Software Requirements

- Definition: **Description and Specifications of a system**
- Topics covered:
  - Functional and Non-functional requirement
  - User Requirements
  - System requirements
  - The software requirements document

Software Requirements

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed
- Requirements may be functional or non-functional
  - Functional requirements describe system services or functions
  - Non-functional requirements is a constraint on the system or on the development process

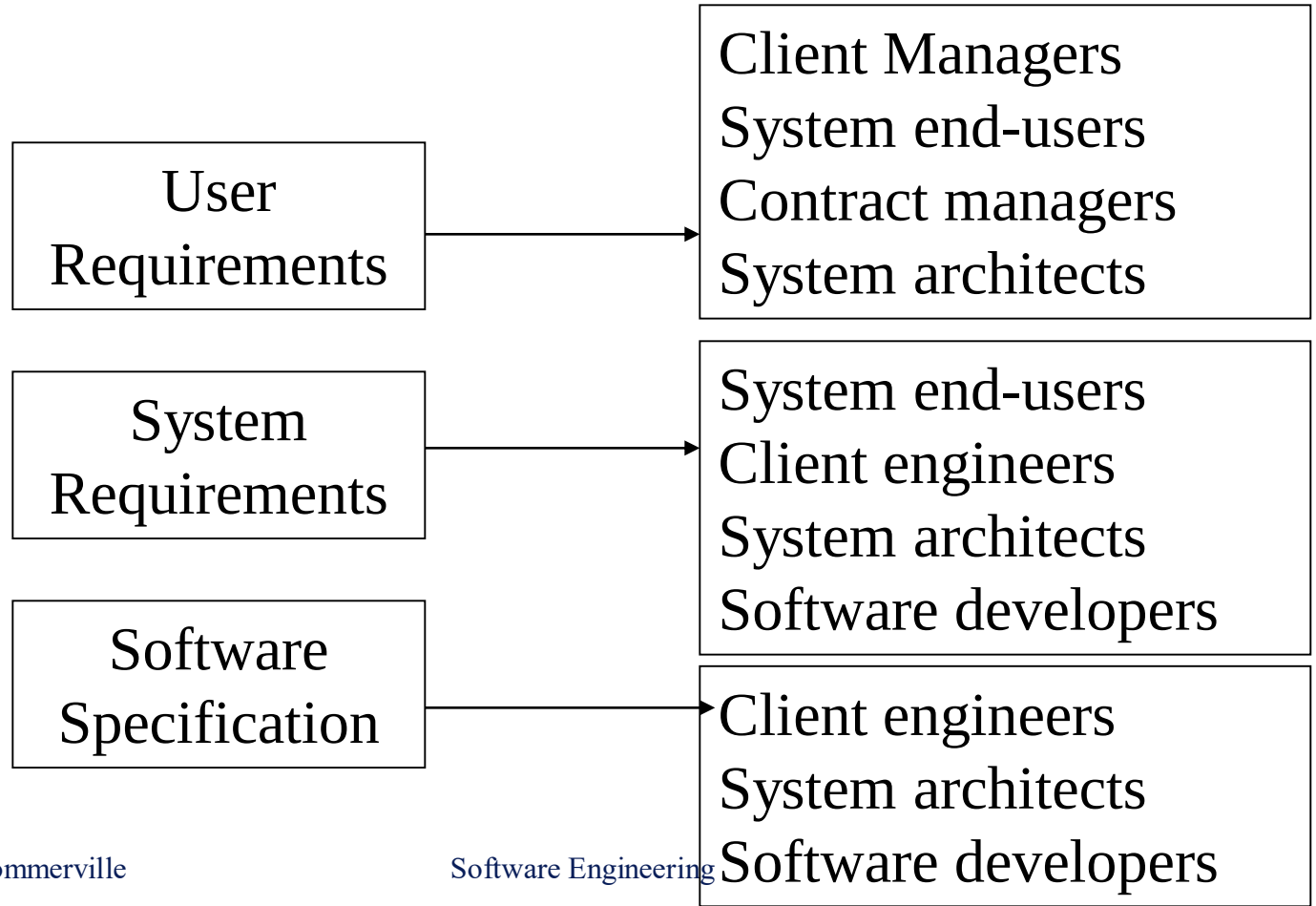
What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail
  - Both these statements may be called requirements

Types of requirements

- User requirements
  - Statements in natural language (NL) plus diagrams of the services the system provides and its operational constraints. Written for customers
- System requirements
  - A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor
- Software specification
  - A detailed software description which can serve as a basis for a design or implementation. Written for developers

