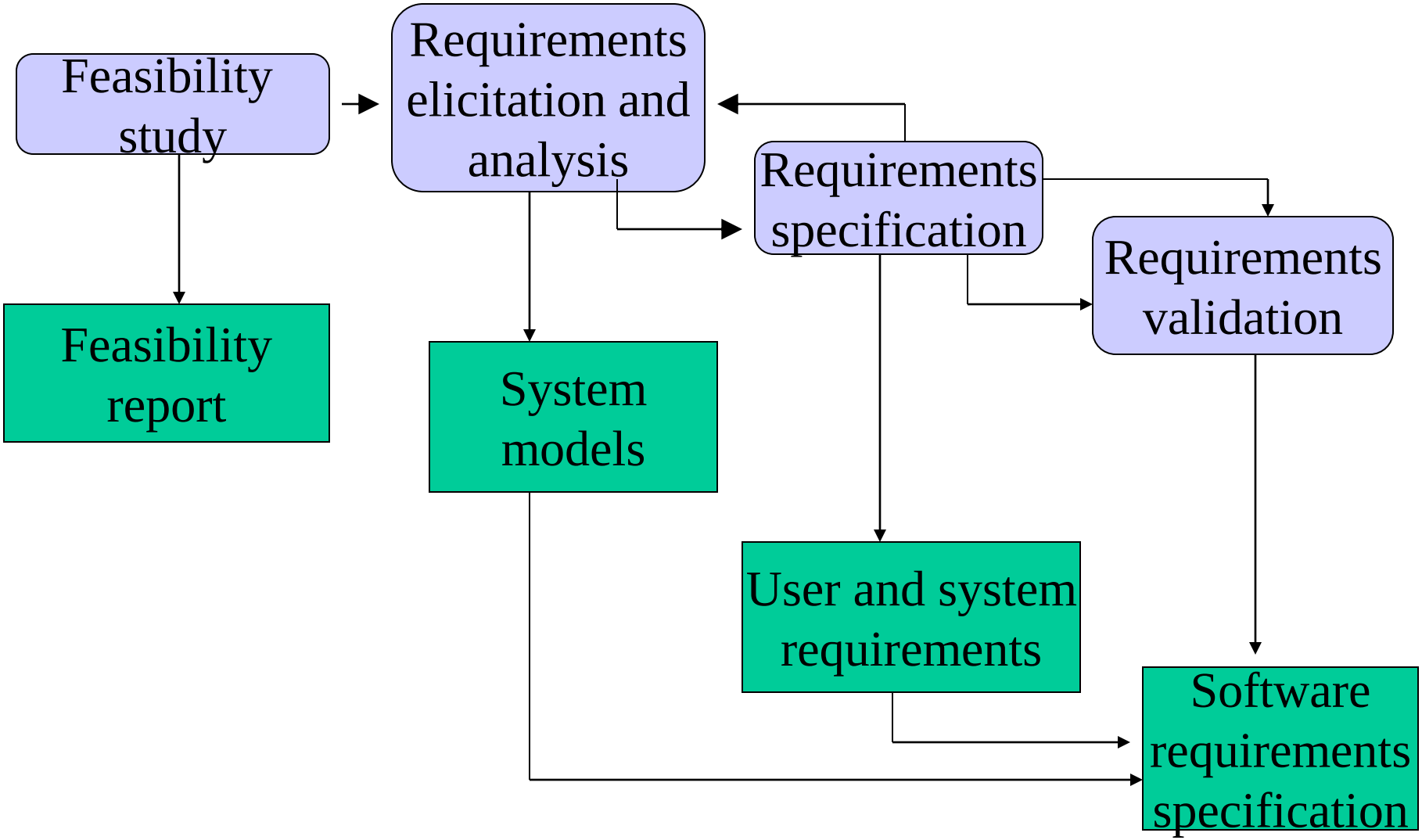
Software Engineering

Requirements and Specification

Learning Outcomes

- Be able to identify the requirement engineering process
- Be able to structure a requirements document
- Be able to write verifiable functional and non-functional requirements
- Be able to describe evolution of requirements
- Be able to identify requirements validation process

The Requirements engineering process



Requirements Engineering Process

Feasibility Study

A feasibility study is a short, focused study which aims to check the feasibility of the project considering resources, cost/benefit, technology availability etc.

Requirement elicitation and analysis

In this activity, technical software development staff work with customers and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hardware constraints and so on.

Requirements specification

A detailed and precise description of the system requirements is set out to act as a basis for a contract between client and software developer.

Requirements validation

In this activity, checks should be carried out to make sure that the requirements are accurate and complete. It is necessary to check the correctness of the specification of requirements.

Requirements Documents

User requirements (Requirements definition)

These are statements in a natural language plus diagrams, of what services the system is expected to provide and constraints under which it must operate.

