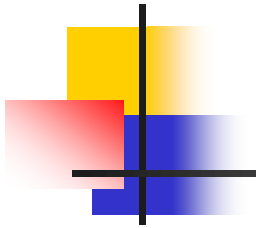
CHAPTER 4
Capturing the Requirements

- Eliciting requirements from customers
- Modeling requirements
- Reviewing requirements to ensure their quality
- Documenting requirements for use by the design and test teams



Why are requirements important?

--See Sidebar 4.1 on Page 142.

人们并不清楚应该做什么，却一直忙碌不停地开发。

需求分析

- 软件需求分析是软件生存期的一个重要阶段，是软件开发项目得以成功的基础。其最根本的任务是确定为了满足用户的需要软件系统必须做什么。
- 软件需求分析是一个不断发现和决定的过程，在此过程中，软件开发者和软件申请者（用户）同样起着重要的作用。
- 在需求分析与说明过程中，需要大量交换意见，其间充满着传错信息和发生误解的可能性：

  “我知道你相信你明白了你认为我所说的是什么，但是我不能肯定你是否意识到你听到的并不是我所指的意思 ……”。

4.1 The Requirements Process

4.1.1. Requirement 需求的定义

- 需求来源于用户的一些“需要”，这些“需要”被分析、确认后形成完整的文档，该文档详细地说明了产品“必须或应当”做什么。
- 所以如果只有一些零碎的对话、资料或邮件，你就以为自己已经掌握了需求，那是自欺欺人。

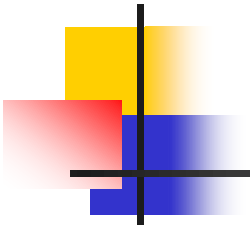
A proposed software system has a purpose, usually expressed in terms of goals or desired behavior.

A **Requirement**:

Is an expression of desired behavior.

Requirements:

Are specific descriptions of functions or characteristics that address the general purpose of the system.



Note: 关注客户及其问题，而不是如何实现。

Requirements focus on the customer and the problem, not on the solution or the implementation.

4.1.2. The requirements process

Step 1: Elicition 需求引导

Step 2: Analysis 分析和建模

Step 3: Specification 形成文档

Step 4: Validation 验证和确认

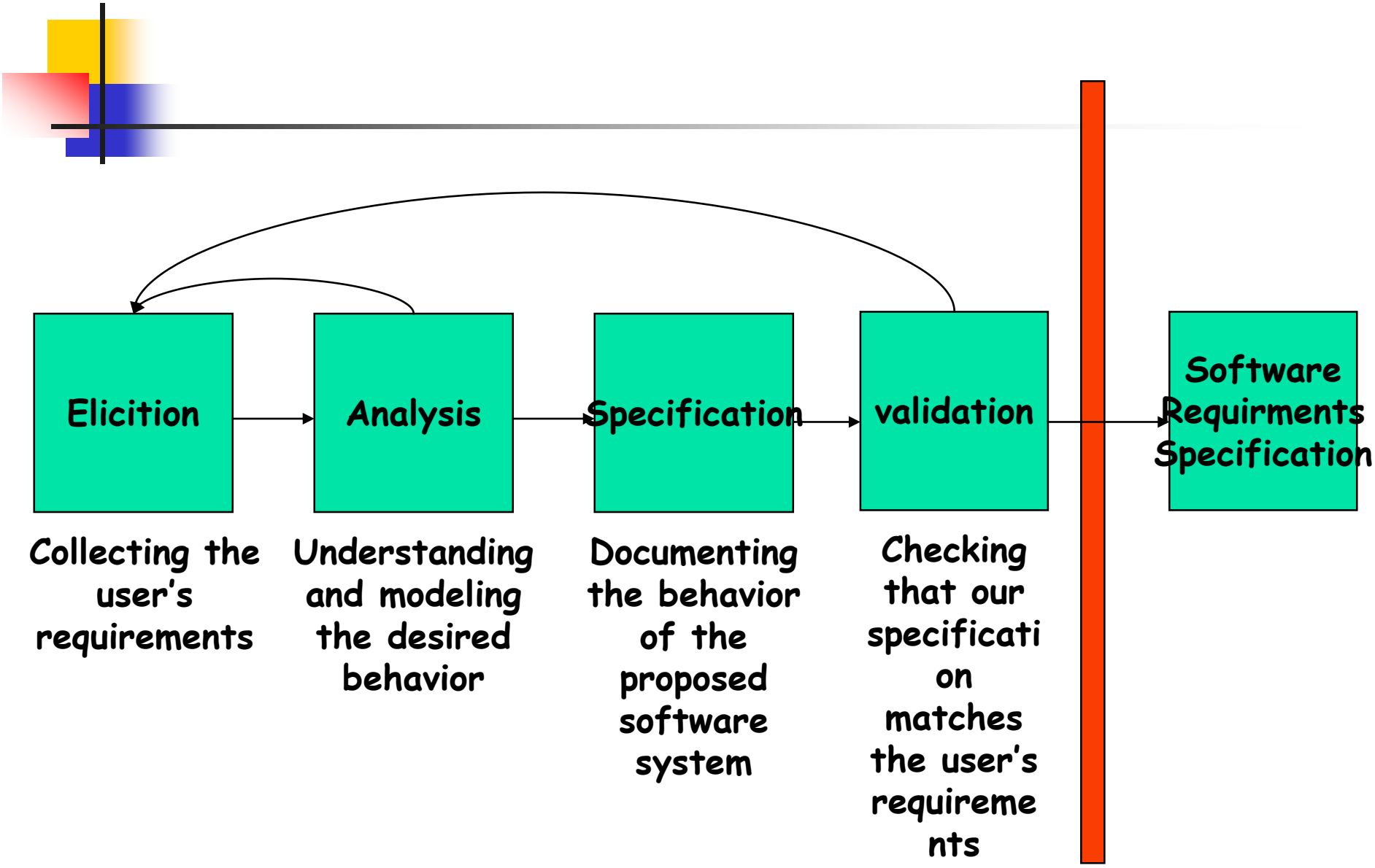
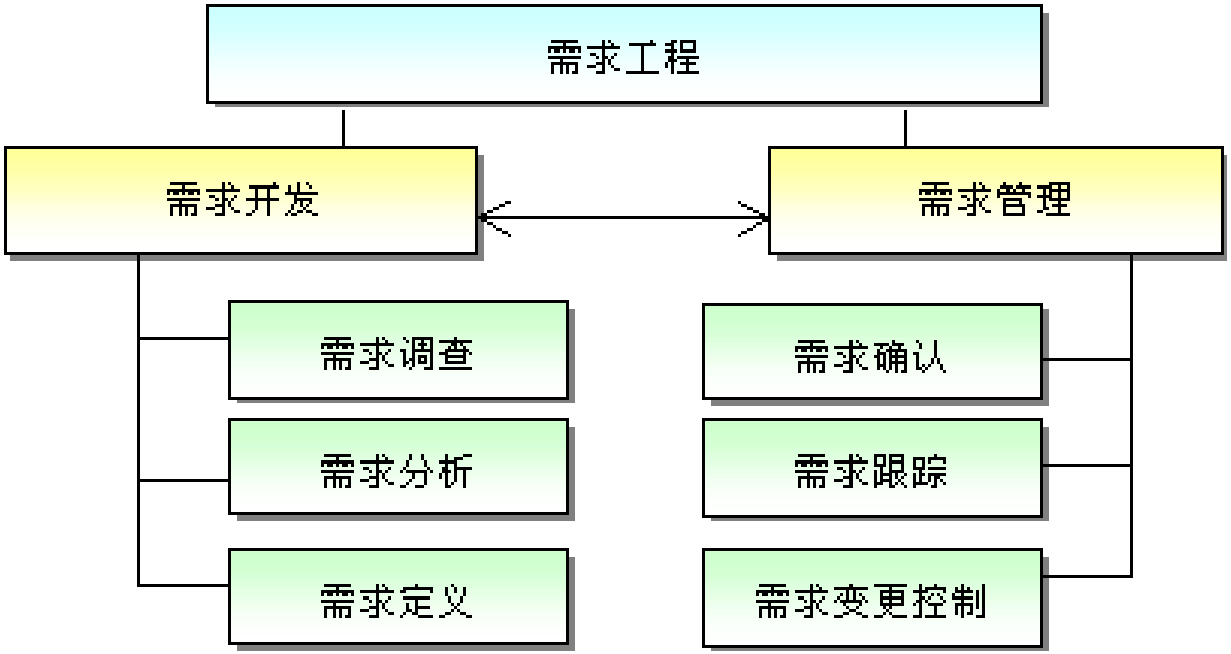
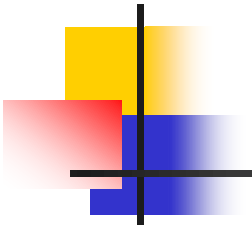


Fig 4.1 The process of the determining requirement

需求工程

- 把所有与需求直接相关的活动通称为需求工程。
- 需求工程中的活动可分为两大类，一类属于需求开发，另一类属于需求管理。
- 需求工程的结构图





4.2 Requirements Elicitation

1. Requirements elicitation is critical/difficult.

- Requirements are ill-formed and ill-understood by everyone.
- Customers know their business ,but cannot always describe their problems to outsiders.
- Developers know about computer solutions, nut not always about how possible solutions will affect the customers' business activities.
- People have different meanings for the same words.

Impedance mismatches

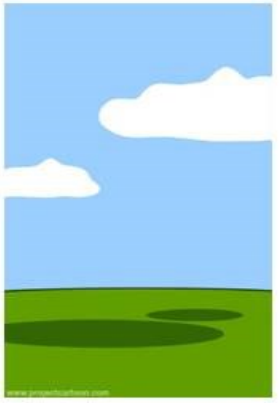
How the customer explained it

How the Project Leader understood it

How the Business Consultant described it

How the Analyst designed it

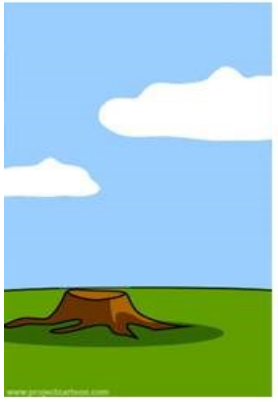
How the Programmer wrote it



How the project was documented

What Operations installed

How it performed under load



How it was supported

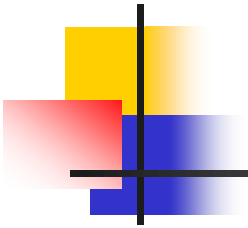
What marketing advertised

How the customer was billed

What the customer really needed

2. Stakeholders: 利益相关者

- **Clients who are the ones paying for the software development;** 客户
- **Customers who buy the software;** 顾客
- **Users who will use the software;** 用户
- **Domain experts who are familiar with the software to be developed;** 领域专家
- **Market researchers;** 市场调查人员
- **Lawyers or auditors;** 律师/审计员
- **Software engineers or other technology experts.** 技术专家



冲突：

- ◆Each stakeholders has a particular view
- ◆Different participants may expect different level of detail in the requirements documentation;
- ◆Users and developers may have preconceptions about what the other group values and acts;

One of the many skills of a requirements analyst

- ◆Is the ability to understand each view
- ◆to capture the requirements in a way that reflects the concerns of each participants

Good requirements analysis require excellent interpersonal skills and as well as solid technical skills.

3. means of eliciting requirements 途径

- Interviewing users or stakeholders in groups;
- Reviewing available documentation;
- Observing the current system to get information and better understand;
- Apprenticing with users to learn about the task;
- Using domain specific strategies;
- Brainstorming with current and potential users about how to improve the proposed product.

用户访谈
Interviews

问卷调查
Questionnaires

专题讨论会
Workshops

头脑风暴
Brain Storming

原型系统
Prototypes

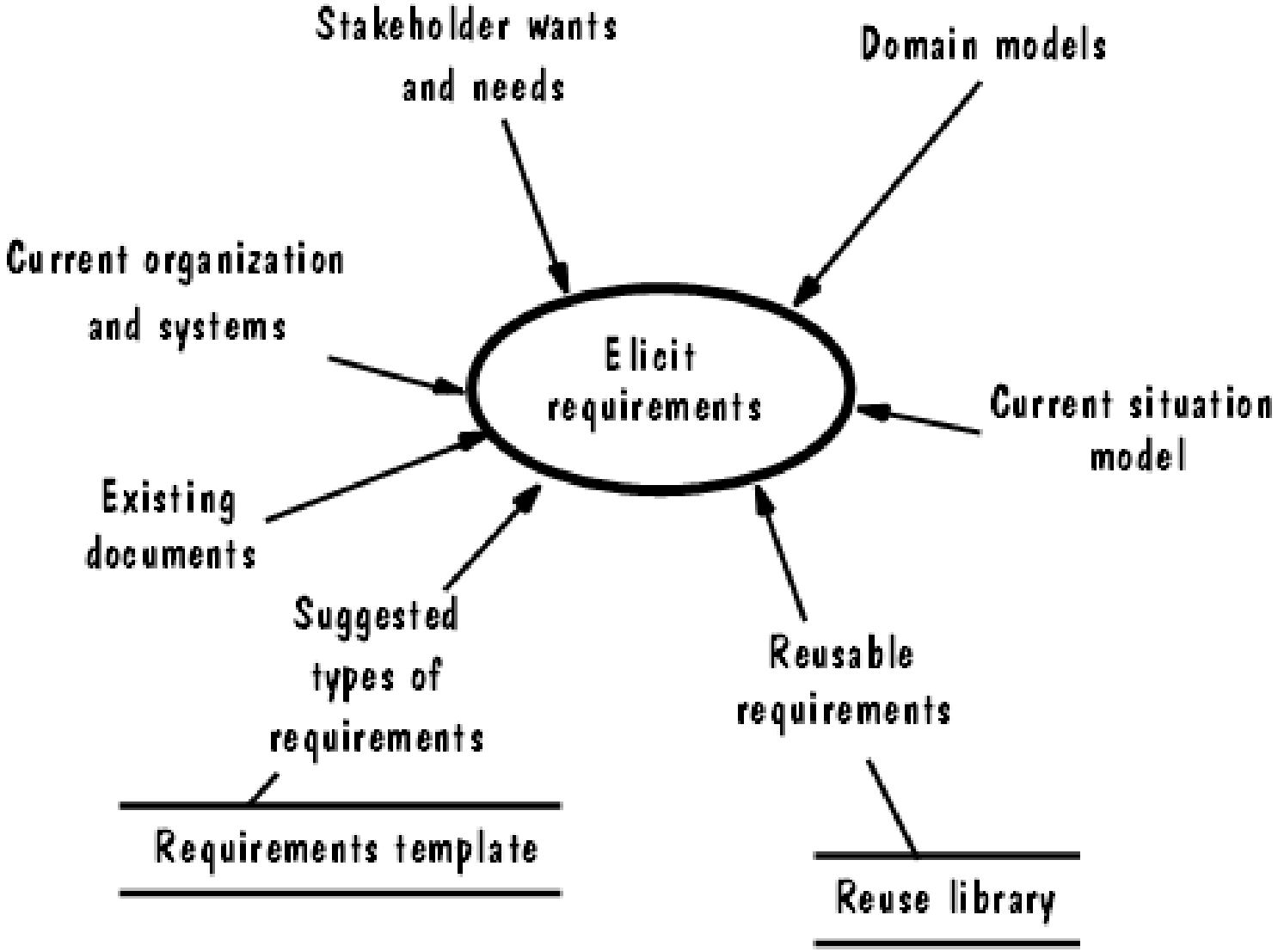


Fig 4.2 Sources of possible requirements (Robertson and Robertson 1999)

4. The goal of the Requirements process 目的

- Requirements are elicited at the beginning of development, and the goal is to determine the nature of the customer's problem
  - Problem and understanding is mostly easily stated in terms of the customer's business
- Requirements should be focused on the customer and the problem, not on the solution or implementation
  - A discussion of any solution is premature until the problem is clearly defined

4.3 Types of Requirements

1. Functional requirement and Nonfunctional requirement

   See Table 4.2.