Requirements Targets



Requirements Types:

1. Functional requirements: services the system should provide
2. Non-functional requirements: constraints on the services of functions offered by the system. e.g. speed, time to market
3. Domain requirements: related to the application domain of the system (may be functional or non-functional requirements)

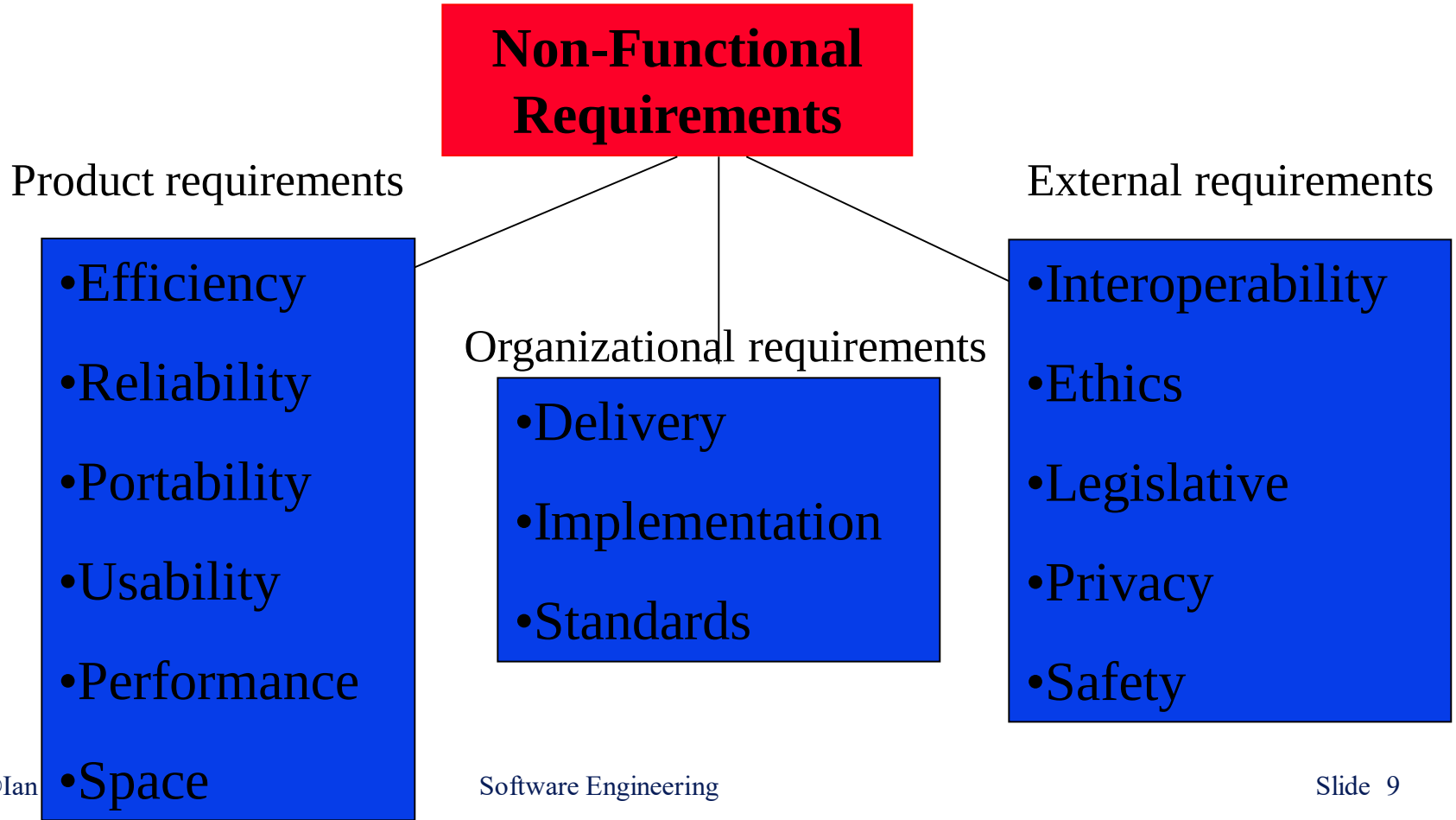
Functional requirements

- Functionality or services that the system is expected to provide.
- Functional requirements may also explicitly state what the system shouldn't do.
- Functional requirements specification should be:
  - Complete: All services required by the user should be defined
  - Consistent: should not have contradictory definition (also avoid ambiguity→ don't leave room for different interpretations)