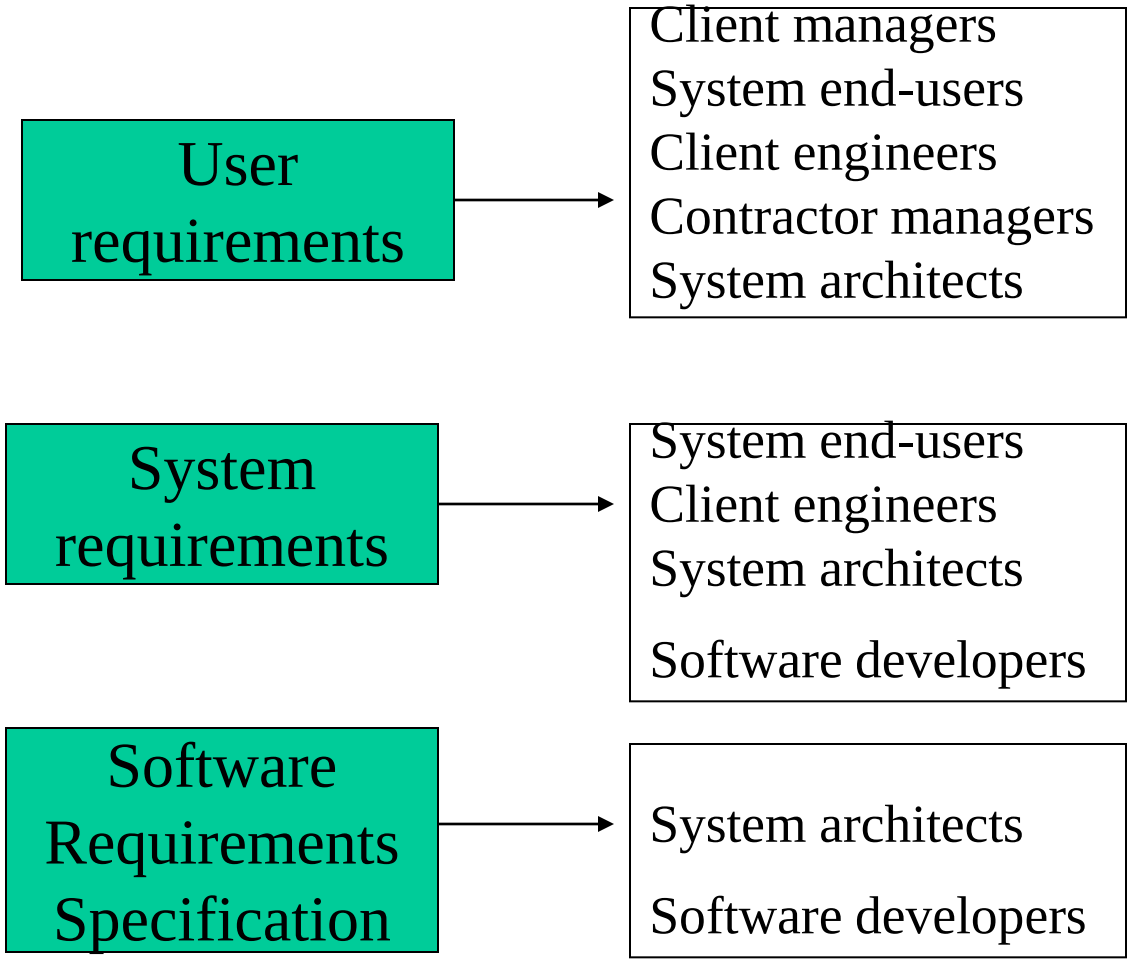
System requirements

The system services and constraints in detail. This is detailed functional specification. It may serve as a contract between the system buyer and the software developer.

Software Requirements Specification (SRS)

A detailed and technical specification of the requirements. This is the basis for design and implementation. This is an abstract description of the software.

Readers of Requirements Documents



An Example

User Requirements Definition

1. The software must provide a means of representing and accessing external files created by other tools.

System requirements specification

1.1 The user should be provided with facilities to define the type of external files.

1.2 Each external file type may have an associated tools which may be applied to the file.

1.3 Each external file type may be represented in a specific icon on the user's display.

1.4 Facilities should be provided for the icon representing an external file type to be defined by the user

Type of Requirements

Functional Requirements

These are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

Non-functional Requirements

These are constraints on the services or functions offered by the system.

Domain Requirements

These are requirements that come from the application domain of the system and that reflect characteristics of the domain.

Functional requirements

Functionality or services that the system is expected to provide.

Example:

Some functional requirements of a university library system

1. The user should be able to search for a library item by specifying a key word.
2. The library staff member should be able to issue a library item by scanning the bar codes of the library item and the student card.
3. Students can reserve a library item on line.

Some non-functional Requirements

Product Requirements

- Performance requirements
- Reliability
- Portability
- Interface requirements

Organizational Requirements

- Delivery requirements
- Implementation requirements
- Standards requirements

External requirements

- Interoperability requirements
- Ethical requirements
- safety requirements

Non- functional requirements

Examples

Product requirement

The system should be easy to use by non-experienced users and hence should provide a graphical user interface.

Organizational requirement

The system development process and deliverable documents shall confirm to the process and deliverables defined in ISO 9000 and IEEE.

External requirement

The system shall not disclose and personnel information about customers apart from their name and reference number to the operators of the system.