(1) Functional requirement 功能需求 - 系统应该提供的服务、如何对输入做出反应以及系统在特定条件下的行为描述

(2) Quality/Nonfunctional Requirement 质量/非功能性需求 - 软件系统必须拥有的质量特性，诸如响应时间、易使用性、高可靠性或低维护代价等

(3) Design Constraint 设计约束 - 已经做出的设计决策或限制问题解决方案集的设计决策，例如平台或构件接口的选择

(4) Process Constraint 过程约束 - 对用于构件系统的技术和资源的限制，例如客户要求使用敏捷开发方法。

2. Making Requirements Testable

---See Sidebar 4.4

- 量化
- 避免代词
- 每个名次仅在一处定义

3. Resolving Conflicts

- **Separate the requirements into 3 categories**
  - **Requirements that absolutely must be met (Essential);**
  - **Requirements that are highly desirable but not necessary (Desirable);**
  - **Requirements that are possible but could be eliminated (Optional).**
  - **The purposes doing so:**
    - **If the system as defined will cost too much, or take too long to develop**
    - **Category 3 requirements can be dropped**
    - **Category 2 requirements can be analyzed for elimination or postponement**
    - **Prioritization can be helpful in resolving conflicts among quality requirements.**

4. Requirements can serve several purposes

- 客户和分析人员：
- 设计人员：
- 测试人员：
- 维护人员：
- **To ensure both we and the customers understand and use the requirements properly, it is important that the requirements be of high quality.**
  - **To this end, we check the requirements to make sure they have the desired characteristics.**

5. 2 Kinds of Requirements Documents

- **Requirements definition**
  - **written in terms that the customer can understand**
  - **It is complete listing of everything the customer expects the system to do**
  - **It represent the common understanding of the developer and the customer**
  - **It is usually written jointly by the developer and the customer**

用户需求说明书的参考模板

0. 文档介绍

1. 产品介绍

提示：（1）说明产品是什么，什么用途。（2）介绍产品的开发背景。

2. 产品面向的用户群体

提示：（1）描述本产品面向的用户（客户、最终用户）的特征，（2）说明本产品将给他们带来什么好处？他们选择本产品的可能性有多大？

3. 产品应当遵循的标准或规范

提示：阐述本产品应当遵循什么标准、规范或业务规则（Business Rules），违反标准、规范或业务规则的产品通常不太可能被接受。

4. 产品的功能性需求

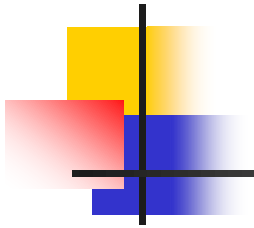
功能类别	功能名称、标识符	描述
Feature A	Function A.1	
	…	
Feature B	Function B.1	
	…	
Feature C	Function C.1	
	…	

5. 产品的非功能性需求

需求类别	需求名称、标识符	描述
用户界面需求		
软硬件需求		
质量需求		

6. 其它需求

附录：用户需求调查报告



- **Requirements specification**
  - **restates the requirements definition in technical terms**
  - **It is complete listing of everything the customer expects the system to do**
  - **It is appropriate for the development of s system design**
  - **It is the technical counterpart to the requirements definition**
  - **It is written by the requirements analysts**

软件需求说明书的参考模板

0. 文档介绍

1. 产品介绍

2. 产品面向的用户群体

3. 产品应当遵循的标准或规范

4. 产品范围

5. 产品中的角色

提示：阐述本产品的各种角色及其职责。各种角色的具体行为将在功能性需求中描述。

角色名称	职责描述

6. 产品的功能性需求

6.0 需求分类

功能类别	功能名称、标识符	描述
Feature A	Function A.1	
	…	
Feature B	Function B.1	
	…	

6.m Feature M

6.m.n Function M.N

名称、标识符	
优先级	
功能描述	
输入、输出 操作序列等	
其它说明	

7. 产品的非功能性需求

需求类别	需求名称、标识符	描述
用户界面需求		
软硬件需求		
质量需求		

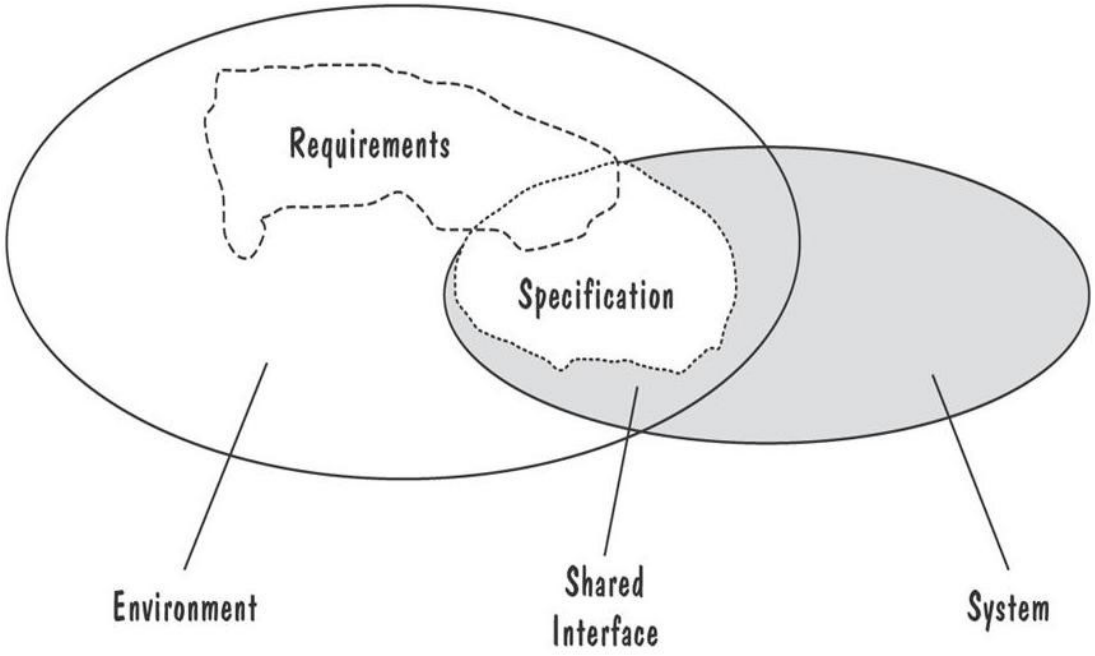
8. 其它需求

附录 A：需求建模

附录 B：需求评审报告摘要

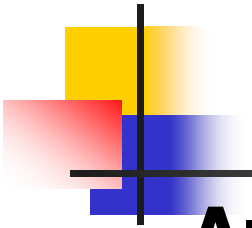
附录 C：需求承诺

- Requirements defined anywhwere within the environment's domain, including the system's interface
- Specification restricted only to the intersection between environment and system domain

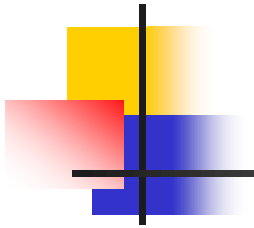


4.4 Characteristics of Requirements

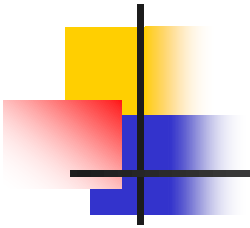
- **Are the requirements correct ?** 正确性
  - **Both we and the customer should review them to assure that they stated without error.**
- **Are the requirements Consistent ?** 一致性
  - **To make sure there no conflicting or ambiguous requirements.**
  - **2 requirements are inconsistent if it is impossible to satisfy them simultaneously.**
- **Are the requirements unambiguous ?** 确定性
  - **If multiple readers of the requirements can walk away with different but valid interpretations.**



- **Are the requirements Complete ? 完整性**
  - **To make sure that every possible states, state changes, inputs, products, constraints and other needed things are described by some requirements.**
- **Are the requirements feasible ? 可行性**
  - **To make sure there is no requirement which can not be done realistically.**
- **Is each requirement relevant ? 相关性**
  - **A requirement may restricts the developers unnecessarily or includes functions that are not directly related to the problem at hand.**
  - **We should review the requirements to retain only those that work directly to solve the problem.**



- **Are the requirements testable ?** 可测试性
  - **We must be able to write tests that demonstrate that the requirements have been met.**
- **Are the requirements traceable ?** 可跟踪性
  - **Can each function be traced back to a set of requirements that mandate it ?**



Examples

- “The system shall provide real-time response to queries.”
  - Question : What “real-time response” is ?
  - It is better written as :
- “The system shall response to queries in not more than 2 seconds.”
- “Accuracy shall be sufficient to support mission planning.”
  - How can we test to see if it satisfies this requirement ?
- “In identifying the position of the satellite, position error shall be less than 50 feet along orbit, less than 30 feet off orbit.”

4.5 Modeling Notations

- It is important to have standard notations for modeling, documenting, and communicating decisions.
  - Define requirements in a more rigorous and controlled fashion.
- Modeling can help us to understand the requirements thoroughly
  - Holes in the models reveal unknown or ambiguous behavior
  - Multiple, conflicting outputs to the same input reveal inconsistencies in the requirements
  - Tools can be developed to check for completeness and consistency, and the trace-ability

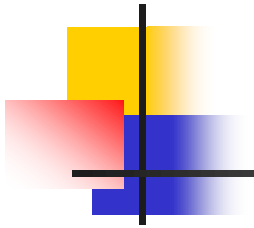
1. Entity-Relationship Diagrams

（1）作用

通过标识对象/实体的名称、属性、联系，建立问题的概念模型

（2）建模符号

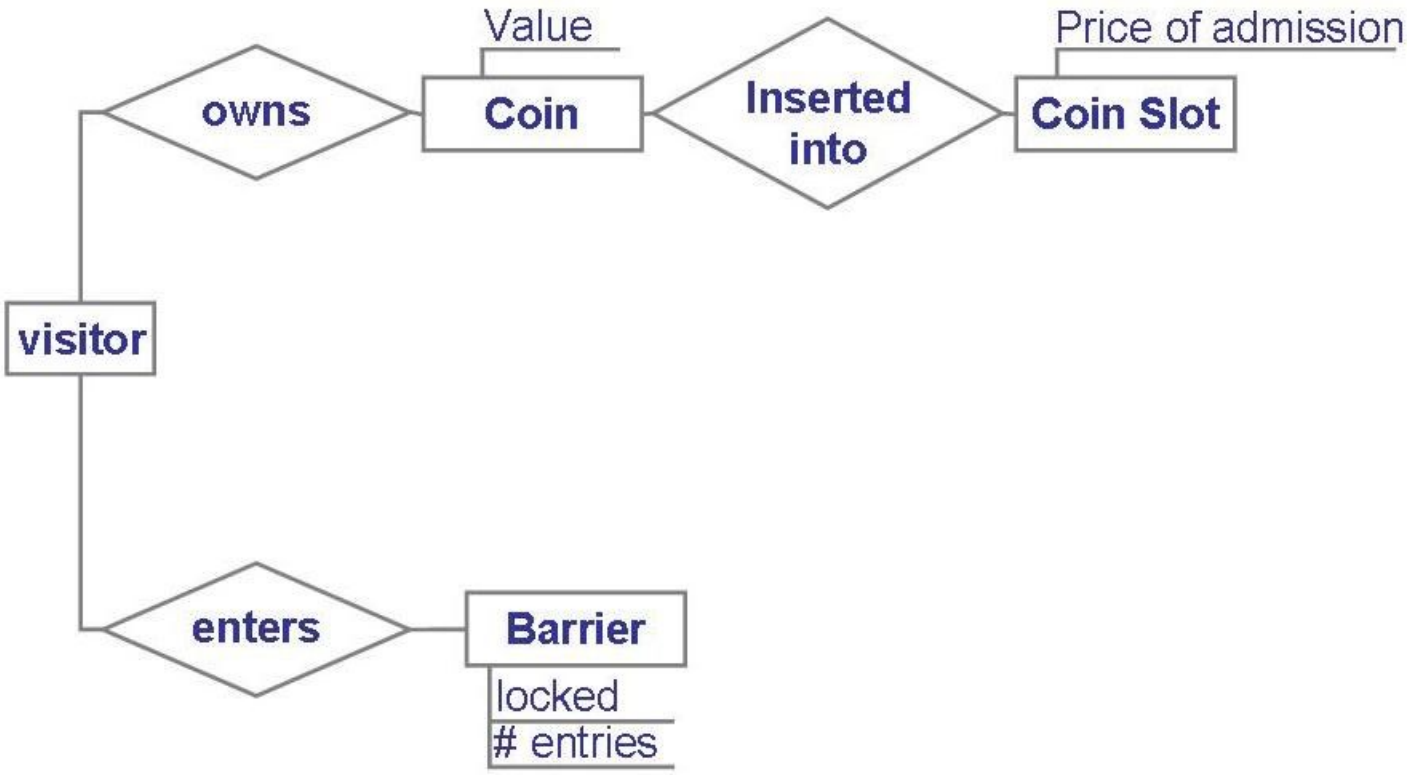
- 实体/**Entity – a rectangle, represent a collection of objects**
- 联系/**Relationship – edge between 2 entities, with a diamond in the middle of the edge specifying the type of relationship.**



- 属性/**Attribute – is an annotation on an entity that describes data or properties associated with the entity.**
- 可变实体/**Mutable entity – whose membership or relations to members of other entities may change over time.**（表达联系的实体）