Non-Functional requirements

- Requirements that are not directly concerned with the specific functions delivered by the system
- Typically relate to the system as a whole rather than the individual system features
- Often could be deciding factor on the survival of the system (e.g. reliability, cost, response time)

Non-Functional requirements classifications:



Domain requirements

- Domain requirements are derived from the application domain of the system rather than from the specific needs of the system users.
- May be new functional requirements, constrain existing requirements or set out how particular computation must take place.
- Example: tolerance level of landing gear on an aircraft (different on dirt, asphalt, water), or what happens to fiber optics line in case of sever weather during winter Olympics (Only domain-area experts know)

Problems with natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up
- Requirements amalgamation
  - Several different requirements may be expressed together
- Ambiguity
  - The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult
- Over-flexibility
  - The same thing may be said in a number of different ways in the specification

Alternatives to NL specification

- Structured Natural language (via standard forms & templates)
- Program Description Language (PDL)
- Use-Cases (scenario-based technique)
- Mathematical specification (notations based on mathematical concepts such as finite-state machines or set.)

Structured language specifications

- A limited form of natural language may be used to express requirements
- This removes some of the problems resulting from ambiguity and flexibility and imposes a degree of uniformity on a specification
- Often best supported using a form-based approach

Form-based specification

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function: Add node

Description: Adds a node to an existing design.

Inputs: Node type, Node Position

Outputs: Design identifier

Pre/Post conditions:

Other attributes:

Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1

PDL-based requirements definition

- Requirements may be defined operationally using a language like a programming language but with more flexibility of expression
- Most appropriate in two situations
  - Where an operation is specified as a sequence of actions and the order is important
  - When hardware and software interfaces have to be specified
  - Example: ATM machine

PDL disadvantages

- PDL may not be sufficiently expressive to express the system functionality in an understandable way
- Notation is only understandable to people with programming language knowledge
- The requirement may be taken as a design specification rather than a model to help understand the system

ATM Specification: a PDL example

```
Class ATM {
        // declaration here
        public static void main (string args[]) InvalidCard {
                try {
                thisCard.read();  //may throw Invalid card
    exception
                pin = KeyPaD.READpIN(); attempts = 1;
                While (!thisCard.pin.equal(pin) & attempts < 4)
                        pin = KeyPad.readPin(); attempts += 1;
                .
                .
                .
```

The requirements document

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Requirements Engineering (RE) processes

- **Processes used to discover, analyse and validate system requirements**
- RE vary widely depending on the application domain, the people involved and the organization developing the requirements
- However, there are a number of generic activities common to all processes
  - Requirements elicitation
  - Requirements analysis
  - Requirements validation
  - Requirements management

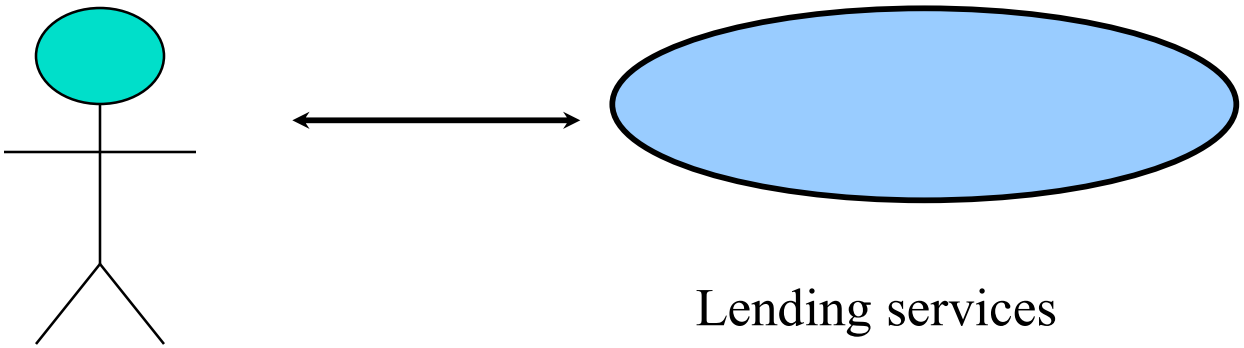
Problems of requirements analysis

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders may have conflicting requirements
- Organizational and political factors may influence the system requirements
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change