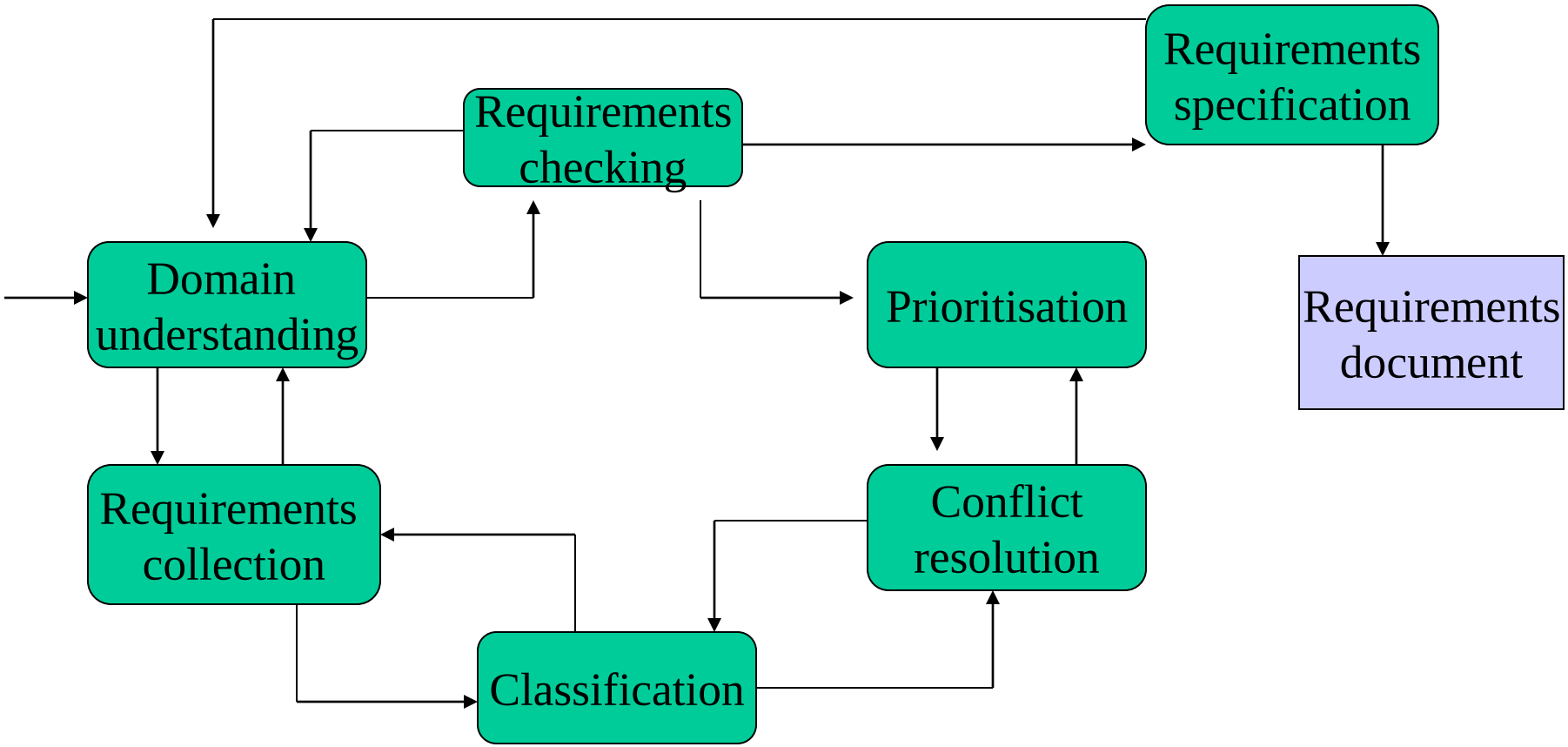
Domain requirements

Domain requirements are derived from the application domain of the system rather than from the specific needs of the system users. They may be new functional or non-functional requirements in their own right, constrain existing requirements or set out how particular computation must be carried out.

Examples

1. Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the users requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.
2. There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.

The requirements elicitation and analysis process



Requirement Analysis Tools

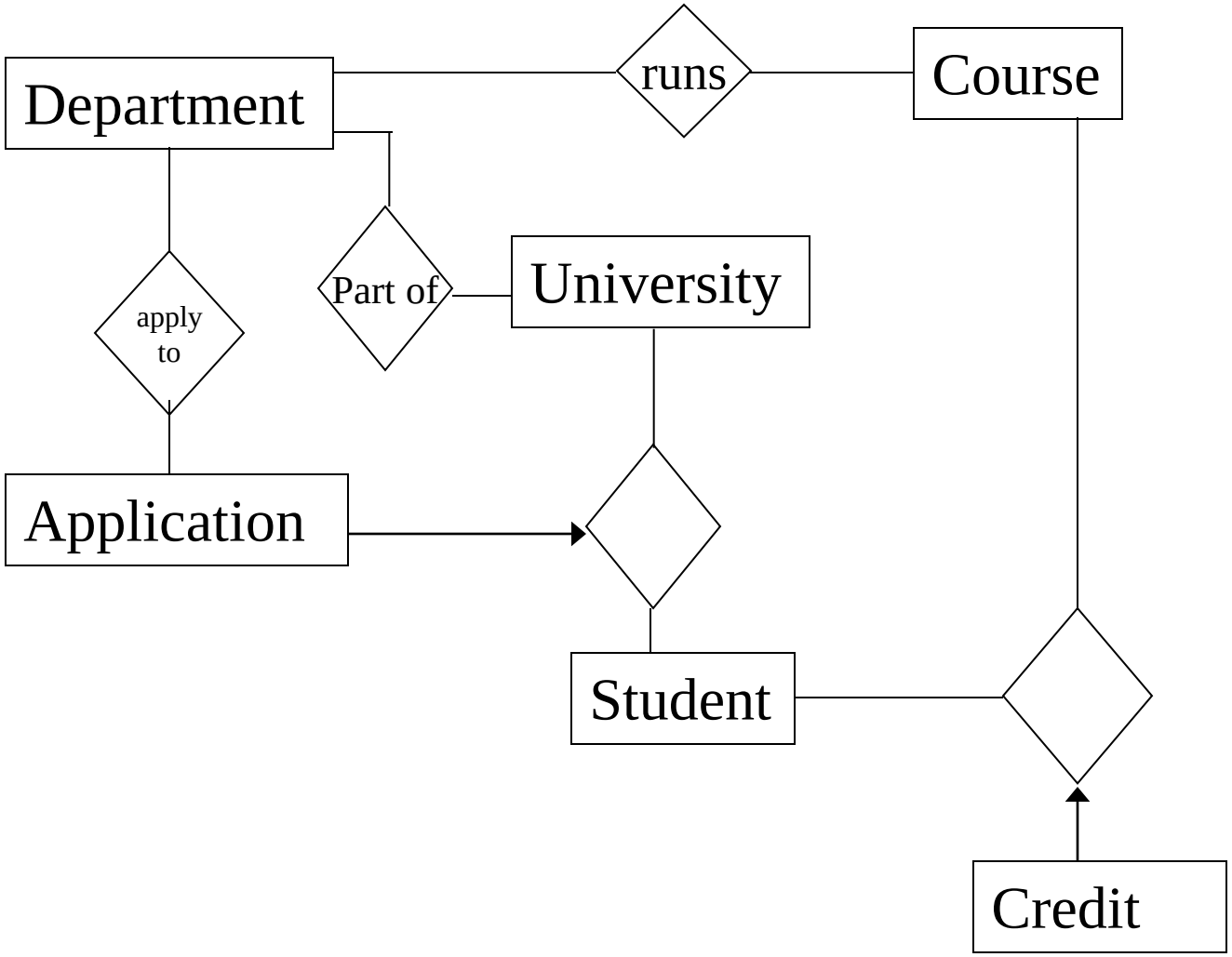
Traditional structured methods (P+D)

- Entity relationship diagrams (ERD)
- Data Flow Diagrams (DFDs)
- Entity State Transition diagrams

Object Oriented methods (O-O)