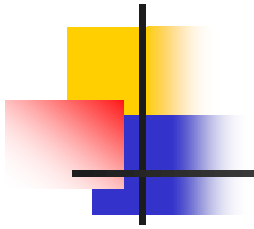
- 主要用于数据库设计的概念模型
- **Fig. 4.4, p.158.**

- Entity diagram of turnstile problem



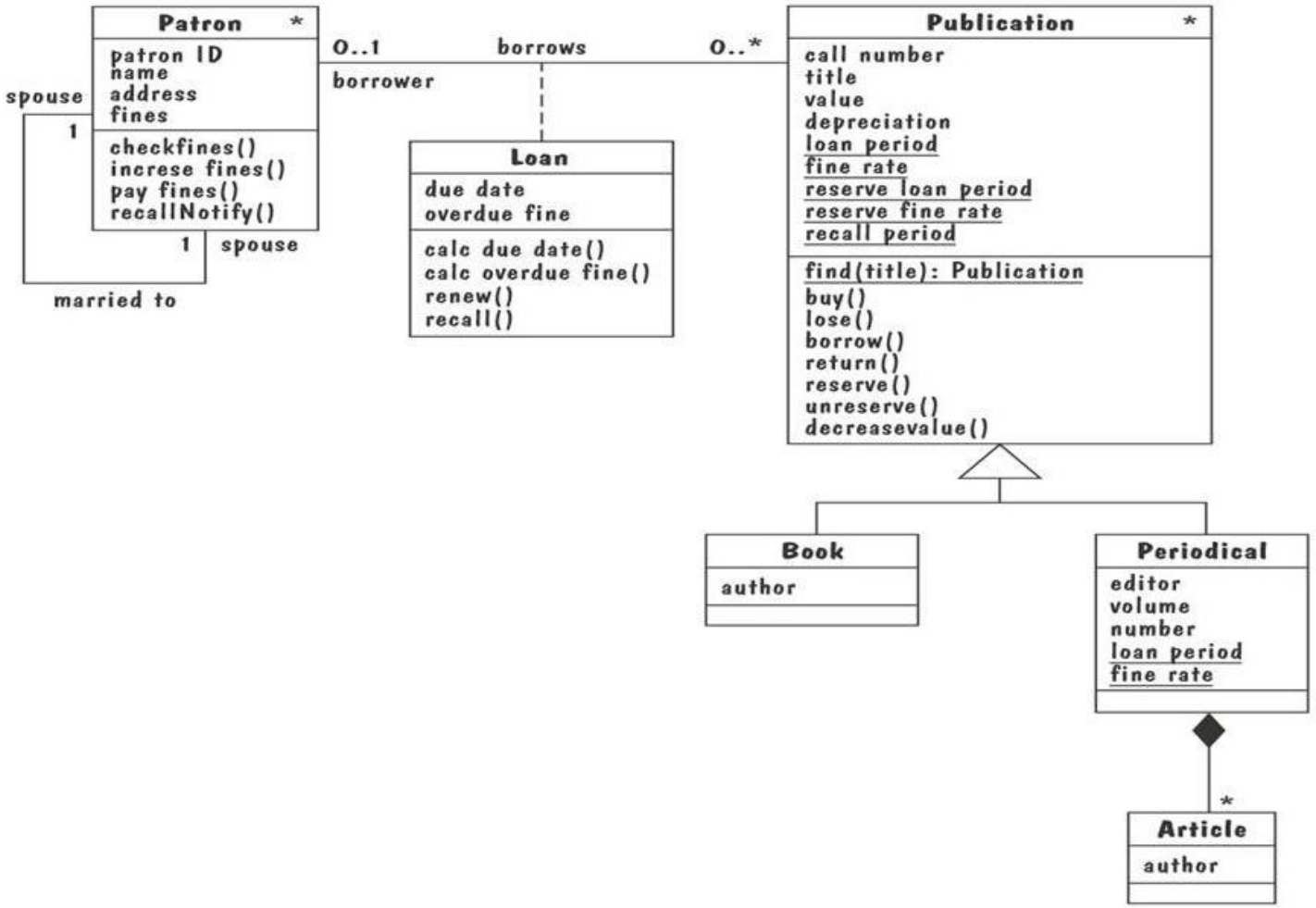
Entity-Relationship Diagrams: UML Class Diagram

- **Class diagram – a sophisticated ER diagram**
- **Class**
  - **Real-world entities 现实世界的实体/对象**
  - **attributes and operations 属性和行为**
  - **Scope of class members – public or private / class scope 成员的可见范围**
- **Association 联系的类型**
  - **indicate relationship between class instances.**
  - **Aggregation association, or HAS-A relationships, represented as white diamond on one end.**
  - **Composition association, instance of the compound class are physically constructed from instances of the compound classes, represented as black diamond on one end.**



- **Generalization vs Specialization – represented with white triangles, called sub-type or IS-A relation.**
- **Association can have labels that describe the relationships**
- **Role names**
- **Multiplicities**
- **Association class**

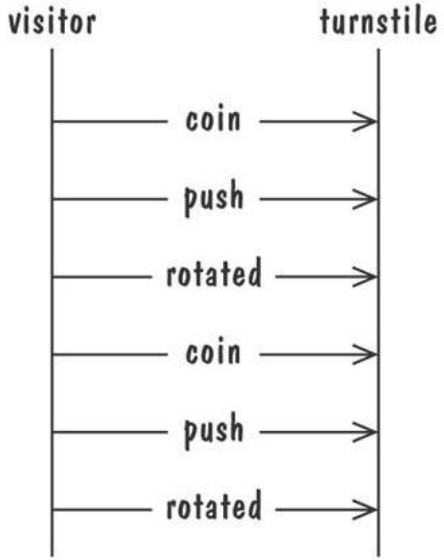
- **Fig. 4.5, p.160.**

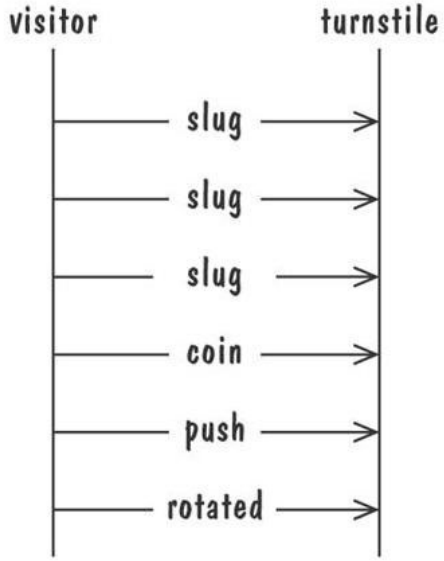

Patron
*
patron ID
name
address
fines
checkfines()
increse fines()
pay fines()
recallNotify()
spouse
1
1
spouse
married to
0..1
borrows
0..*
borrower
Loan
due date
overdue fine
calc due date()
calc overdue fine()
renew()
recall()
Publication
*
call number
title
value
depreciation
loan period
fine rate
reserve loan period
reserve fine rate
recall period
find(title): Publication
buy()
lose()
borrow()
return()
reserve()
unreserve()
decreasevalue()
Book
author
Periodical
editor
volume
number
loan period
fine rate
*
Article
author

2. Event Traces

- **Is a graphical description of sequence of events that are exchanged between real-world entities.**
- 建模元素：
  - 实体
  - 时间轴
  - 实体之间的事件/交互（通过消息）
- 每个图仅表示一种可能的事件跟踪行为
- **Fig. 4.6, p.162.**

- Graphical representation of two traces for the turnstile problem
  - trace on the left represents typical behavior
  - trace on the right shows exceptional behavior

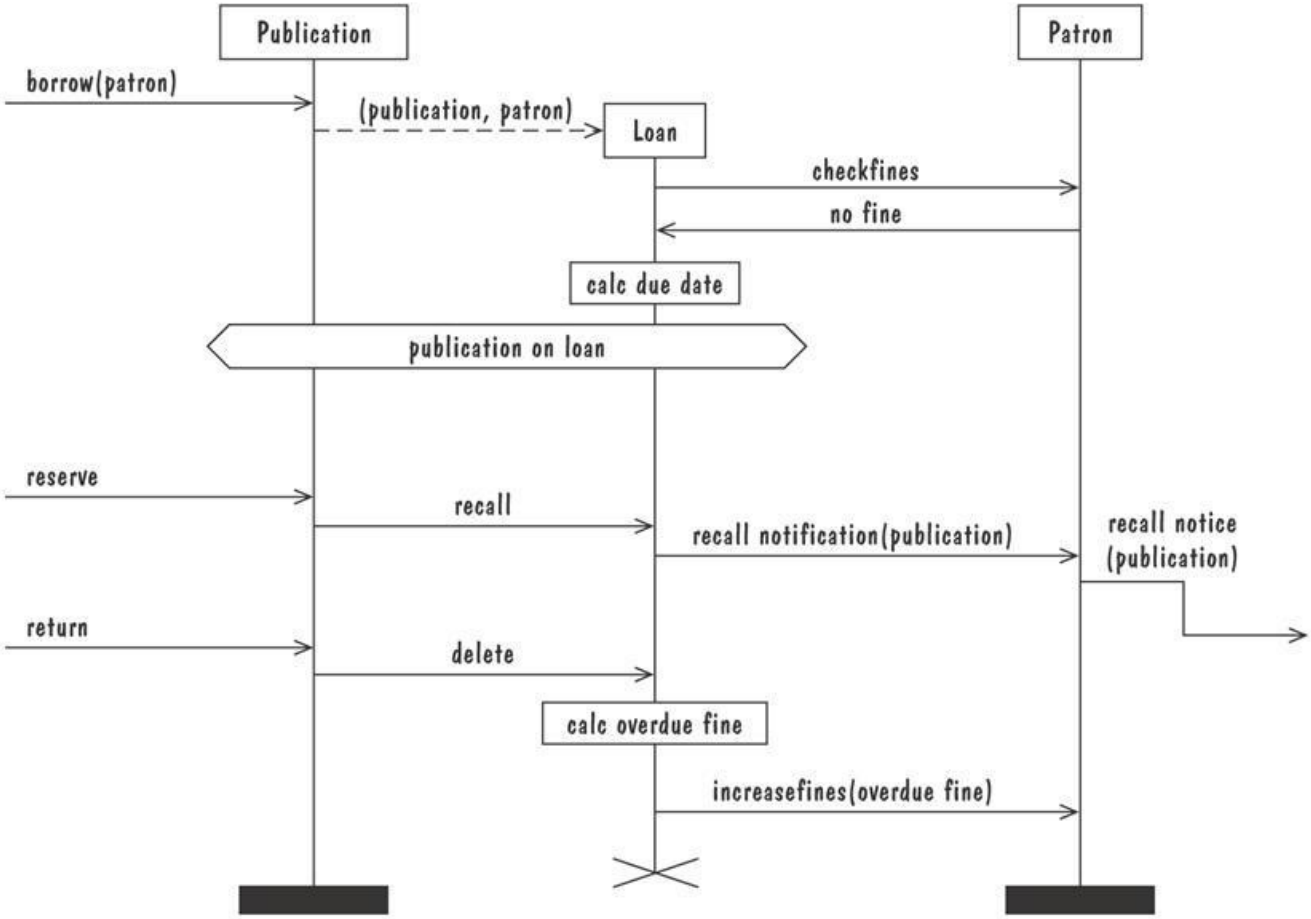




Event Traces: UML Sequence Chart

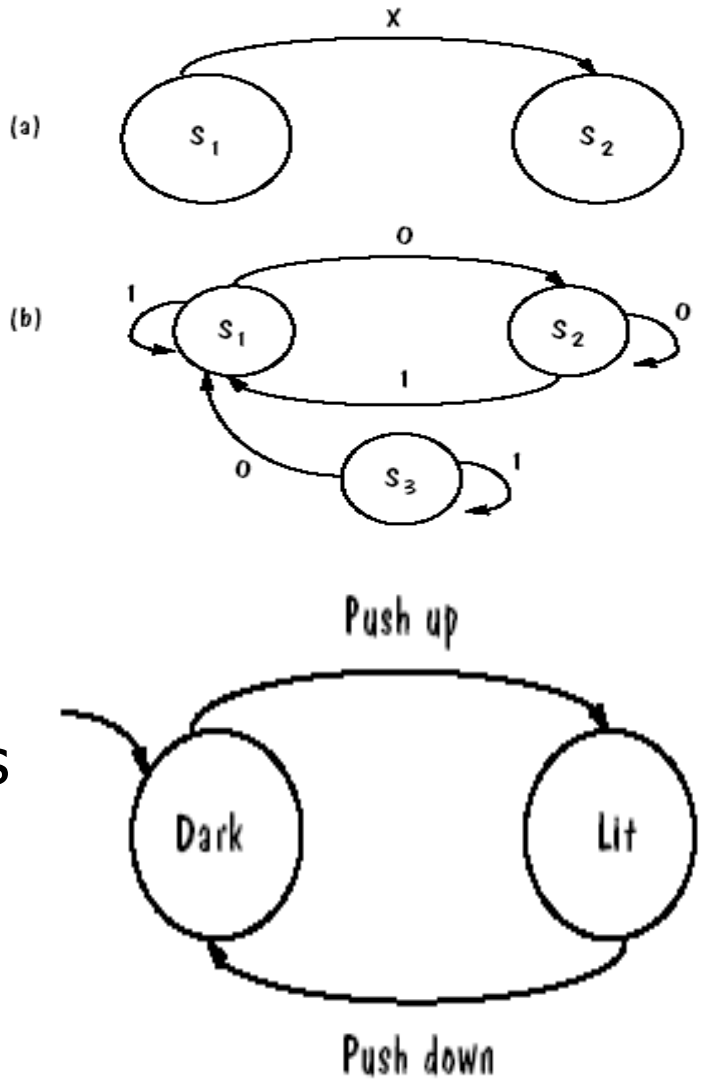
- Is an enhanced event-trace notation,With facilities for
  - creating and destroying entities
  - specifying actions and times
  - composing events.
- Entity
- Message
- Action
- condition
- Fig. 4.7, p.163.

- Message sequence chart for library loan transaction

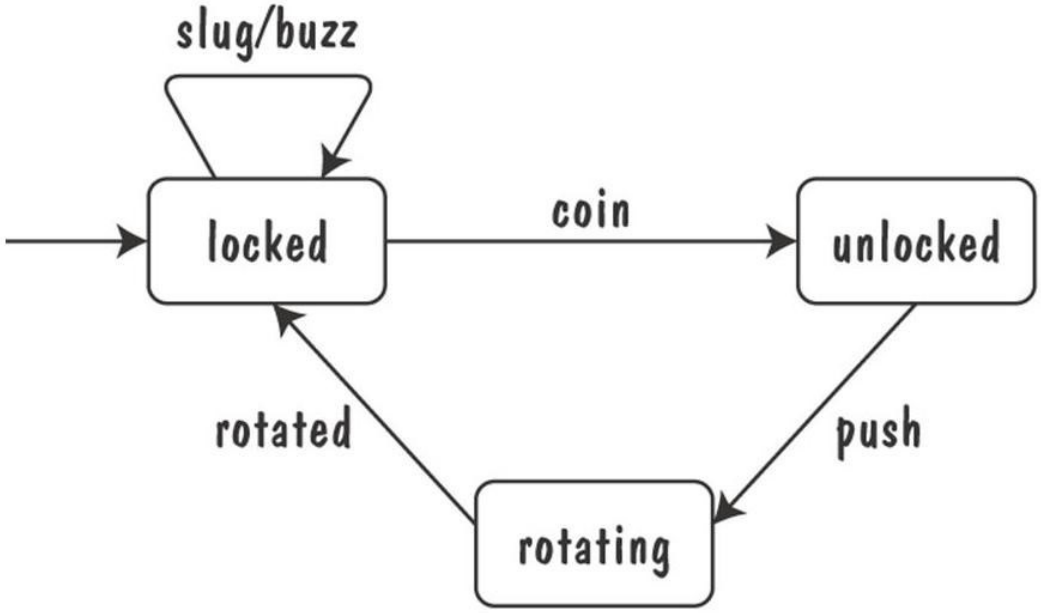


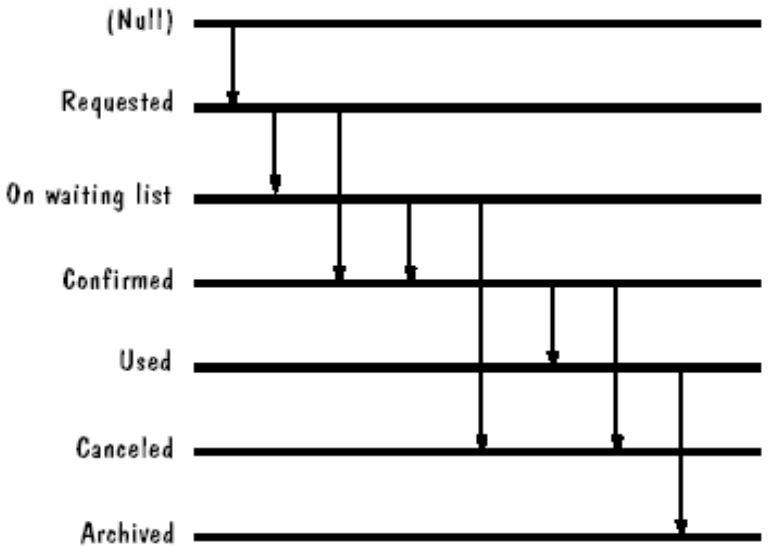
3. State Machine

- 在一个图中表示事件跟踪的集合/描述动态行为
- 建模元素
  - State
  - Transition
    - triggering-event/output-event
- Deterministic state machine
  - for every state and event there is unique response.
- Fig. 4.8, p.164.



