。) Modify logic to check password age.(修改逻辑以检查密码时效。) Write code to invalidate expired passwords. 编写代码使过期密码失效。 Update unit tests. (更新单元测试。)	≈ 8 person-hours (a 16x increase) ≈ 8 人时 (成本增加 16 倍)

Defect in Testing Phase (测试阶段的缺陷)

The Flaw / 缺陷描述:

During security testing, a tester finds that temporary passwords never expire, creating a major vulnerability.
在安全测试期间，测试人员发现临时密码永不过期，这造成了一个严重的安全漏洞。

Cost to Fix / 修复成本:

Action(s) 措施	Cost 成本
All work from the coding phase fix. (包含编码阶段的所有修复工作。) Code must be reviewed again. (代码必须再次进行审查。) Intensive regression testing of the entire authentication module. 对整个认证模块进行深入的回归测试。 Test team updates test plan and cases. (测试团队更新测试计划和用例。) Potential release delay.(可能导致发布延迟。)	≈ 40 person-hours (an 80x increase) ≈ 40 人时 (成本增加 80 倍)

Defect in Production (Post-Release) 运营阶段（发布后）的缺陷

The Flaw / 缺陷描述:

A malicious actor exploits the non-expiring passwords to gain unauthorized access. A data breach occurs.

恶意攻击者利用永不过期的密码获取未授权访问。发生数据泄露。

Action(s) 措施

Emergency Fix & Deployment: "War room" assembly, high-priority patch.

紧急修复与部署： 组建“战时会议室”，开发最高优先级的补丁。

System Downtime: Service outage, losing revenue. 系统停机： 服务中断，导致收入损失。

Customer Notification: Legally mandated process. 用户通知： 法律要求的流程。

Cost to Fix (Catastrophic) / 修复成本（灾难性）：

Regulatory Fines: Potentially millions 监管罚款： 可能高达数百万美元

Legal Fees & Settlements: From class-action lawsuits.律师费与和解金： 来自集体诉讼。

Brand Damage & Customer Churn: Long-term financial impact.

品牌损害与客户流失： 长期财务影响。

Total Cost: Hundreds of thousands to millions of dollars (1000x+ increase)

总成本：数十万至数百万美元 (成本增加千倍以上)

缺陷发现越晚代价越高
The later the defect is discovered, the more costly it is

经济意义

Economic significance

前期精准的需求可以显著降低后期的返工成本，提高项目的经济效益和交付效率。

Precise requirements in the early stage can significantly reduce rework costs in the later stage, improving project economic efficiency and delivery efficiency.

2 Traditional vs. Agile Approaches to RE (传统与敏捷的需求工程方法)

传统方法前期锁定需求
Traditional methods lock in demand upfront

传统方法特点 Characteristics of traditional methods

传统方法以瀑布模型和V模型为代表，特点是前期进行详细的规格说明，假设需求可以早期稳定。

The traditional method is represented by the waterfall model and the V model, which is characterized by detailed specifications in the early stage, assuming that the demand can be stabilized early.

优势 advantage

传统方法通过严格的评审和基线控制，降低变更风险，确保项目按计划进行。

Traditional methods reduce the risk of change through rigorous review and baseline controls, ensuring projects stay on schedule.

适用场景 Applicable scenarios

这种方法适用于需求明确且变更少的项目，如大型、复杂、安全关键或合同驱动的系统。

This approach is suitable for projects with clear requirements and low change, such as large, complex, safety-critical, or contract-driven systems.

局限性 limitations

然而，它对需求的早期稳定要求高，一旦需求变更，调整成本较大。

However, it requires high early stability of demand, and once the demand changes, the adjustment cost is large.

敏捷方法迭代演进需求
Agile methodologies iterate and evolve requirements

敏捷方法特点 Agile methodology characteristics

敏捷方法通过短周期交付和持续反馈，允许需求在开发过程中不断演进和修正，适应快速变化的环境。

Agile methodologies allow requirements to **evolve and revise** during development through short cycle times and continuous feedback, adapting to rapidly changing environments.