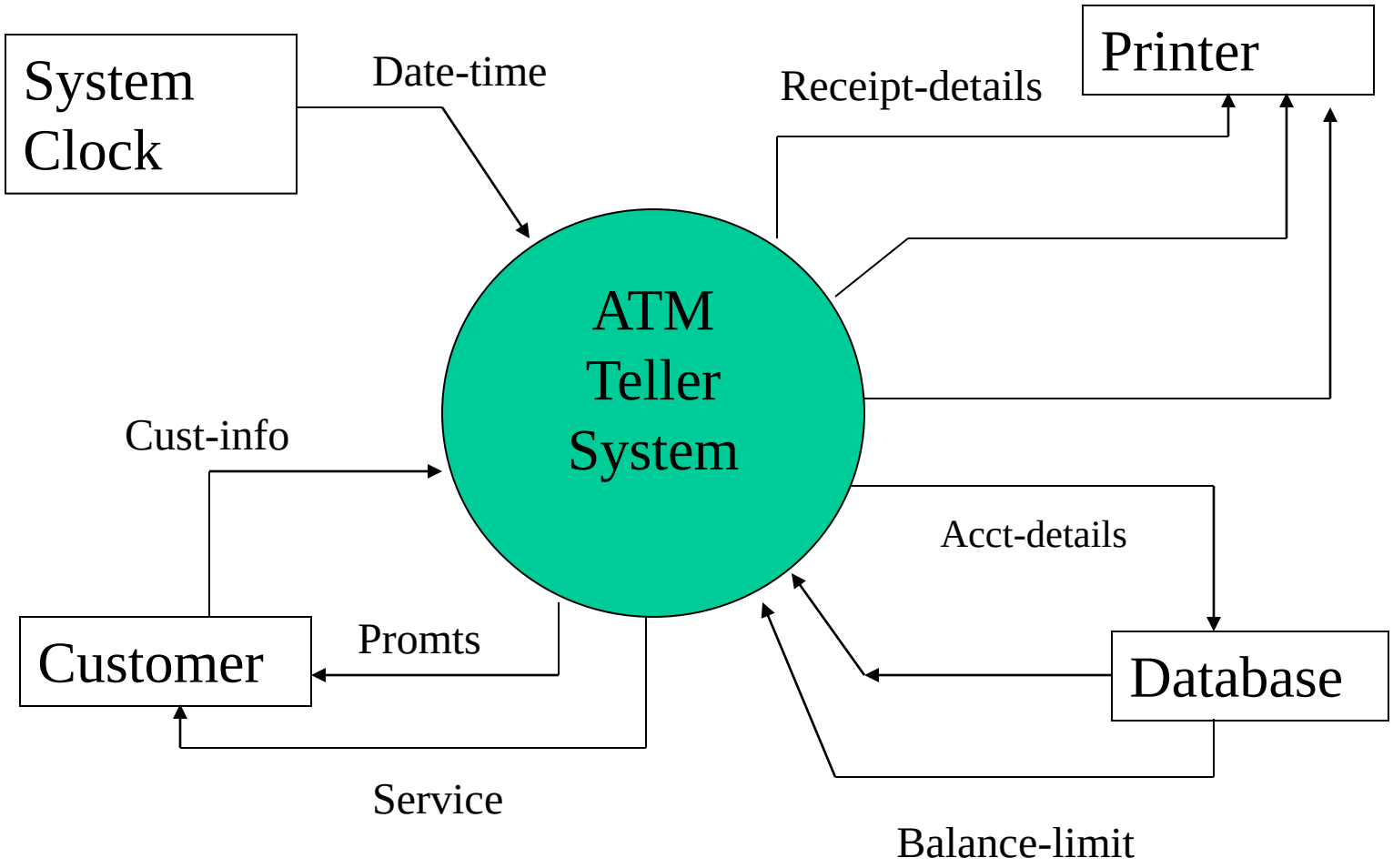
- Use Case Diagrams
- Class Diagrams
- Sequence Diagrams
- State Transition Diagrams

Entity Relationship Diagrams

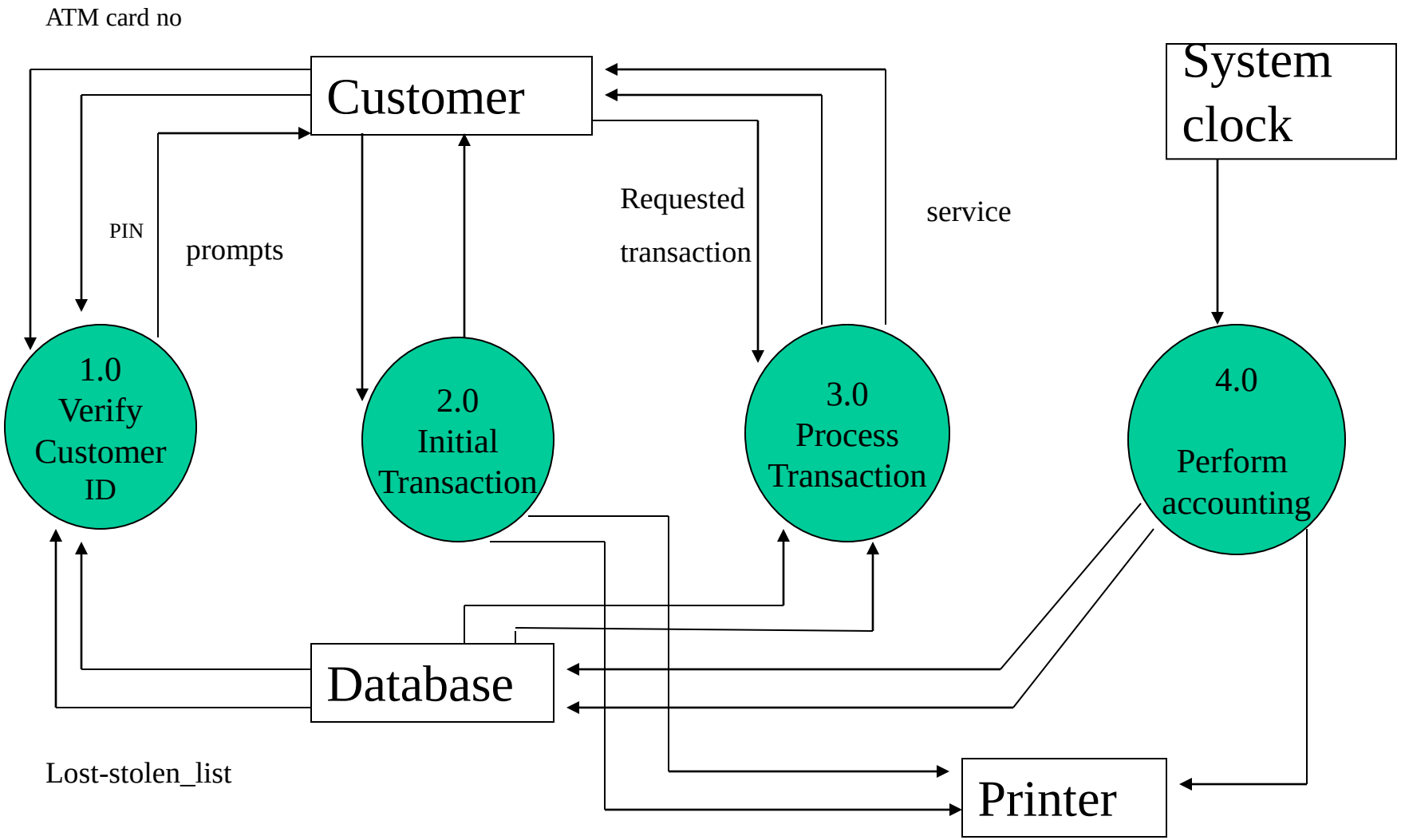


Data Flow Diagrams

Context Diagram



A First Level DFD



Process specification

Internal Specification

- Pseudo code specification

- Decision Tables

External specification

- Pre-condition-Post condition

Process specification - Pseudo code

Verify customer ID (DFD process 1.0)

```
Upon insertion of the customer's ATM card
  check the ATM-card-no against the lost-stolen-list
If the entered ATM-card-no is on the lost-stolen-list  Then
      Retain card and suspend processing
If the entered ATM-card-no is not on the lost-stolen-list Then
    Repeat
        Prompt customer for his PIN
        Compare the entered PIN to the PIN listed in the Bank's database for this customer
    Until
        the customer has entered his correct PIN  or
        fails to correctly enter his PIN after 3 attempts
If a valid PIN has been entered  Then
      Continue processing
Else
      Suspend processing
     Return ATM card
```