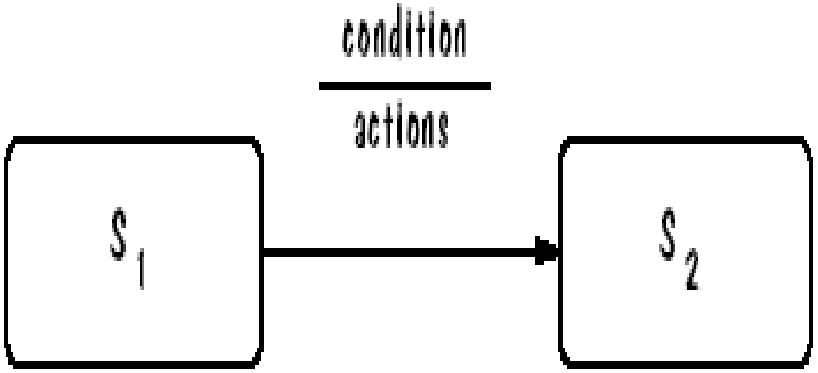
- Finite state machine model of the tunstile problem

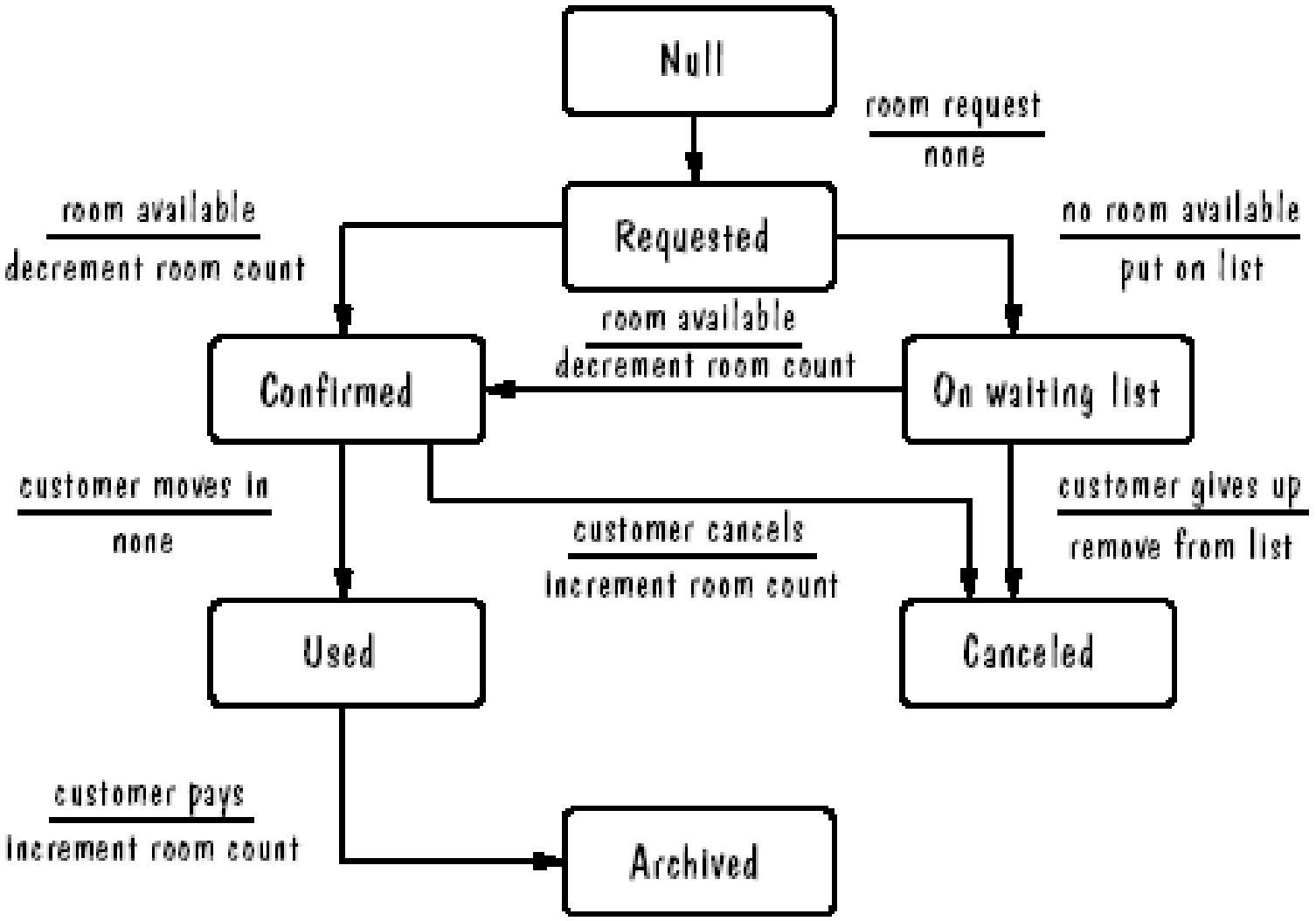


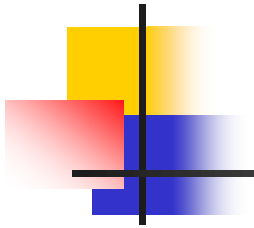


State Machine: UML State-Chart Diagram

- 描述类图中对象的动态行为。
- 建模元素
  - **Start, and Exit states**
  - **Transition --- event/action/ condition**
    - **Event(args) [condition] /action* ^object.event(args)***
    - **[] condition**
    - **/ action**
    - **^ output event**
    - *** multiple**



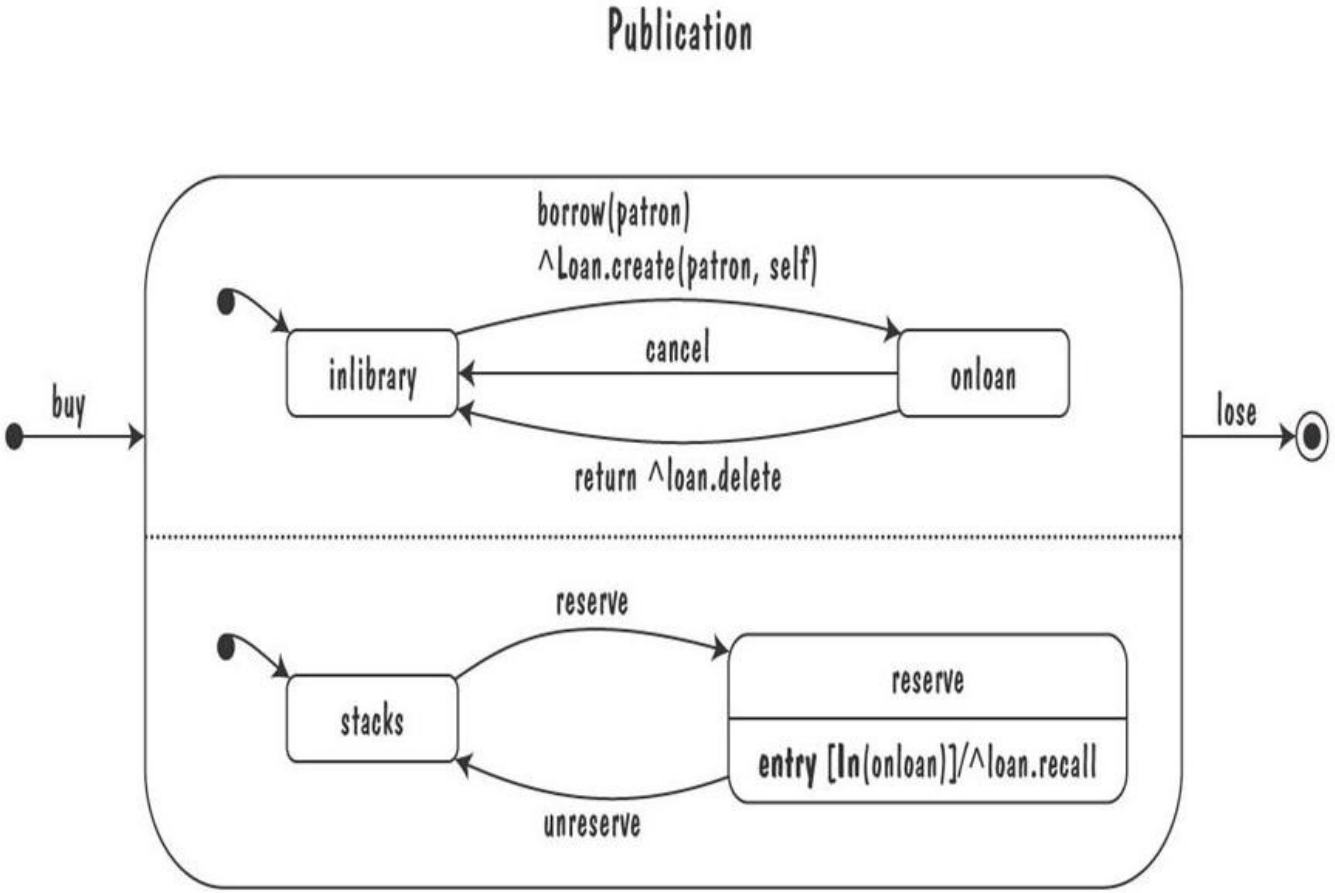

Null
room request
none
Requested
room available
decrement room count
no room available
put on list
Confirmed
room available
decrement room count
On waiting list
customer moves in
none
customer cancels
increment room count
customer gives up
remove from list
Used
Canceled
customer pays
increment room count
Archived



- State
  - State name
  - Local variables
  - Actions and activities
- Figure 4.9: UML statechart diagram for the Publication class.
- Figure 4.10: Messy UML statechart diagram for Publication class.
- Figure 4.11: UML statechart diagram for Loan class.

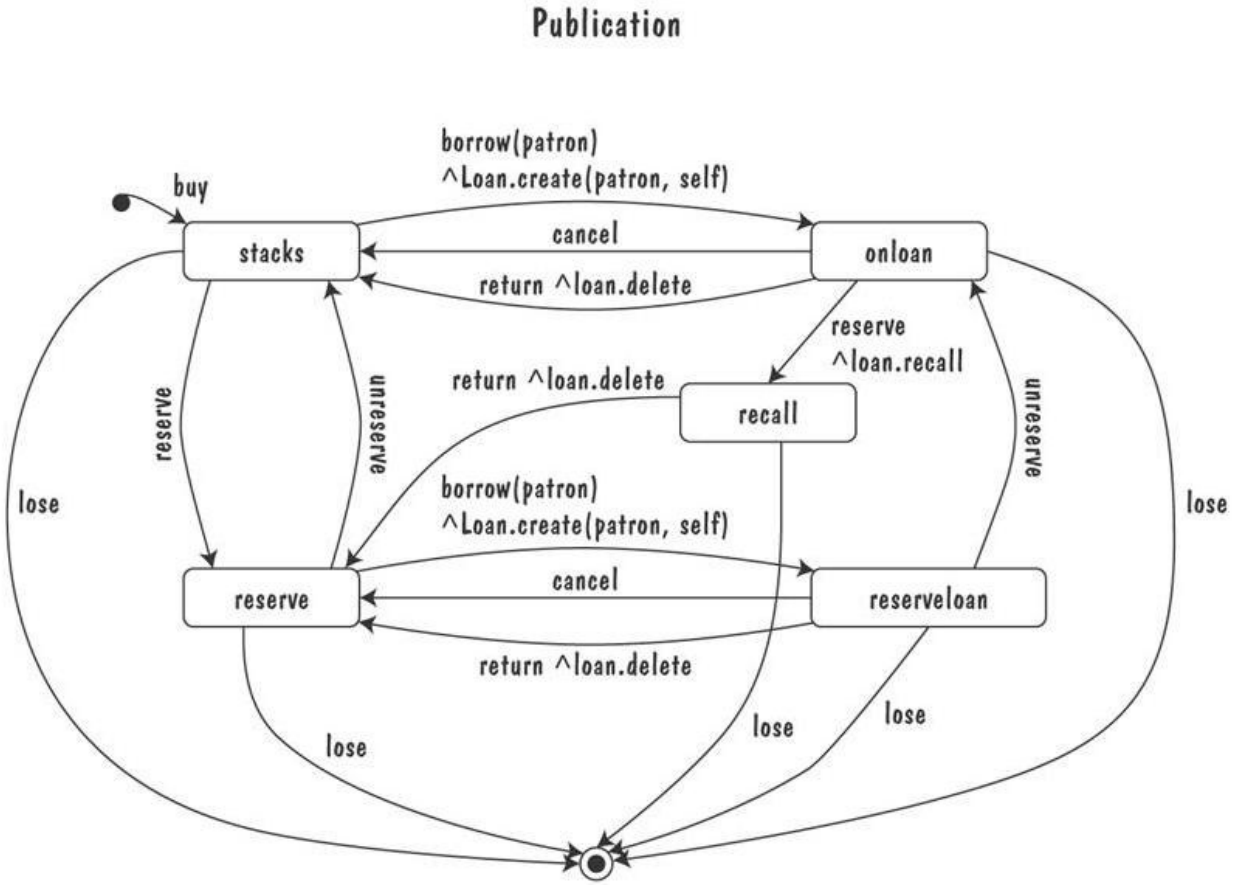
UML Statechart Diagrams (continued)

- The UML statechart diagram for the `Publication` class from the Library class model