大型关键系统需传统RE奠基
Large critical systems require traditional RE foundations

传统RE的重要性
The importance of traditional RE

对于规模庞大、安全关键或合同强约束的项目，传统需求工程提供了严谨、系统化的基础，确保项目的可追溯性和合规性。
For large-scale, safety-critical or contractually constrained projects, traditional requirements engineering provides a rigorous and systematic foundation to ensure project traceability and compliance.

核心论点 Core arguments

传统方法在大型、复杂、安全关键或基于合同的系统中不可或缺，它为项目提供了坚实的基石。
Traditional methods are integral in large, complex, safety-critical, or contract-based systems, providing a solid foundation for projects.

3 Two Key Types of Requirements (需求的两种关键类型)

功能需求定义系统行为
Functional requirements define system behavior

01

功能需求定义
Functional requirements definition

功能需求描述系统必须执行的具体行为，例如用户可以自助重置密码。这些需求是系统功能的基础。

Functional requirements describe the specific behaviors that the system must perform. These requirements are the basis for the functionality of the system.

02

可验证性 Verifiability

功能需求必须是可验证的，通常用‘系统应…’的句式书写，以便在测试阶段进行验证。

Functional requirements must be verifiable, usually written in the sentence 'The system should...' for validation during the testing phase.

03

Example 示例

例如，‘系统应允许用户重置密码’，这是一个典型的功能需求，明确了系统的一个具体功能。

For example, 'the system should allow users to reset their passwords', which is a typical functional requirement and clarifies a specific function of the system.

非功能需求约束系统质量
Non-functional requirements constrain system quality

01

非功能需求定义
Non-functional requirements definition

非功能需求规定系统执行功能时的质量属性，如性能、安全、可用性和可靠性。
Non-functional requirements specify the quality attributes of a system when performing its functions, such as performance, safety, availability, and reliability.

02

记录方式
Recording method

非功能需求通常记录在《补充规格说明》中，为系统设计提供全面的约束条件。
Non-functional requirements are typically documented in the Supplemental Specification, providing comprehensive constraints for the system design.

03

忽视后果
Ignore the consequences

忽视非功能需求可能导致系统架构返工，影响用户体验和系统稳定性。
Ignoring non-functional requirements can lead to system architecture rework, affecting user experience and system stability.

功能需求（示例)
Functional Requirements (example)

3. System Features

3.1 Manage Membership

* FR-1.1: The system shall allow a Librarian to register a new Member by capturing their full name, address, phone number, and email address.

* FR-1.2: The system shall automatically assign a unique Member ID and generate a physical library card number upon registration.

* FR-1.3: The system shall allow a Librarian to update a Member's details and deactivate their account.

* FR-1.4: The system shall prevent the registration of a duplicate Member (based on a combination of full name and email).

功能需求（示例)
Functional Requirements (example)

3.3 Process Loans

* FR-3.1: The system shall allow a Librarian to process a loan by scanning a Member ID and one or more Item barcodes.

* FR-3.2: The system shall only permit a loan if:

  * The Member's account is active and in good standing (fines below threshold).

  * The Member has not reached the maximum loan limit.

  * The Item is available (not already on loan, lost, etc.).

* FR-3.3: Upon successful loan, the system shall record the transaction, set the due date (e.g., 21 days from the loan date), and update the Item's status to "On Loan".

Understanding the Stakeholders
第二部分：理解项目干系人

2.1 Stakeholder Analysis: Who Cares?
2.1 干系人分析：谁关心这个系统？

A stakeholder is any **person or group with a vested interest** in the system.
干系人 是对系统拥有既得利益的任何个人或团体。

Goal / 目标:

To identify all relevant stakeholders and understand their influence, interests, and potential impact on the project's success.
识别所有相关的干系人，并理解他们的影响力、利益点以及对项目成功的潜在影响。