Process specification - Decision Table

Calculate new balances (DFD Process 4.1)

Condition	Decision Rule				
	1	2	3	4	5
Balance Inquiry	Y	N	N	N	N
Deposit	N	Y	N	N	N
Fund Transfer	N	N	Y	N	N
Loan payment	N	N	N	Y	N
Withdrawal	N	N	N	N	Y
ACTION					
Subtract amount from balance			X	X	X
Add amount to balance		X			
No Action	X				

Process specification - An External specification

- **pre-condition** – what must be true for the function to execute correctly
- **post-condition** - What is guaranteed to be true after the function executes
- **Example :** A merge function may be specified as follows

  merge(item1:array,item2 : array) : newArray

- **Pre**

  SizeOf (item1)>0, SizeOf(item2)>0, (both arrays have at least one element)

  item1[i] <item1[i+1], 1<i<noOfitems1 (in order)

  item2[i] <item2[i+1], 1<i<noOfitems2 (in order)

- **Post**

  SizeOf(newArray)=SizeOf(item1) + SizeOf(item2)

  newArray[i] < newArray[i+1], 1<i<SizeOf(newArray)

- These are all testable

Data Specification

All dataflows in DFDs should be specified using a combination of elementary dataflows or previously defined dataflows.

The following symbols can be used in dataflow specifications

and	+
or	[ ]
repeat	{ }
optional	( )
comment	* *

Data specification

Examples:

Acct-designation = 1{alphanumeric-character}15

Acct-number = 8{numeric-character}8

Amount = rupees+cents

Acct-selection = [savings/checking/loan-acct/credit-card]

(acct-designation)

Cust-acct-list = 1{acct-designation+acct-number}8

Balance-display = * display of current balance to customer*

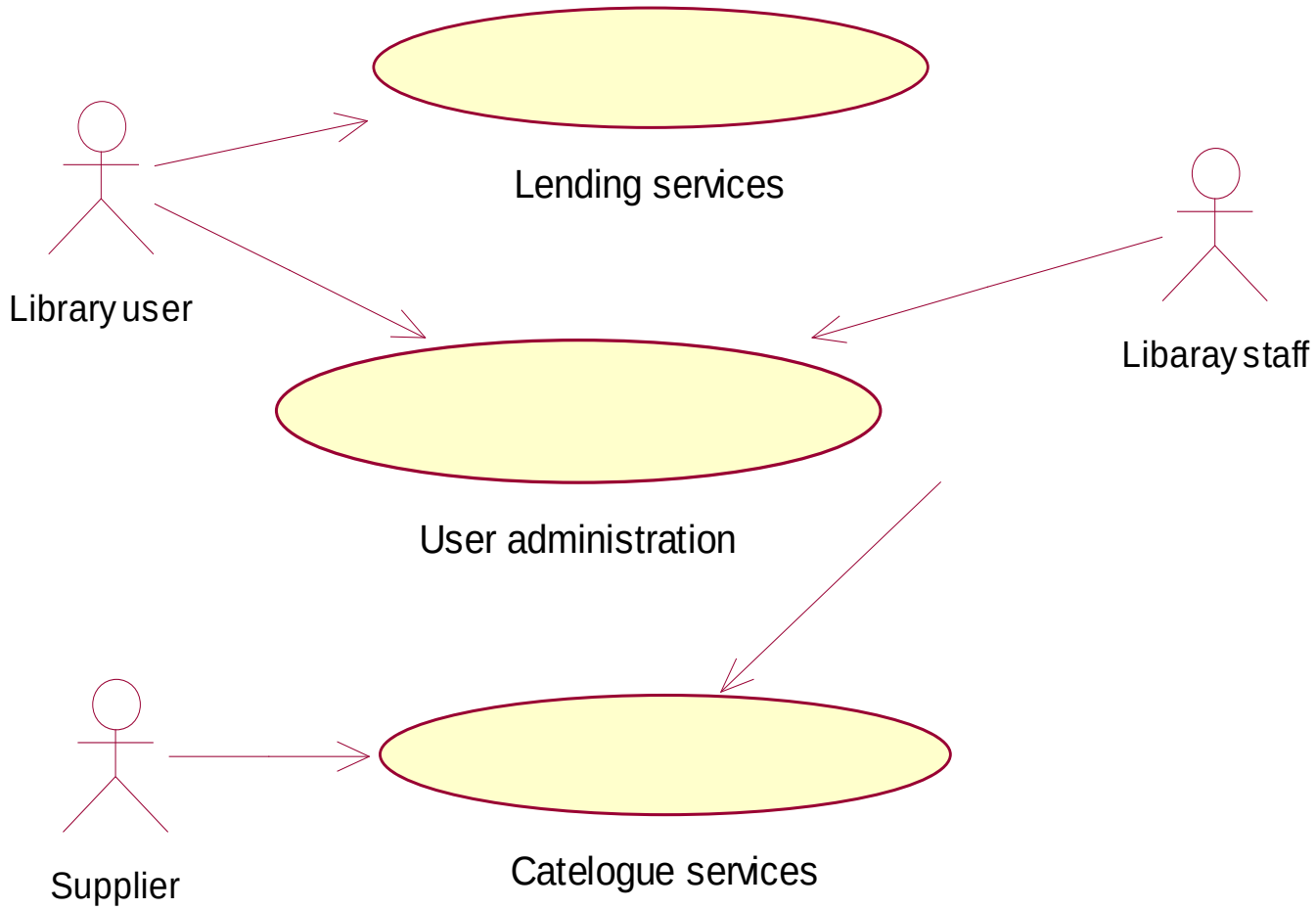
Requirement Specification practice– Good