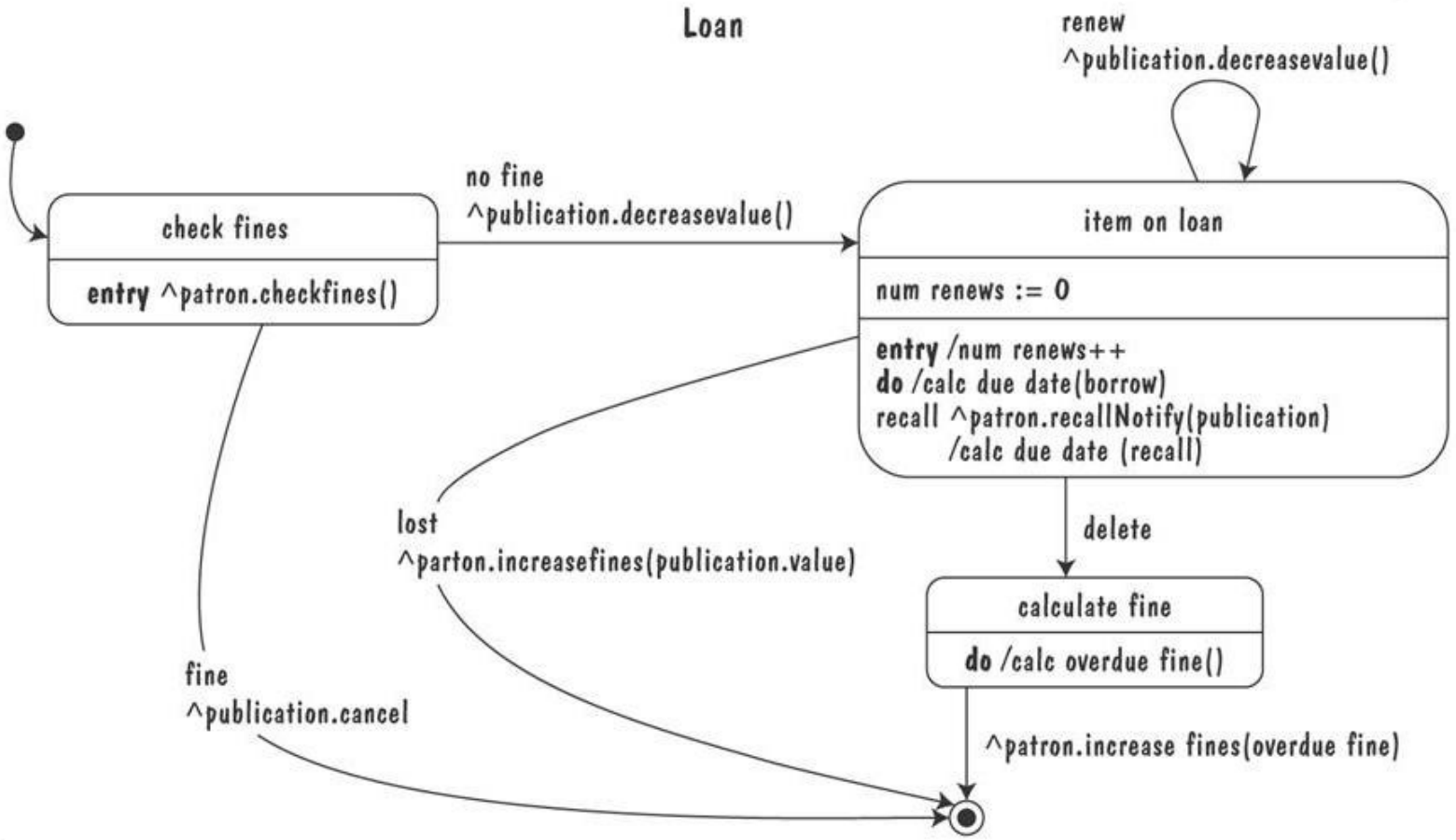
UML Statechart Diagrams (continued)

- An equivalent statechart for `Publication` class that does not make use of state hierarchy or concurrency
  - comparatively messy and and repetitive



UML Statechart Diagrams (continued)

- The UML statechart diagram for `Loan` association class illustrates how states can be annotated with local variables, actions and activities



State Machine: Petri Nets

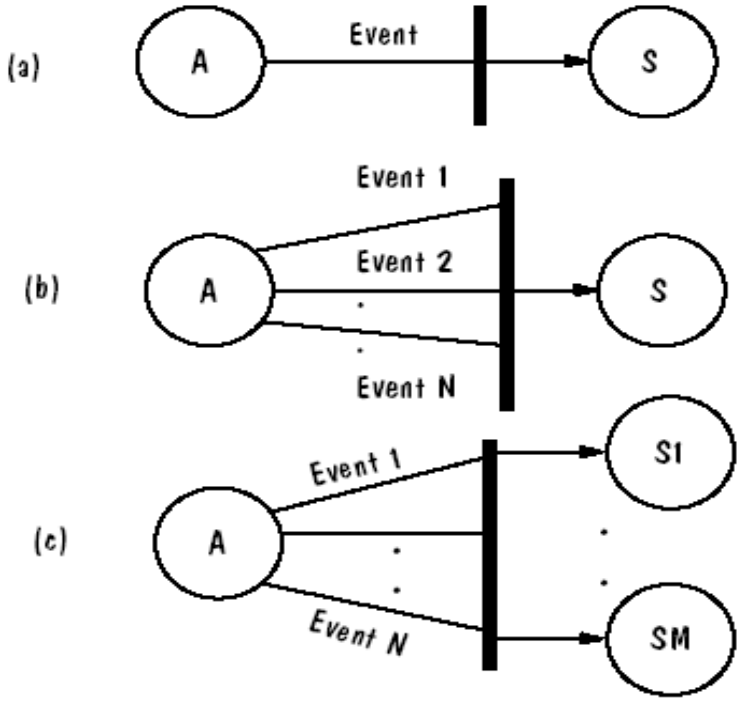
- 用途
  - **Representing concurrency**描述并发事件
    - 多个事件同时发生时，系统必须进行并发处理
    - 一个主要的问题是对事件进行同步
  - 状态转移模型
    - **several events trigger the move from one to another state :**

      F(StateA, Event1,…,EventN) -> StateS
    - **In more general:**

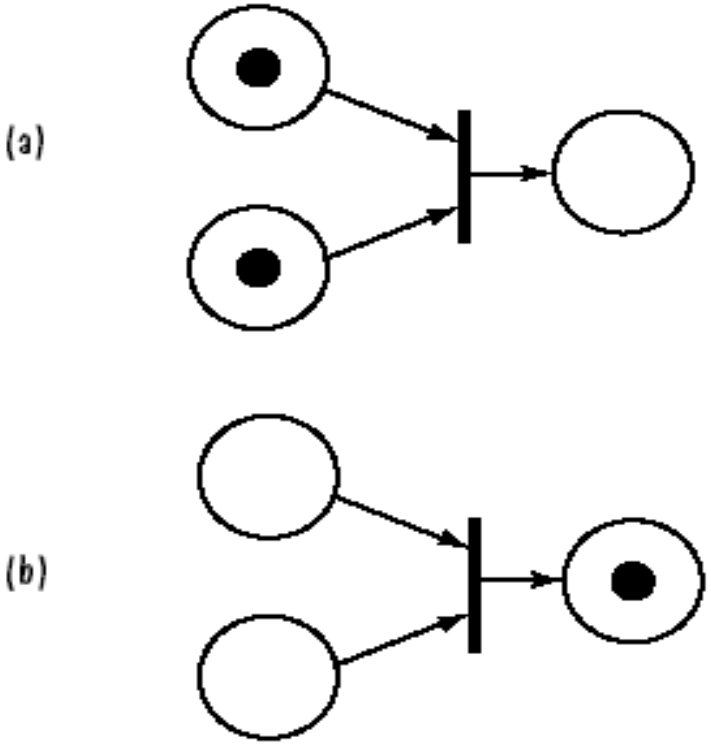
      F(StateA, Event1,…,EventN) -> State1,…,StateM

State Machines Example: Petri Nets

- A form or state-transition notation that is used to model concurrent activities and their interaction
  - Circles (*places*) represent activities or conditions 库所
  - Bars represents *transitions* 转移
  - *Arcs* connect a transition with its input places and its ouput places 转移
  - The places are populated with *tokens,* which act as enabling conditions for the transitions 令牌
  - Each arc can be assigned *a weight* that specifies how many tokens are removed from arc's input place, when the transition fires



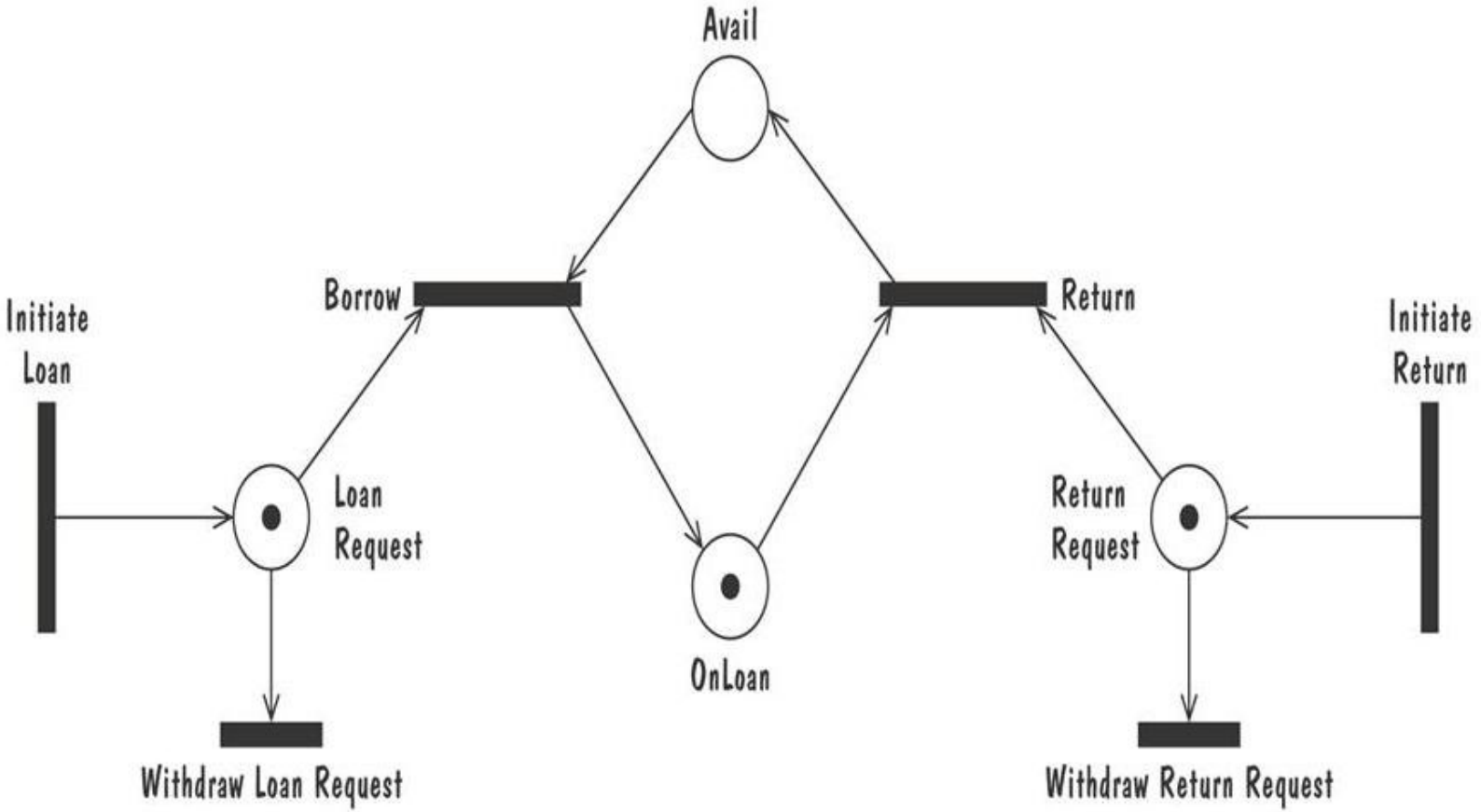
Three types of transitions



Tokens associated with firing rules

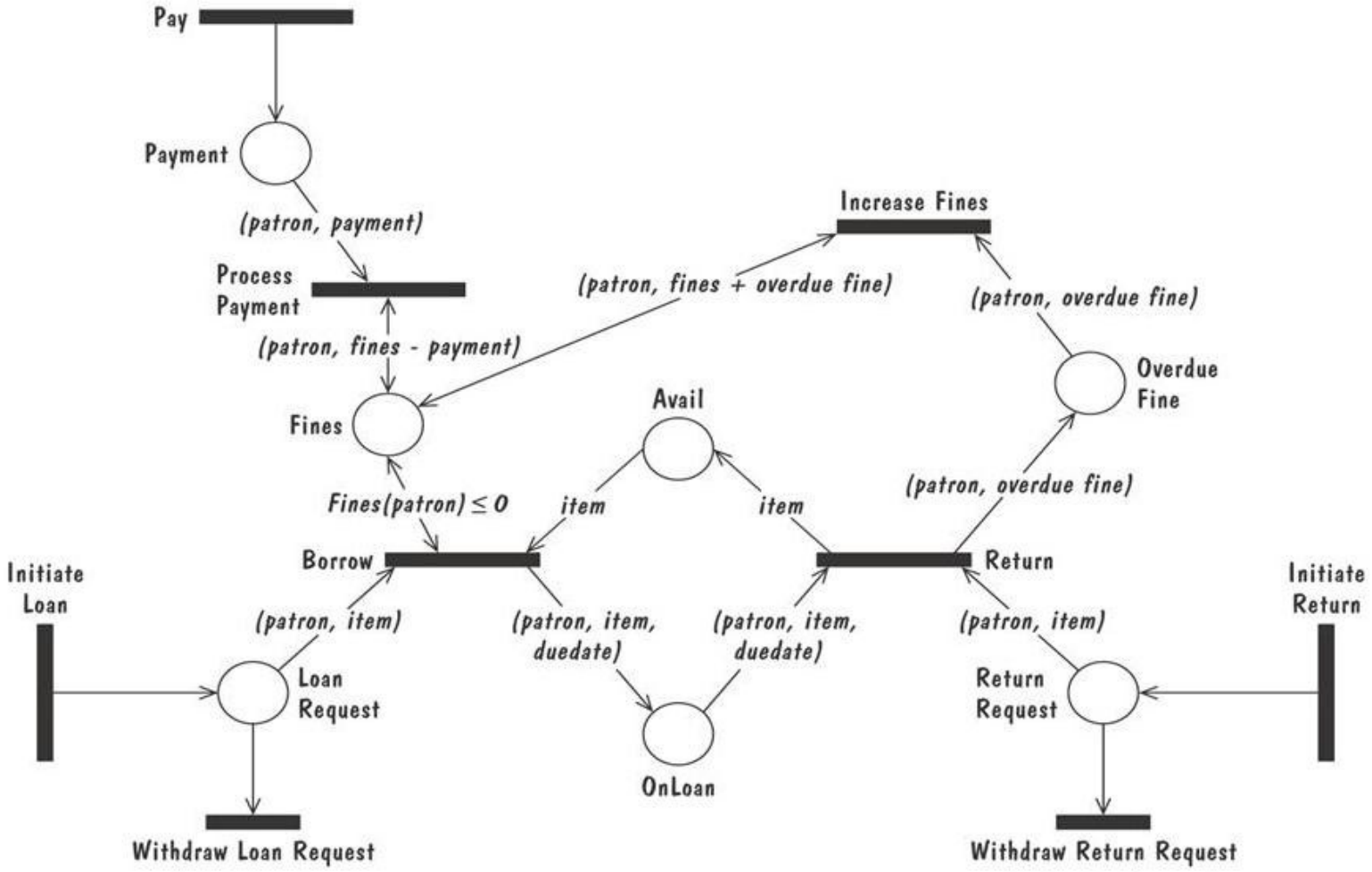
Petri Nets (continued)