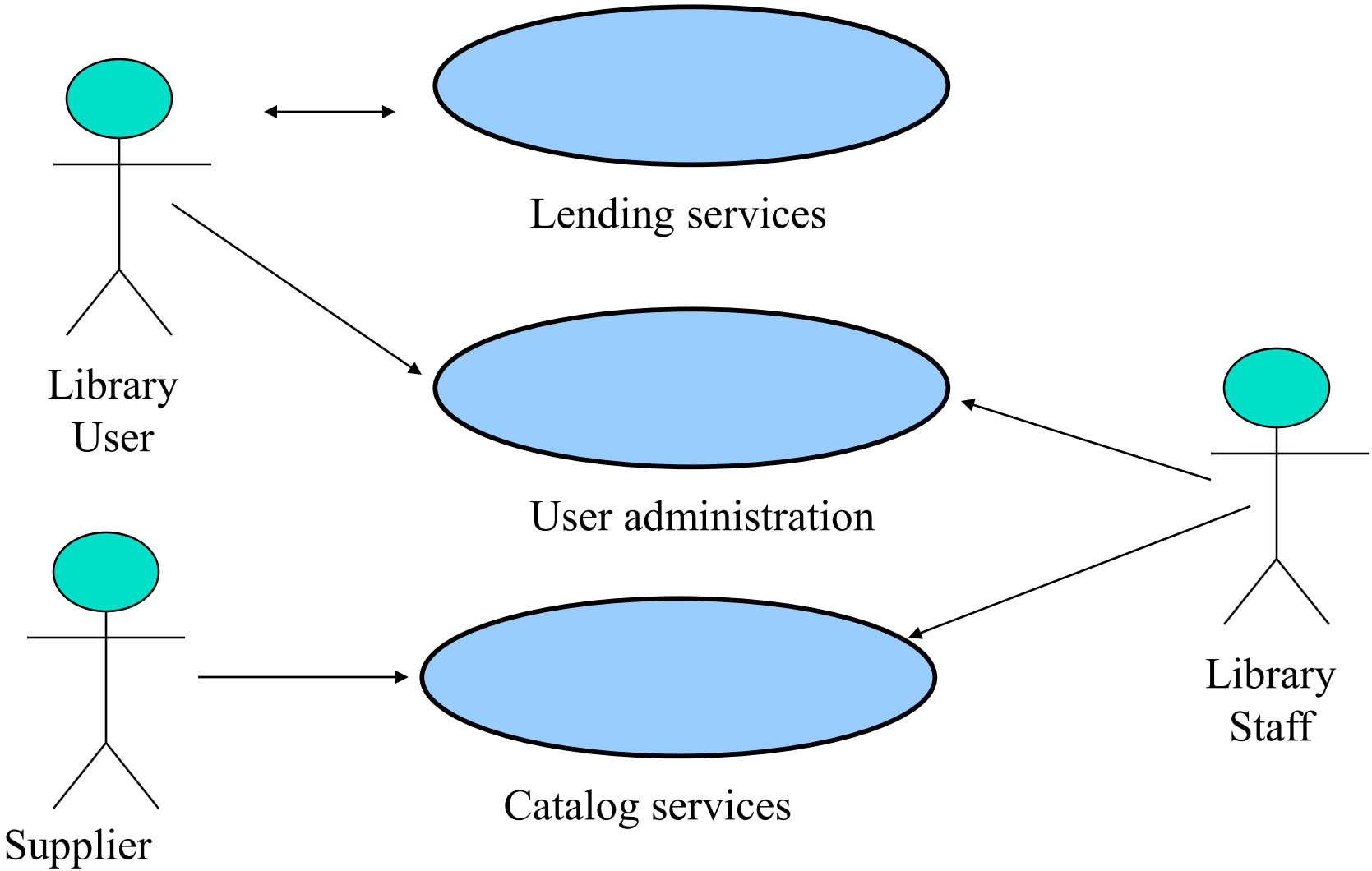
Use cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself
- A set of use cases should describe all possible interactions with the system
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system

Lending use-case



Library use-cases



Ethnography

- Ethnography is an observational technique that can be used to understand social and organizational requirements.
- Developed in a project studying the air traffic control process
- Problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant

Enduring and volatile requirements

- Enduring requirements. Stable requirements derived from the core activity of the customer organisation. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
- Volatile requirements. Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

Classification of requirements

- Mutable requirements
  - Requirements that change due to the system's environment
- Emergent requirements
  - Requirements that emerge as understanding of the system develops
- Consequential requirements
  - Requirements that result from the introduction of the computer system
- Compatibility requirements
  - Requirements that depend on other systems or organisational processes