Use a form based approach

* Describe each function in separate form

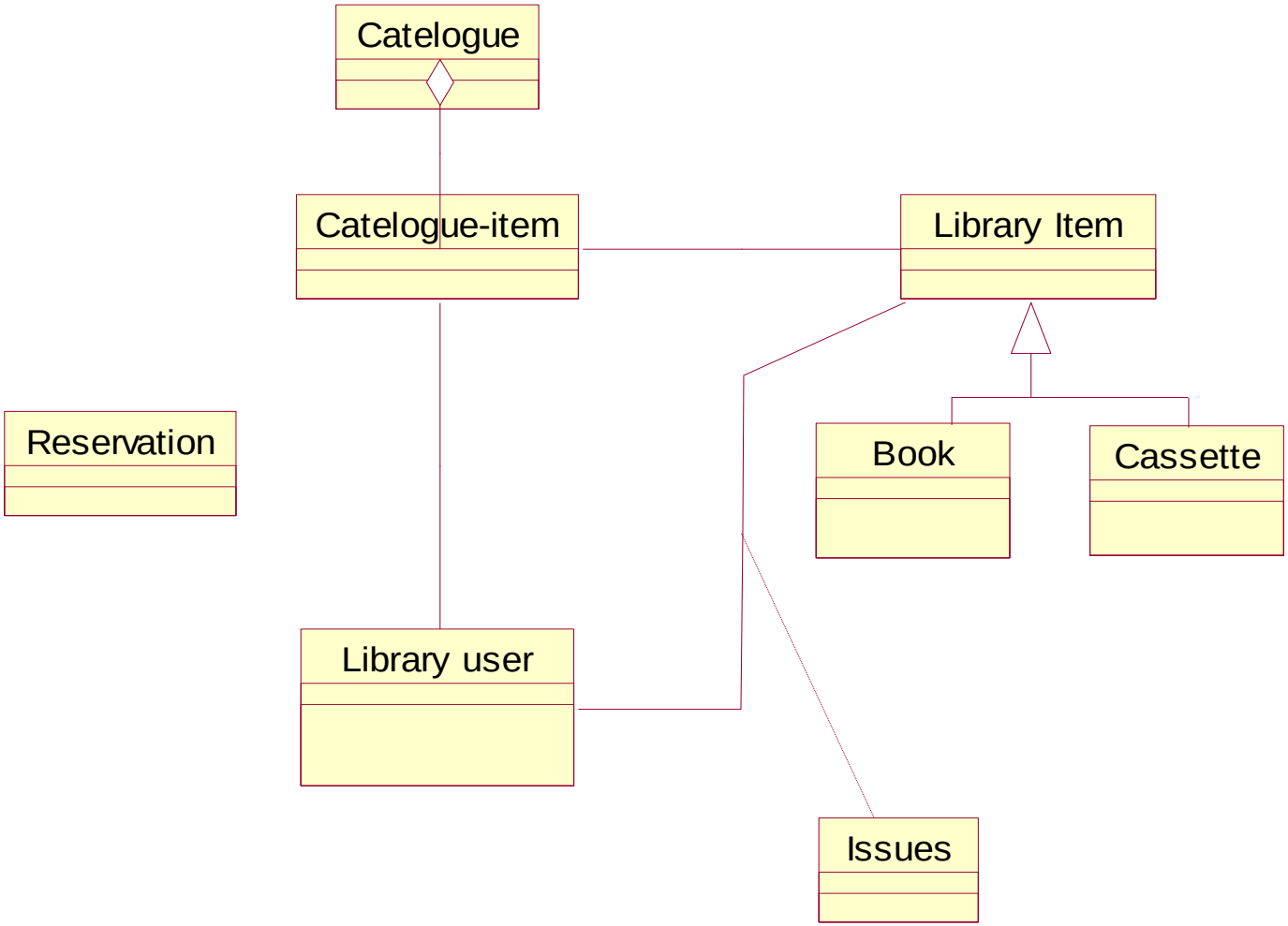
Function	:	obtain balance
Description	:	on request, returns the balance for a customer account
Inputs	:	Account number, PIN
Source	:	Keypad
Outputs	:	Account balance
Destination	:	Screen display
Requires	:	Customer has inserted card and entered PIN
Pre-condition	:	Card is from member bank
Post-condition	:	Balance is displayed
Side effects	:	None