- Petri net of book loan



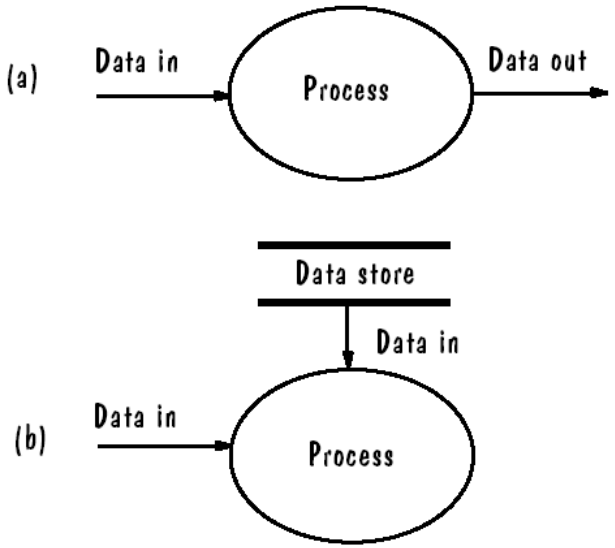
Petri Nets (continued)

- A high level Petri net specification for the library problem



4. 功能模型：Data-Flow Diagrams

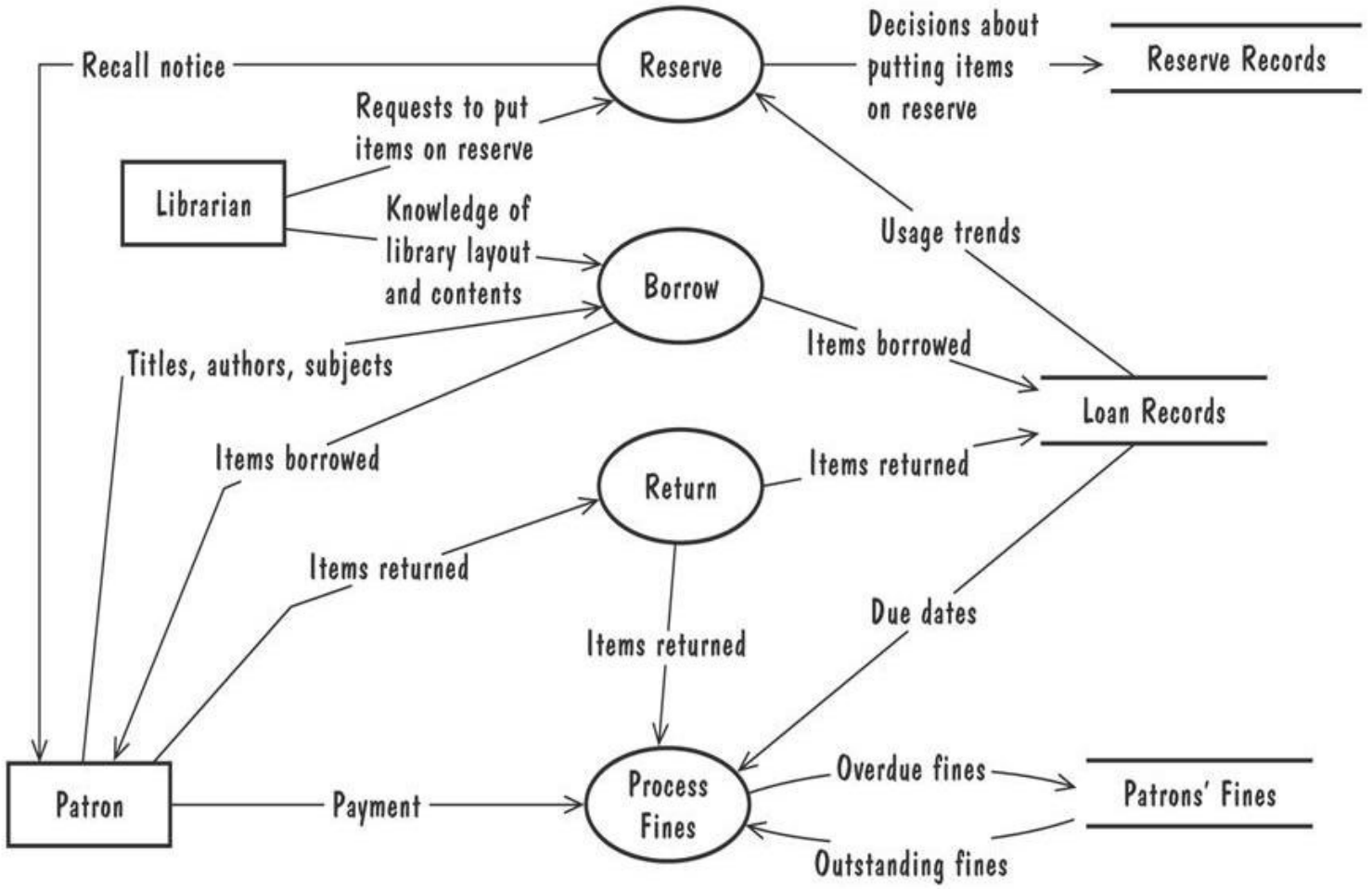
- 用途
  - 描述分层的功能模型（不同的抽象级别）
  - 功能和数据的流动（数据是怎样流入系统，经过怎样的加工、变换，怎样流出系统）
- 建模元素
  - **Process/Function** 过程/功能/加工
  - **Data flow** 数据流
  - **Data store** 数据存储
  - **Actors** 系统之外的实体



- 对**Process**进行分层，获得不同的抽象层次，注意数据流的平衡和子图的易理解性

Data-Flow Diagrams (continued)

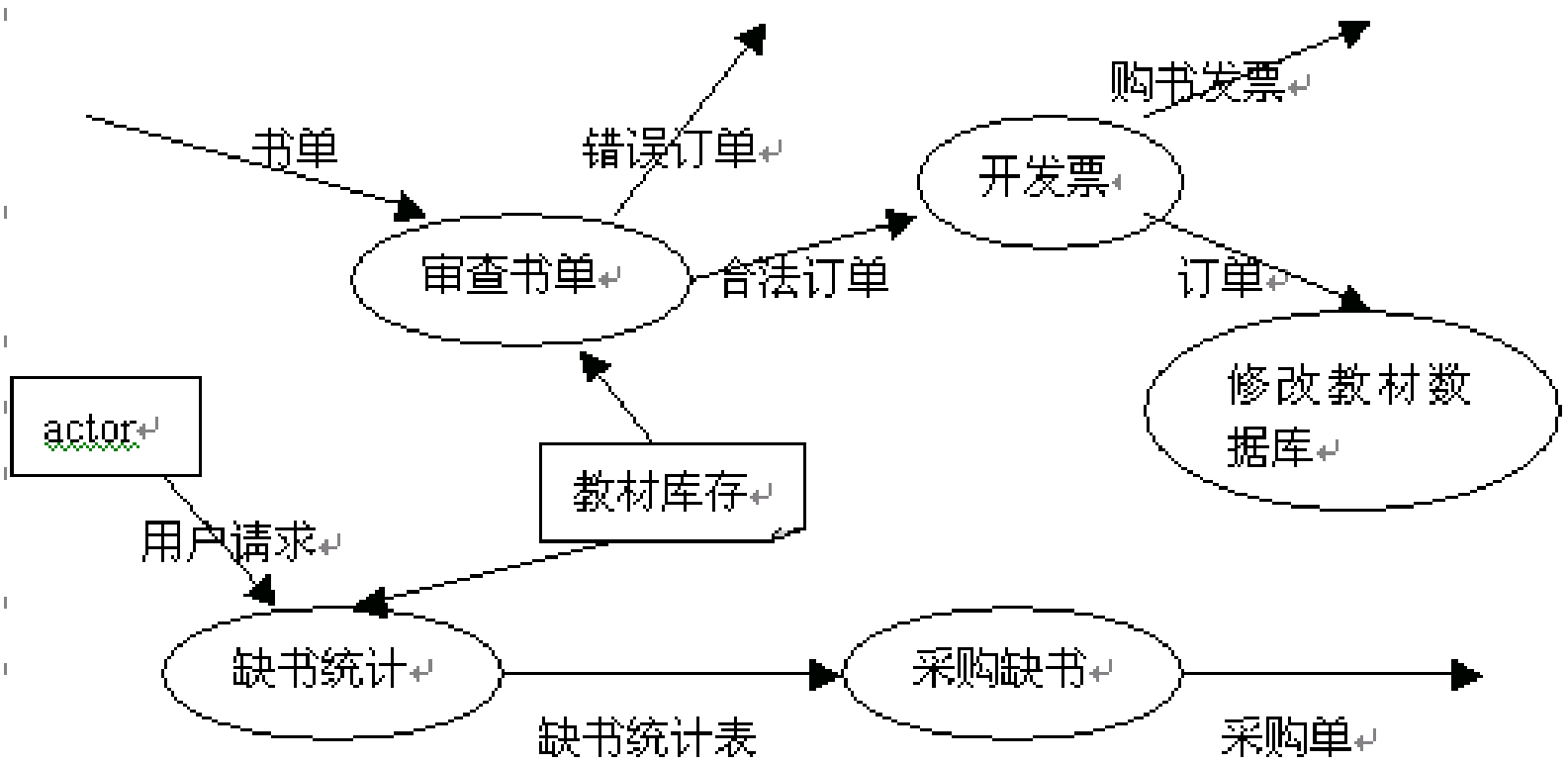
- A high-level data-flow diagram for the library problem



Requirement Definition:

用数据流描述微机教材销售系统。该系统有审查书单，开发票，修改教材数据库，缺书统计，采购缺书的功能。

Requirement Specification(DFD):



- Requirements Hierarchy
  - DFD depicts a high-level view of the system
  - Each process bubble can be represented as a separate diagram to show the details of the data transformation
  - In this way, the requirements hierarchy is formed.

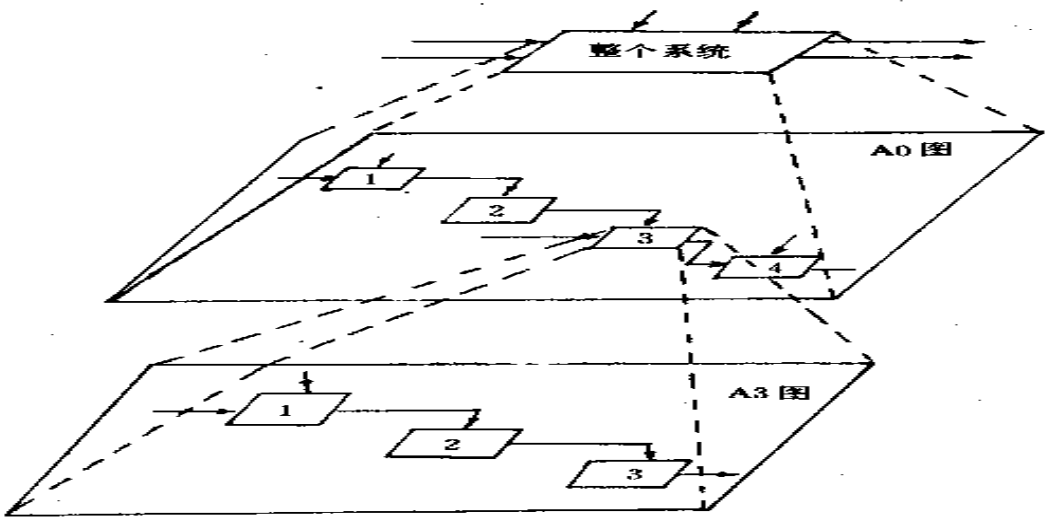
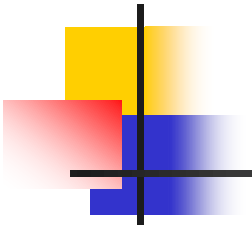


图17.2 递阶层次分解图



Data-Flow Diagrams (continued)

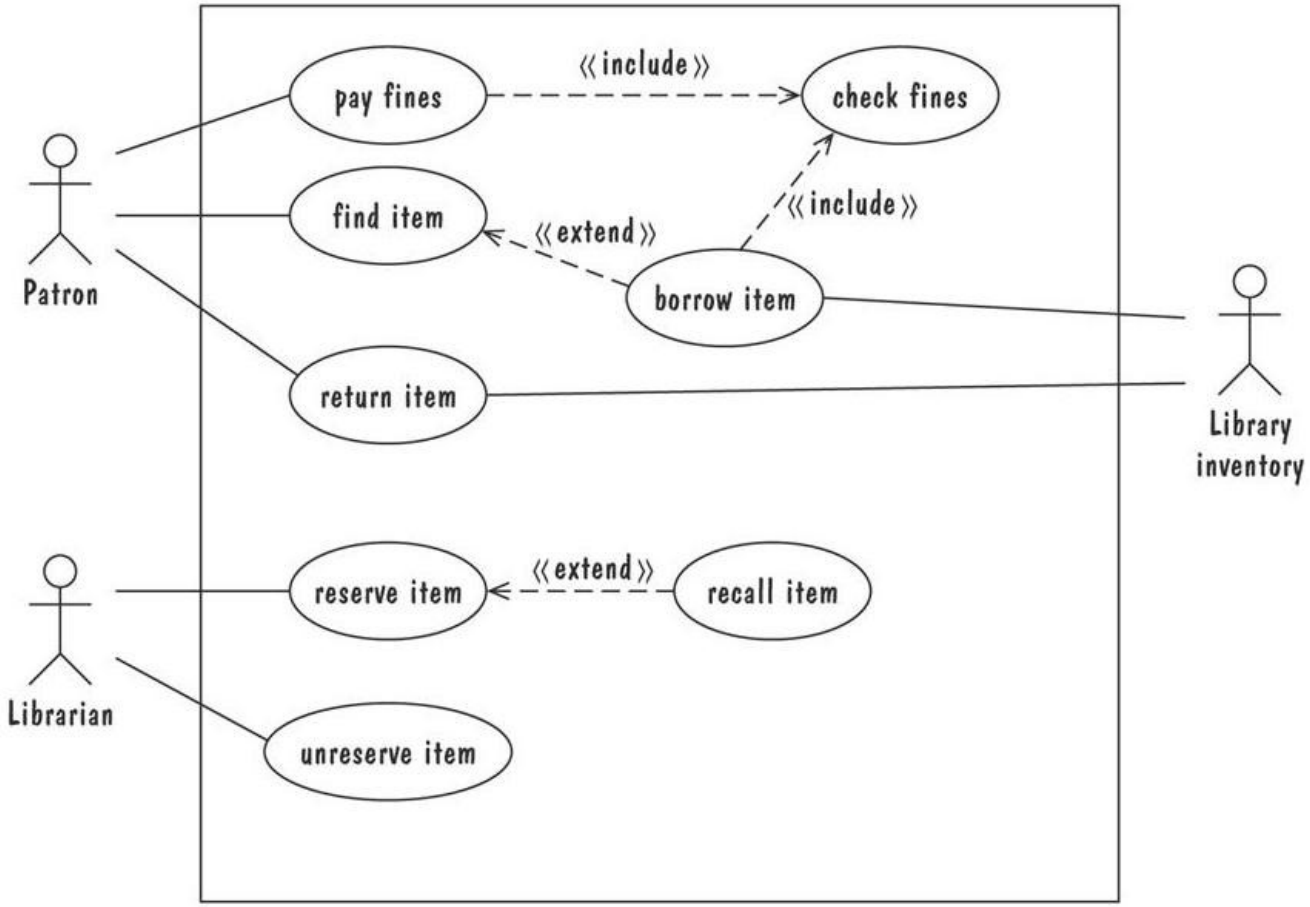
- Advantage:
  - Provides an intuitive model of a proposed system's high-level functionality and of the data dependencies among various processes
- Disadvantage:
  - Can be aggravatingly ambiguous to a software developer who is less familiar with the problem being modeled

功能模型：UML Use case Diagrams

- 用途
  - 以用户/外部实体的角度描述系统功能模型（类似顶层的数据流图）
- 建模元素
  - **System boundry** 系统边界
  - **Actor** 活动者
  - **Use case** 用例
  - **Relations** 活动者与用例、用例之间的联系
- 示例
  - **Figure 4.15 Library use cases.**

Use Cases (continued)

- Library use cases including borrowing a book, returning a borrowed book, and paying a library fine



5. Functions and Relations

- Function or Relation
  - Formal methods model requirements or software behavior as a collection of mathematical **functions** or **relations**
  - Example
    - NextState(s,e) =

unlocked	s=locked AND e=coin
rotating	s=unlocked AND e=push
locked	s=rotating AND e=rotated

    - Output(s,e) =

buzz	s=locked AND e=slug
<none>	Otherwise

  - Functions are semantically equivalent to state machine.
    - Functions can be further described with Pre-Condition and Post-Conditions.