2.2 Identifying Stakeholders
2.2 识别干系人

- **Users / 用户:**
  End-users, managers, maintainers.
  最终用户、管理人员、维护人员。
- **Clients / 客户:**
  Those who pay for the system.
  为系统付费的人或组织。
- **Regulators / 监管机构:**
  Government bodies, standards organizations.
  政府机构、标准组织。
- **Developers / 开发人员:**
  The engineering team.
  工程团队（开发、测试等）。
- **Others / 其他:**
  Legal, marketing, support.
  法律、市场、支持团队。

2.3 Classifying & Prioritizing Stakeholders
2.3 分类与确定干系人优先级

The Power/Interest Grid / 权力/利益网格

	High Power 高权力	Low Power 低权力
High Interest 高利益	Manage Closely 重点管理	Keep Informed 及时告知
	(Key decision-makers 关键决策者)	(End-users 最终用户)
Low Interest 低利益	Keep Satisfied 随时满意	Monitor 监督
	(Senior management 高层管理)	(Minimal effort 最小化投入)

Why This Matters / 为何重要:

Ensures communication efforts are focused correctly and no critical voice is missed.
确保沟通工作重点突出，且不会遗漏关键意见。

2.4 Documenting Stakeholder Analysis
2.4 记录干系人分析

Stakeholder Register/Map / 干系人登记册/地图

A simple table to consolidate analysis.
一个用于汇总分析的简单表格。

Example Table / 示例表格:

Name 姓名	Role 角色	Influence 影响力 (High/Med/Low)	Interests / Key Concerns 利益/关键关注点	Communication Needs 沟通需求
Jane Doe	Project Sponsor	High	On-time delivery, budget	Weekly project status report
	项目发起人	高	按时交付、预算	每周项目状态报告
John Smith	End-User	Low	Ease of use, efficiency	Monthly newsletter, training sessions
	最终用户	低	易用性、效率	月度通讯、培训课程
Dr. Li	Regulatory Auditor	High	Legal compliance, data security	Formal review meetings, audit trails
	监管审计员	高	合法合规、数据安全	

Defining System Behavior with Use Cases & Scenarios
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。"

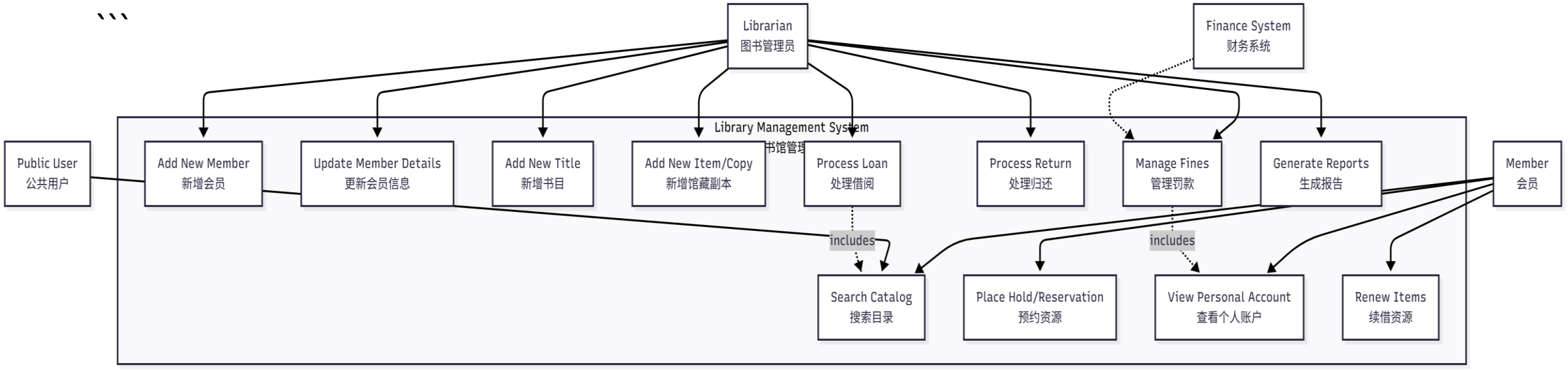
Relationship / 关系:

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:

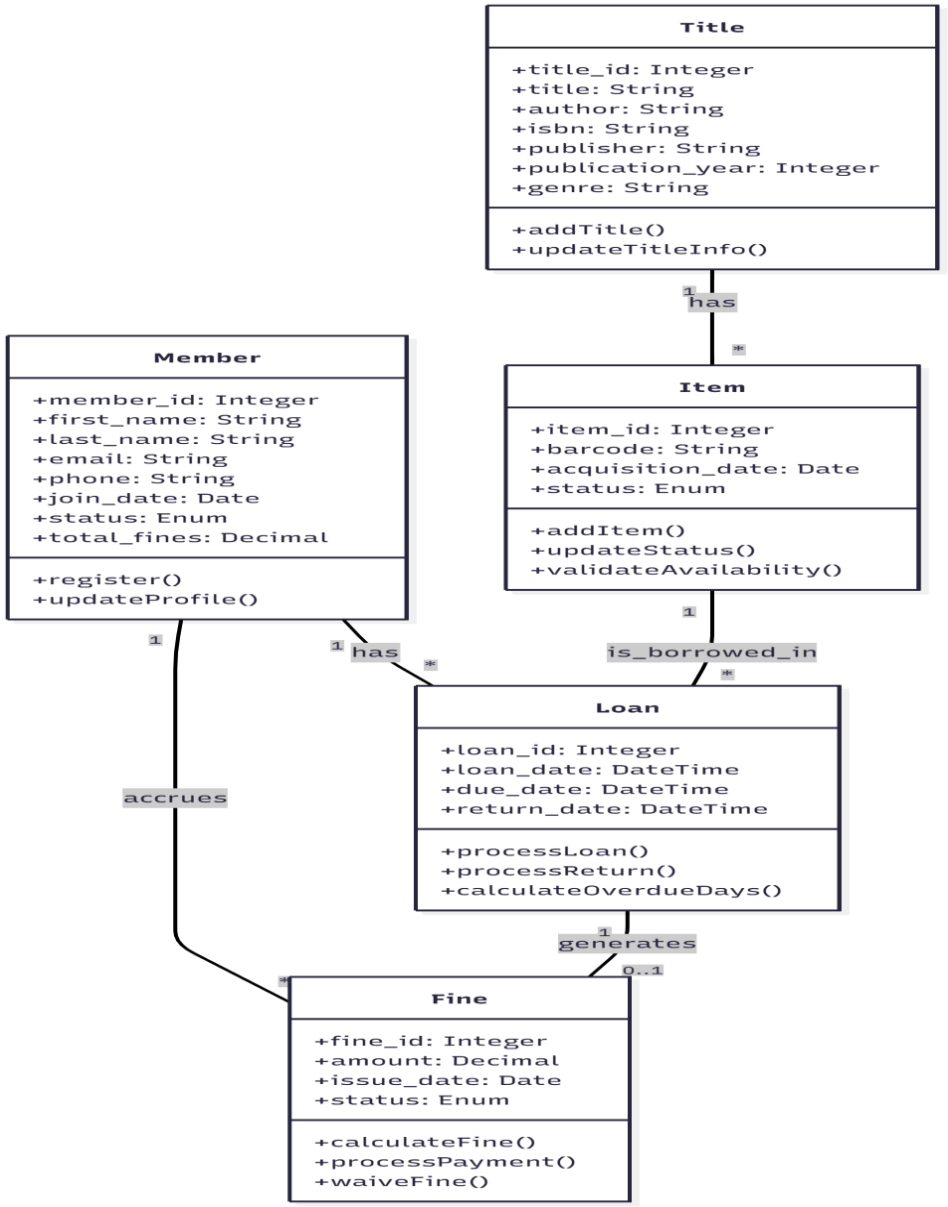
- ❑ **Member (会员)**
- ❑ **Title (书目)**
- ❑ **Item (馆藏)**
- ❑ **Loan (借阅记录)**

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