A Use-case Diagram

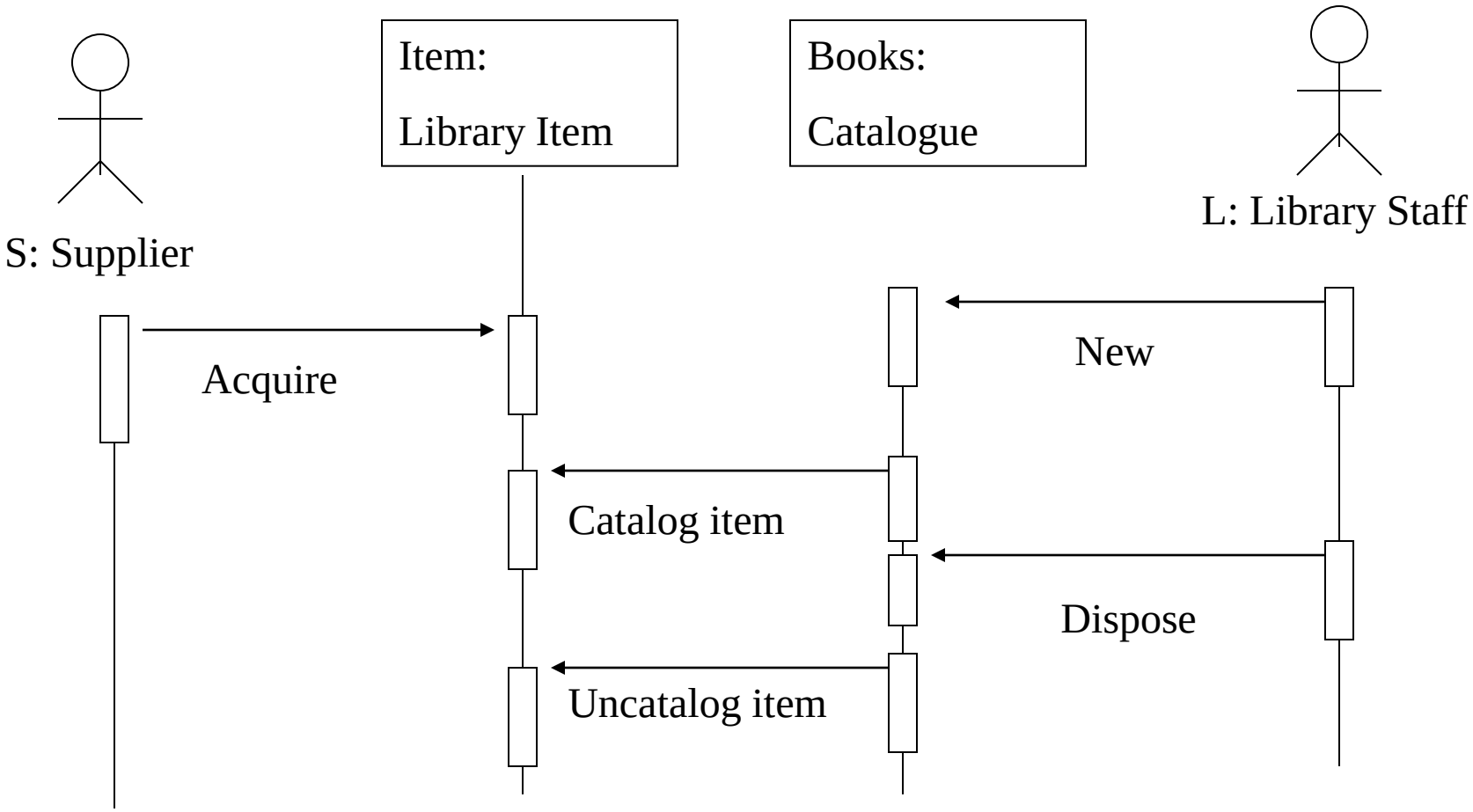


Use-cases are a scenario-based technique for requirements elicitation which were first introduced in the Objectory methods (Jacobson et at 1993).

A Class Diagram



Sequence Diagrams



Non-functional Requirements

- Define system properties and constraints Eg: reliability, performance, interface, security, storage requirements.

- Process requirements may also be specified mandating a particular CASE system, programming language or development method.

- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Requirements verifiability

- Requirements should be written so that they can be objectively verified.
- The problem with this requirement is its use of vague terms such as ‘ errors should be minimised’

  * *Experienced controllers should be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by an experienced users should not exceed two per day*

Non-functional Requirements measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes / No. of rows in code Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure (MTTF) Probability of unavailability Mean time between failures (MTBF) Mean time to recover (MTTR)
Robustness	Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements

Software Requirements Document - IEEE standard

1. Introduction

1.1 Purpose of the requirements document

1.2 Scope of the product

1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview of the remainder of the document

2. **General Description**

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 General constraints

2.5 Assumptions and dependencies

3. Specific requirements

All functional and non-functional requirements

System models (eg. DFD, ERD, Use-Case, Class, Sequence diagrams)

External Interfaces, Performance, database requirements, design constraints

Security, quality characteristics

4. **Appendices**

Use-Case Specification - Example

Use Case 17: Compute Rate of Investment.

- Primary Actor: System.
- Pre-Condition: User logged in & security specified
- Main Scenario :

1. System computes ROI for the security specified (Appendix A shows the detailed method for computing the ROI).

Use-Case Specification - Example

5.1 Appendix A:
The formula for calculation of ROI for shares:

- Suppose that, for the share of a particular company, following were the
- attributes of ith transaction.
- Amount of money transacted = mi
- Time between date of transaction and day when ROI is being calculated = ni

- Type of transaction = buy or sell
- Let sgn(i) = +1 if the transaction was a buy, and -1 if it was a sell.

- Then, the rate of investment, r, is got by solving the equation

- S sgn(i) mi (1+r) ni = M

- where M is the total amount in that share, and summation is over all the
- transactions.

- The formula for calculation of ROI for banks:

- For a banks the ROI is constant, namely the interest rate of that bank.

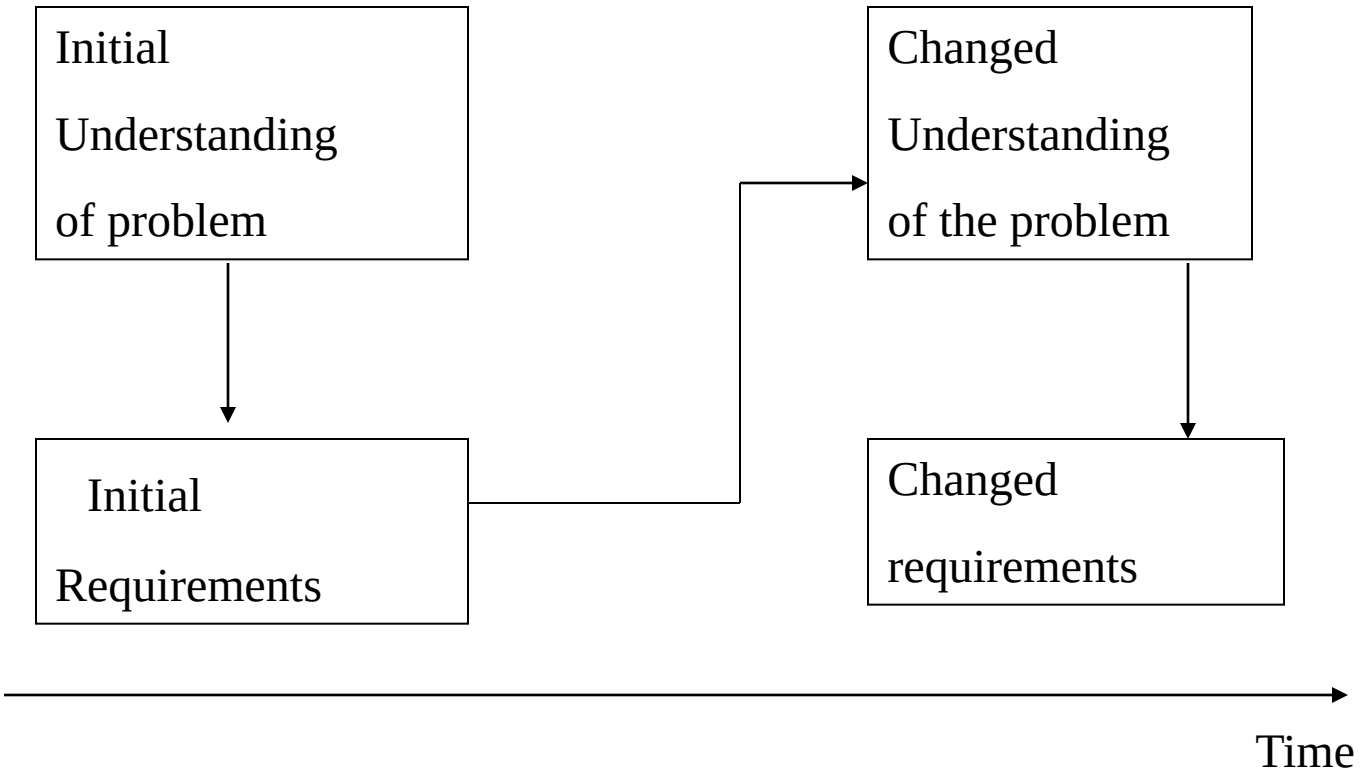
Requirement Evolution

Over the time, The systems environment and the business objectives will almost certainly changed. The requirements must therefore evolve to reflect this.