Functions and Relations: Decision Tables 判定表

- 表达方式
  - **possible conditions satisfied by the system at a given time 所有可能的条件**
  - **Rules for reacting to stimuli when certain conditions are met and actions to be taken as a result 触发规则**

Composition of Decision Table:

Condition Stub	Condition Items
Action Stub	Action Items

- **Table 4.1, p .152**
  - **Decision table for library functions borrow, return, reserve, and unreserve**

(event) borrow	T	T	T	F	F	F	F	F
(event) return	F	F	F	T	T	F	F	F
(event) reserve	F	F	F	F	F	T	T	F
(event) unreserve	F	F	F	F	F	F	F	T
item out on loan	F	T	-	-	-	F	T	F
item on reserve	-	-	-	F	T	-	-	-
patron.fines > $0.00	F	-	T	-	-	-	-	-
(Re-)Calculate due date	X						X	
Put item in stacks				X				X
Put item on reserve shelf					X	X		
Send recall notice							X	
Reject event		X	X					

Requirement Definition:

如果金额超过500元，又未过期，则发出批准单和提货单；如果金额超过500元，但过期了，则不发出批准单；如果金额低于500元，则不论是否过期都发出批准单和提货单，在过期的情况下还需发出通知单。

金　额 状　态	>500 未过期	>500 已过期	<=500 未过期	<=500 已过期
发出批准单	X		X	X
发出批准单	X		X	X
发出批准单				X

Functions and Relations: Decision Tables 判定表

- This can generate very large tables
  - The number of the states is equal to the number of combinations of conditions, which is 2^n
  - There may exist redundant rules which are covered by other rules
  - We can reduce the size and make them easier to understand
- What can the table tell us about the requirements ?
  - If every possible set of conditions results in an action, then the requirements specification is complete
  - And we can examine the table for consistency and eliminate any conflicting cases.

6. Logic

- 前面的方法（除**E-R**图）：**operational notation** 基于操作，针对特定情况，系统如何反应 **case-based behavior**
- **Operational VS Descriptive requirements.**
  - **Describe a problem or a solution In terms of**
    - **Its *operational or Situational behavior*; VS**
    - **of *its properties or its invariant behavior*.**
- **A Logic consists of a language for expressing descriptive properties.**
  - **First-order-logic, comprising typed variables, constants, functions, sets, predicates, operators.**
  - **Example:**
    - **num_coin >= num_entries //invariant**
    - **(num_coin > num_entries) => (barrier = unlocked)**
    - **(barrier = locked) ⇔ ┐may-enter**

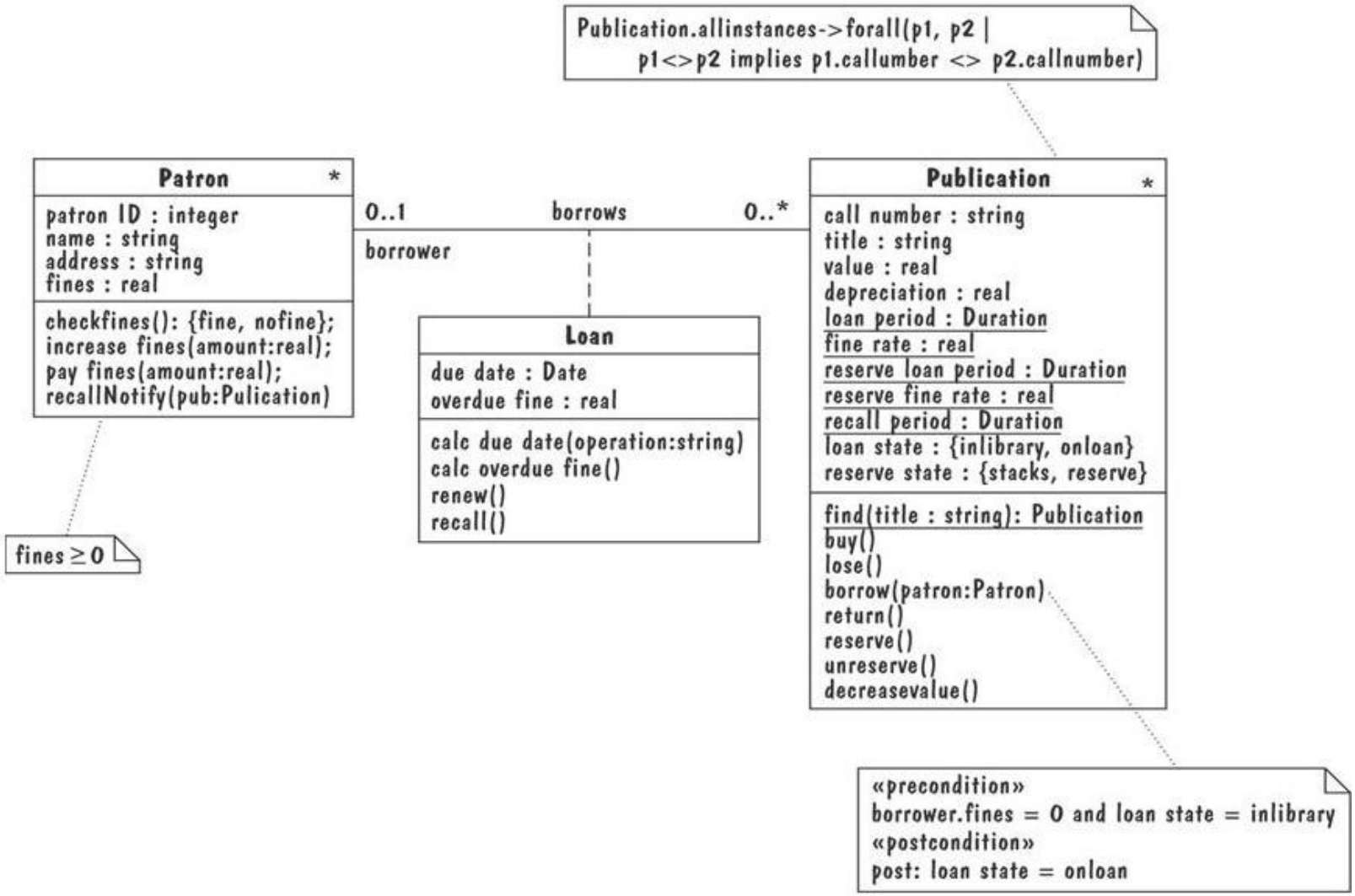
Temporal Logic 时间逻辑

- □f : f is true now and throughout the rest execution;
- ◊ f : f is true now or at some future point in the execution;
- ○ f : f is true in the next point of the execution;
- f W g : f is true until a point where g is true, but g may never be true.
- Example:
  - □(insert_coin => ○(may_enter W push))
  - □(Vn(insert_coin ^ num_coins=n) => ○(num_coins=n+1))

Logic: OCL(Object Constraint Language)

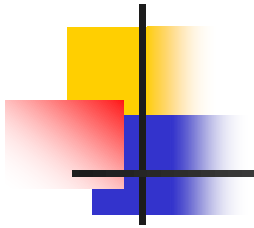
- 用途：
  - 描述对象模型的约束
  - Not originally designed for UML, now is tightly coupled with UML, and is annotated as labels.
- 建模符号：
  - Construct *allinstances*: return all instances of a class;
  - Construct *forall*, *and*, *implies*;
  - Symbol ->(→): applies the attribute or operation of its right operand to all of the objects in its left operand;
- Example: Fig.4.18, p.180

Library classes annotated with OCL properties



Logic: Z notation

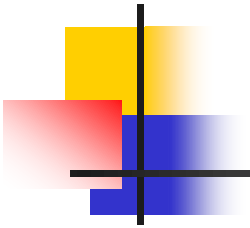
- 特点 A Formal Specification Language
  - It express requirements in mathematical way
    - They can be evaluated using proofs and automated techniques
  - It combines abstract data modeling with set theory and first-order predicate logic
    - Used to specify system states and valid state changes
  - Tools can be used to check reachable states, deadlocks, non-determinism, and generate a finite-state machine



- 建模符号
  - Mapping, partial mapping, domain substraction operator
  - A Unprimed variable represent a state before an operation is performed, and A primed variable represent a state after the operation performed
  - 示例See example in p.182
- Formal specification is encouraged
  - Especially for the developments of safety critical systems
  - The mathematical proof technique can reveal significant problems in the requirements
  - Where they are more easily fixed than after its implementation

4.6 Reqtuirements And Spec Languages

- 迷惑：Why the software-engineering community had developed so many types of software models?
- 原因：Each paradigm describes software models from a different perspective:
  - Entities and relationships, events, traces, states, functions, properties, …
- 使用：Each paradigm has its own strength
  - A complete specification may consist of several models, each illustrates a different aspect.

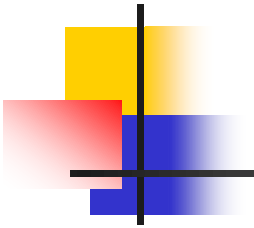


UML

- Contains the following notations
  - Use-case diagram (a high-level DFD)
  - Class diagram (an E-R diagram)
  - Sequence diagram (an event trace)
  - Collaboration diagram (an event trace)
  - State diagram (a state-machine model)
  - OCL properties (logic)
- Beyond requirements
  - Package diagram
  - Component diagram
  - Deployment diagram
  - Extension facilities

4.7 Prototyping Requirements

- Customers or users may directly be involved in the analysis and designing processes
  - To make sure the requirements are clear, complete and correct
  - Customers know what is needed or wanted
    - But they are not certain whether the requirements are realistic
    - We should investigate options to find a feasible solution
    - Prototyping is a way to help us



Prototyping作用/目的

- To elicit the details of proposed system
- To solicit feedback from potential users about
  - what aspects they would like to see improve
  - which features are not so useful
  - what functionality is missing
- Determine whether the customer's problem has a feasible solution
- Assist in exploring options for otimizing quality requirements

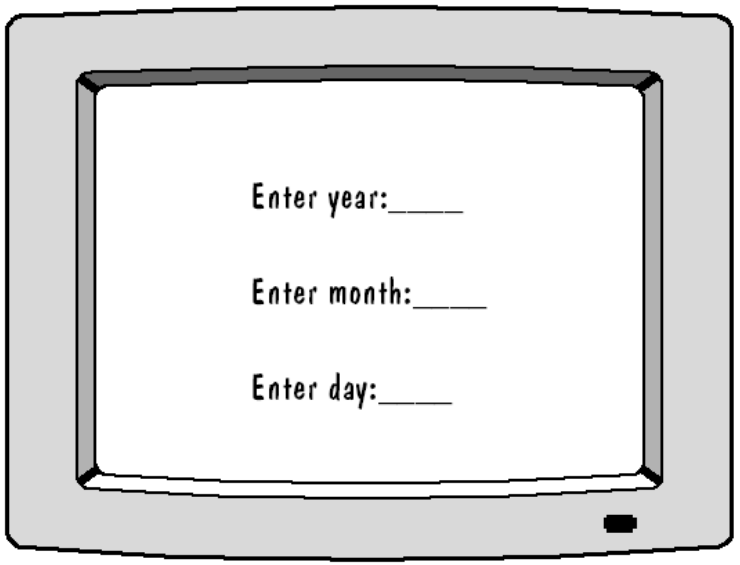


Figure 4.23 First interface prototype

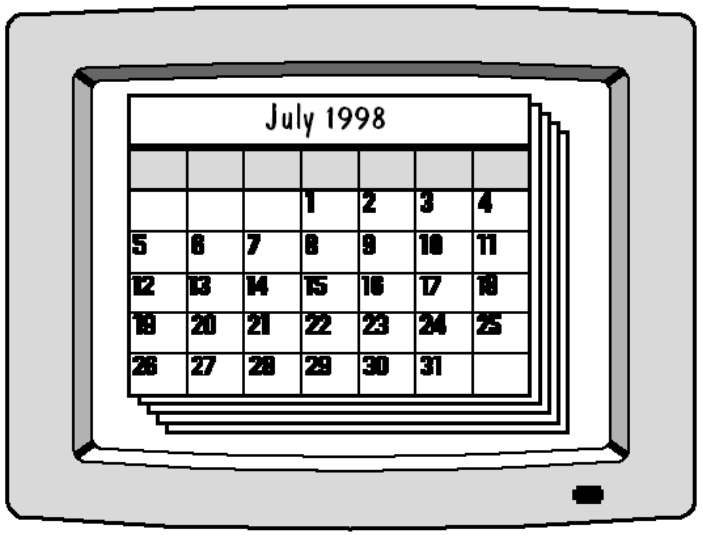


Figure 4.24 Second interface prototype

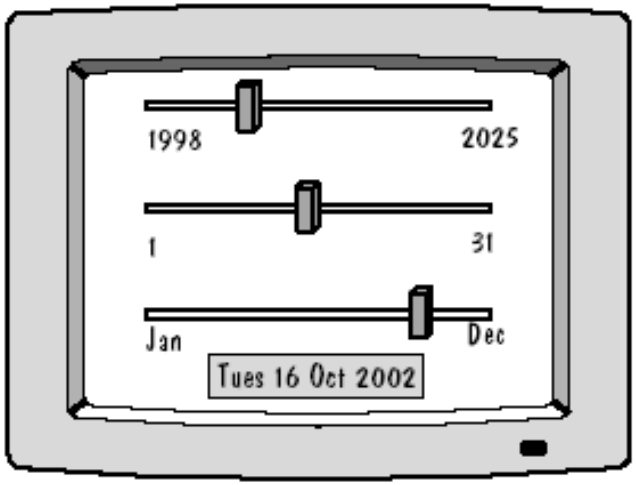
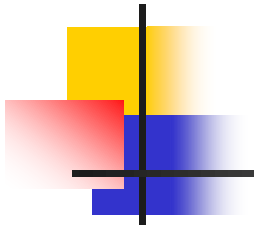


Figure 4.25 Third interface prototype



- Throw-away prototype 抛弃型原型
  - Software developed to learn more about a problem or explore the feasibility or desirability of possible solutions
  - It is exploratory, not intended as a part of the final product
  - Allow us to write “quick-and-dirty”
- Evolutionary prototype 演化型原型
  - Developed not only to help us answer questions but also to be incorporated into the final product
  - Prototype has to eventually exhibit the quality requirements of the final product, and these qualities cannot be retrofitted
- Both techniques are sometimes called rapid prototyping

4.8 Requirements Documentation

- The Necessity for Requirements Documentation
  - We must keep a set of documents requirements, no matter what method we choose for defining requirements
  - We and the customers will refer to them throughout developments and maintenance
- 需求的书写要求
  - They must be organized in such a way that they can be tracked throughout 容易跟踪需求条目
  - Illustration and diagrams should be used to make them clear enough for idea communication, and they must be precise and consistent with the requirements text 图表精确、与文字一致
  - The level at which the requirements are written is important too. 描述的抽象层次

Problems with Requirements Specification

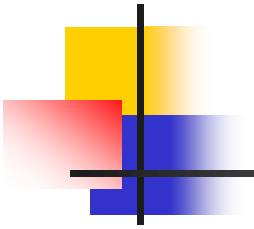
- See Sidebar 4.6, p.171
- Uneven level of specification 不均衡的描述
  - Different Levels of detail, too high or too low
  - Use different formats or writing styles
  - Over-specified requirements
    - When analysts identified particular types of computers and programming language, …
  - Under-specified requirements
- Recommendations
  - Each clause contains only one requirement
  - Avoid having one requirement refer another
  - Collect like requirements together

1. Requirements Definition

- First, outline the general purpose and scope of the system 系统目标和范围
- Give Background and rational of the system development 系统开发的背景和理念
  - Current methods and procedures in enough detail for systems to replace an existing one
- Describe the essential characteristics of an acceptable solution 解决方案的主要特征
- Describe the environment in which the system will operate 操作环境
- Outline a description of the customer's proposal. 客户的期望和要求（约束）
- List any assumptions we make about how the environment behaves. 假定和前提条件

2. Requirements Specification

- The Specification covers exactly the same ground as definition. 描述范围：与需求定义一致
- It is written from the developer's perspective or in terms of the system's interface 描述角度：系统接口
  - Where definition is written in terms of the customer's vocabulary, referring to objects, states, events ,and activities in the customer's world.
- Rewriting the requirements so that they refer only to those real-world objects (states, events, actions) that are sensed or actuated by the proposed system 描述对象：仅保留系统关心的对象



内容

- In the interface, we describe all inputs and outputs in detail,
  - including their sources, destination, value ranges and data formats.
- Next, we restate the required functionality in terms of interfaces' inputs and outputs.
- Finally, we devise fit criteria for each of the customer's quality requirements.

IEEE Standard for SRS Organized by Objects

- Organizations such as IEEE have standards for the content and format of the requirements documents

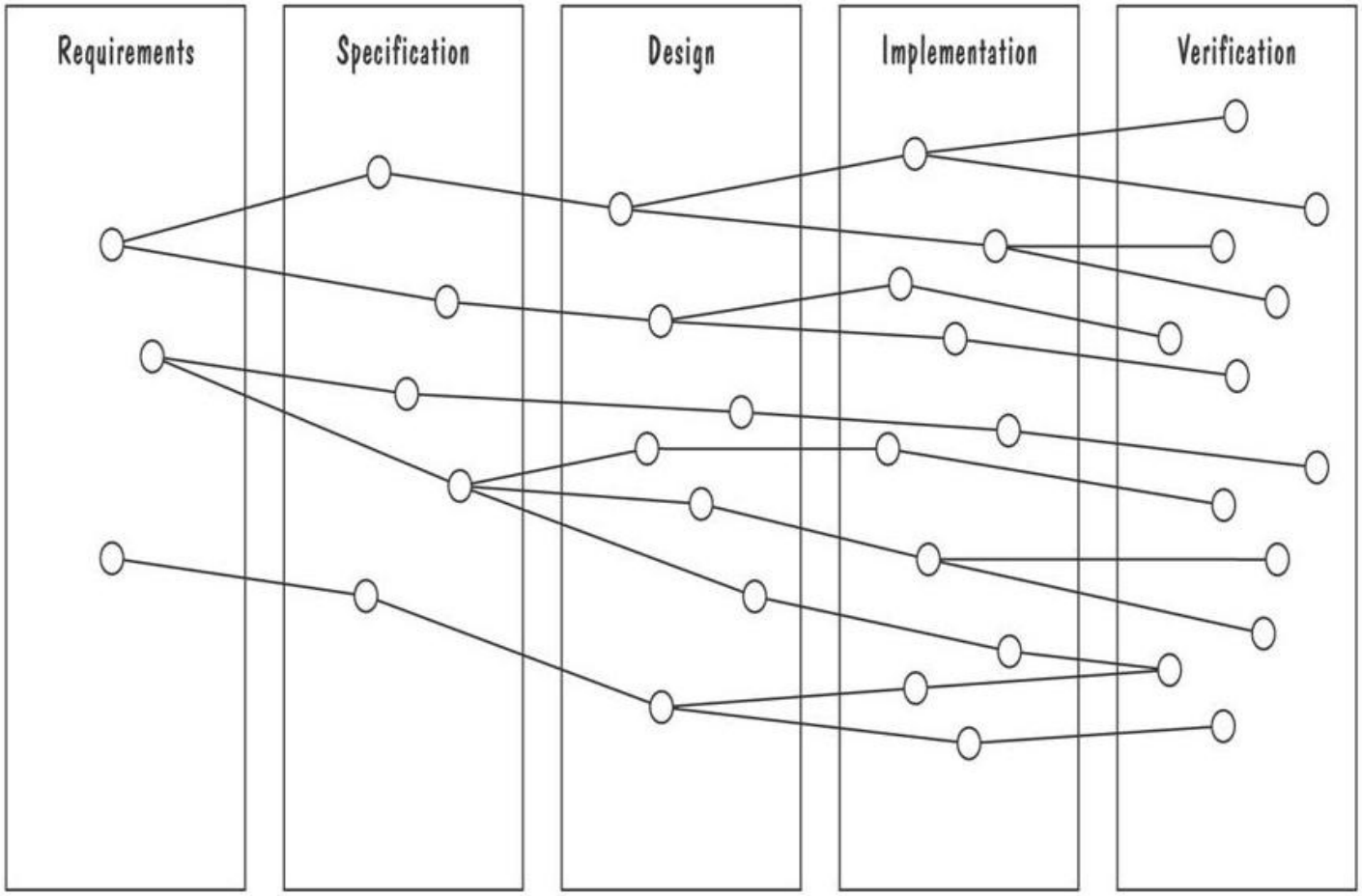
1. Intodruction to the Document
   - 1.1 Purpose of the Product
   - 1.2 Scope of the Product
   - 1.3 Acronyms, Abbreviations, Definitions
   - 1.4 References
   - 1.5 Outline of the rest of the SRS
2. General Description of Product
   - 2.1 Context of Product
   - 2.2 Product Functions
   - 2.3 User Characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and Dependencies
3. Specific Requirements
   - 3.1 External Interface Requirements
     - 3.1.1 User Interfaces
     - 3.1.2 Hardware Interfaces
     - 3.1.3 Software Interfaces
     - 3.1.4 Communications Interfaces
   - 3.2 Functional Requirements
     - 3.2.1 Class 1
     - 3.2.2 Class 2
     - …
   - 3.3 Performance Requirements
   - 3.4 Design Constraints
   - 3.5 Quality Requirements
   - 3.6 Other Requirements
4. Appendices

3. Process Management & Reqs Traceability

- There must be direct correspondence between definition and specification.
  - It is here that the process management methods used throughout the life cycle begin.
  - Process management is a set of procedures that track
    - The requirements that define what the system should do;
    - The design modules that are generated from the requirements;
    - The program code that implements the design;
    - The tests that verify the functionality of the system;
    - The documents that describe the system.
  - See Fig.4.27, p.198

Development Activities

- Horizontal threads show the coordination between development activities

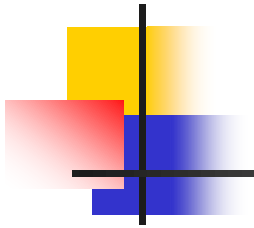


4.9 Validation and Verification

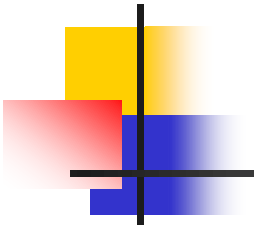
- 需求文档
  - 开发者和客户之间的技术合同（开发者将要交付什么）
  - 设计人员的依据（将要构建什么）
- 确认(validation)和验证(verification)的目的
  - 分析人员和客户准确、全面的理解彼此的意图
  - 我们的意图完全反映在需求文档中
- **Requirement Validation 需求确认**
  - **需求定义文档(requirement definition)精确的反映出客户（所有利益相关人员）的要求**
  - ***We build the right system!***
- **Requirement Verification 需求验证**
  - **验证需求文档和产品是一致的（比如代码与设计、设计与需求等等），在需求过程中，即为验证需求规格和需求定义的一致性。**
  - **We build the system right!**

4.9.1 Validation

- Requirement Validation 需求确认
  - Validation makes sure that the the requirements will meet the customer's needs
- Techniques to perform the validation常用技术
  - The choice of the technique depends on your experience, preference, appropriateness for your definition and specification techniques
  - See Table 4.3, p.199
  - Checklist
  - Walkthrough
  - Formal inspection
  - Review



- Requirements review 需求评审
  - Is a simple way to check the requirements
  - In which representatives from our staff and the customer's examine the requirements
    - Representatives include those who will operate the system, prepare the input, use the output, managers of these employees, …
- 需求评审的内容
  - Review the stated goals and objectives of the system
  - Compare the requirements with the goals and objectives
  - Review the environment in which the system is to operate
  - Review the information flow and proposed functions
  - Assess and document the risk, discuss and compare alternatives
  - Testing the system: how the requirements will be revalidated as the requirements grow and change



需求评审的内容

- When a problem is identified
  - Document it, determine its cause, and take action to fix the problem before design begins.
    - Example : customer may set a reliability or availability goal that developers deem impossible to meet
    - Sidebar 4.8 discuss the nature and number of requirements related problems likely to find
- When the validation is completed
  - The requirements process is completed

4.9.2 Verification

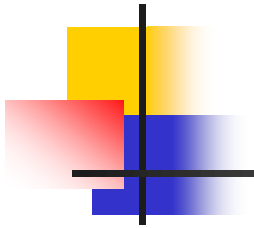
- Requirement Verification 需求验证
  - Is the process of determining that the specification is consistent with the requirements definition
- Make sure that if we implement a system that meets the specification, then the system will satisfy the customer's requirements
- Ensure that each requirement in the definition document is traceable to the specification
- Sidebar 4.9 Computer-Aided Verification

- 需求承诺
  - 需求承诺是指开发方和客户方的责任人对通过了正式技术评审的《产品需求规格说明书》作出承诺，该承诺具有商业合同的效果。
  - 需求承诺的“八股文”如下：
    - 本《产品需求规格说明书》建立在双方对需求的共同理解基础之上，我同意后续的开发工作根据该《产品需求规格说明书》开展。如果需求发生变化，我们将按照“变更控制规程”执行。我明白需求的变更将导致双方重新协商成本、资源和进度等。
    - 甲方签字　　乙方签字
  - 人们在作出承诺之前务必要认真阅读文档，一定要明白签字意味着什么。

4.10 , …,4.18 Are Omitted

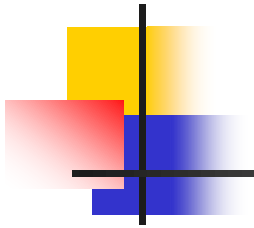
本章小结

- 4.1 The requirements process
  - Requirement 需求及其目的
  - The process of determining the requirements 需求获取的过程
  - Heavy需求过程与Agile需求过程的区别
- 4.2 Requirements elicitation
  - 需求过程中的stakeholders
  - 需求获取的各种途径
- 4.3 Types of requirements
  - Functional requirements 功能需求

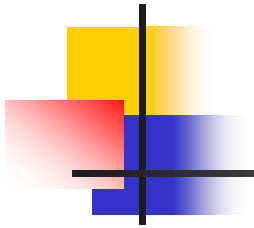


- Quality requirement质量需求/ Nonfunctional requirements非功能需求
- Design constraint 设计约束
- Process constraint 过程约束
- 按照优先级划分的三类需求
- 怎样使需求可测试testable
- 需求文档的作用（对客户、分析人员...）
- 需求定义和需求规格两种需求文档的区别和联系

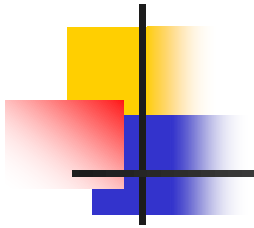
- 4.4 Characteristics of requirements
  - 8个特征及其含义



- 4.5 Modeling notations
  - E-R图/UML 类图
  - Event traces 事件跟踪/Message sequence chart/UML sequence diagram
  - 状态机/UML 状态图/Petri Nets
  - DFD 数据流图/UML用例图
  - Decision tables 判定表/功能及联系图



- 4.7 Prototyping requirements
  - 原型的作用
  - 抛弃型原型
  - 演化型原型
- 4.8 Requirements documentation
  - 需求定义文档的主要内容
  - 需求规格文档的主要内容
  - 过程管理对2种文档的相互跟踪



- 4.9 Validation and verification
  - 需求确认
  - 需求确认的常用技术、手段
  - 需求评审的有关内容
  - 需求验证

本章作业

宠物商店PetStore是一个电子商务网站。以下是功能需求：

客户分为匿名客户和注册客户，对于注册客户以会员方式管理，登记并管理其个人信息，根据其消费积分分为金牌会员、银牌会员、普通会员，以享受不同程度的优惠。

通过浏览器，匿名客户可以查询宠物；创建账户并登陆后才能够使用购物车、创建订单、提交订单、通过信用卡支付等购物活动。每个客户可以同时拥有多个订单，但只能有一个购物车，订单基于购物车内的商品创建。

宠物商店的工作人员能够接受或拒绝客户提交的订单、处理订单、发订单给供应商、接受供应商返回的配送结果。

注册客户可以随时查询其订单状态，当订单尚未处理时，可以取消订单；订单一旦开始处理，不允许客户取消。

本章作业

1. 使用UML用例图表达以上需求。
2. 使用UML类图描述此系统的对象模型。
3. 分析系统中订单的各种状态，并用状态图描述。