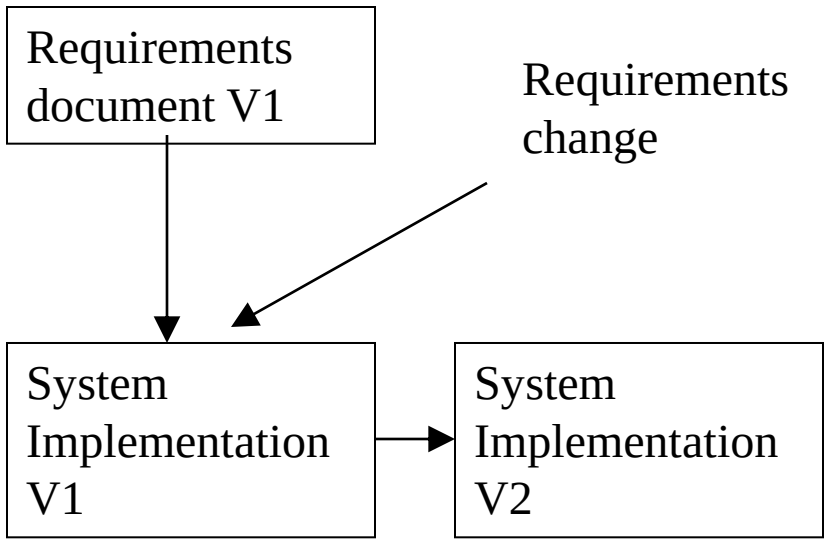
From an evolution perspective, requirements fall into two classes:

1. **Enduring requirements :** These are relatively stable requirements which derive from the core activity of the organisation and which relate directly to the domain of the system.
2. **Volatile requirements :** These are requirements which are likely to change during the system development or after the system has been

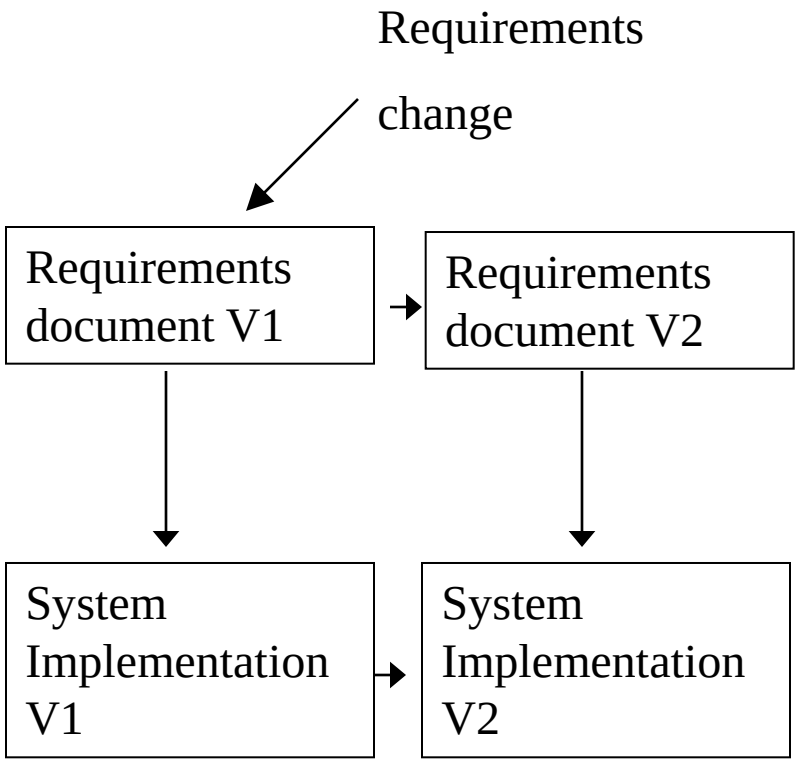
Requirements Evolution



Controlled Evolution of Requirements



Requirements and system inconsistent



Requirements and system consistent

Requirements Validation

- Requirements validation is concerned with showing that the requirements actually define the system which the customer wants.
- Requirements validation is important because errors in a requirements document can lead to extensive rework costs when they are subsequently discovered during development or after the system is in service.

Requirements Validation – Checks

- **Validity checks** - Systems have diverse users with different needs and any set of requirements is inevitably a compromise across the user community.

- **Consistency checks** - Requirements in a document should not conflict.

- **Completeness checks** – The requirements document should include requirements which define all functions and constraints intended by the system user.

- **Realism checks** - Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented.

- **Verifiability** - The requirements should be given in verifiable manner (eg; Using quantifiable measures) to reduce the potential for dispute between customer and developer.

Requirements Validation - Techniques

■**Requirements Reviews** - The requirements are analalysed systematically by a team of reviewers.

■ **Prototyping** - In this approach to validation, an executable model of the system is demonstrated to end-users and customers. They can check whether the requirements satisfy their needs.

■ **Test-case generation** - Requirements should ideally be testable. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsider.

■ **Automated consistency analysis** : If the requirements are expressed as a system model in a structured or formal notation than CASE toolsmay be used to check the consistency of the model.

Review Questions

1. Complete the following table

Document	**Contains**	**Who for?**
Requirement Definition(user)		
Requirements specification		
Software Requirements Specification (SRS)		

2. Give two examples of non-functional requirements

(i)	
(ii)	

3. Identify software metrics for the following non-functional requirements

Reliability	
Size	
Ease of Use	
Performance	

3. A search function returns an array index and may be specified as follows:

Search (X: Intarray, Key : Integer) : Integer;

Write appropriate pre- and post- conditions.

Pre-condition	
Post-Conditions	