ICT 2402 Software Engineering

Software Requirements

Objectives

After completing this lesson you should be able to

- Discuss the concepts of user and system requirements
- Describe functional and non-functional requirements
- Explain how software requirements may be organised in a requirements document

Topics covered

- Functional and non-functional requirements
- User requirements
- System requirements
- Interface specification
- The software requirements document

Requirements vs. Requirements engineering

- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed or the process of finding out, analyzing, documenting and validating requirements is Requirements engineering.

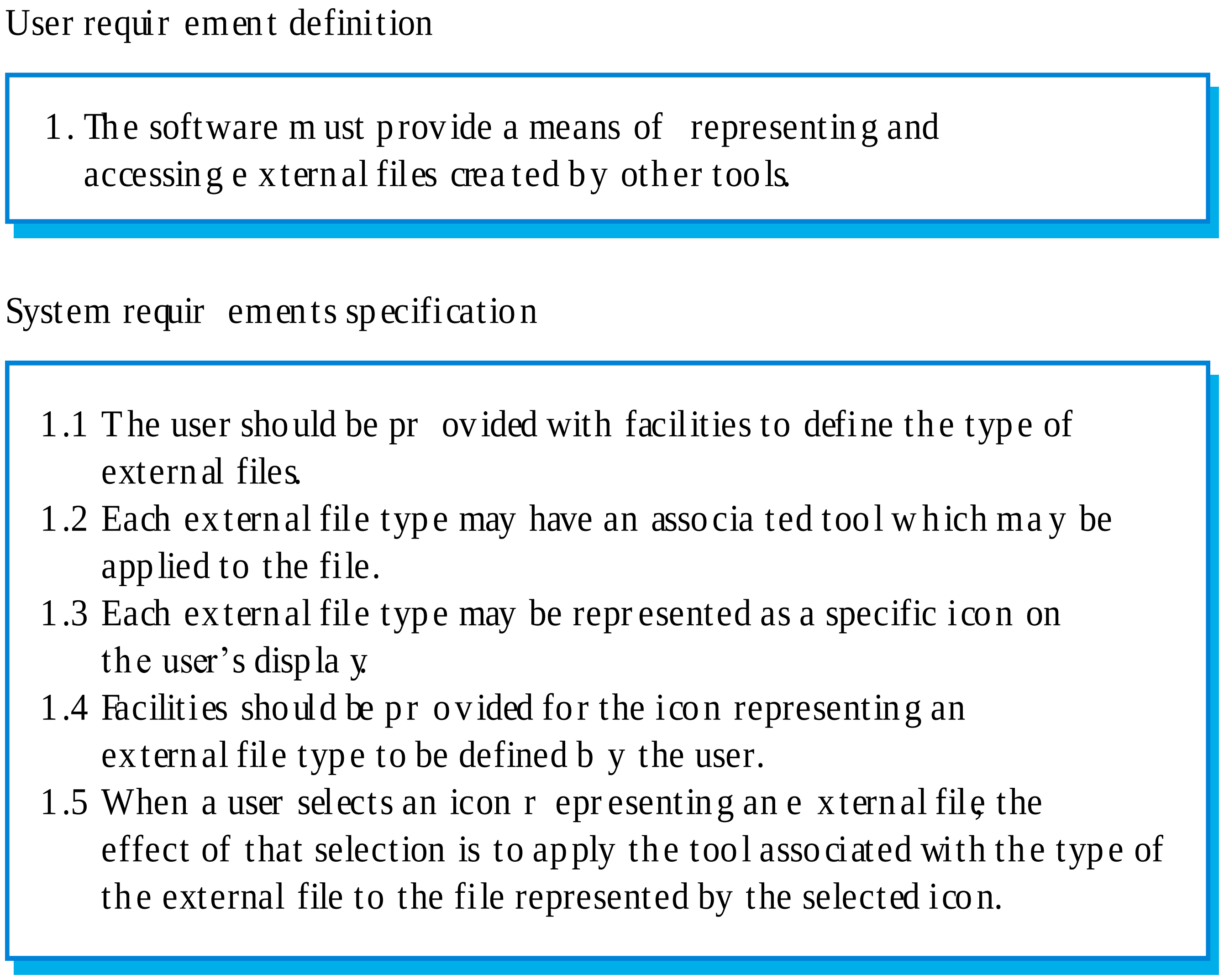
Requirements abstraction (Davis)

"If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the *requirements document* for the system."

Types of requirement

- User requirements
  - High-level abstract requirements
- System requirements
  - Detail description of what the system should do

Definitions and specifications



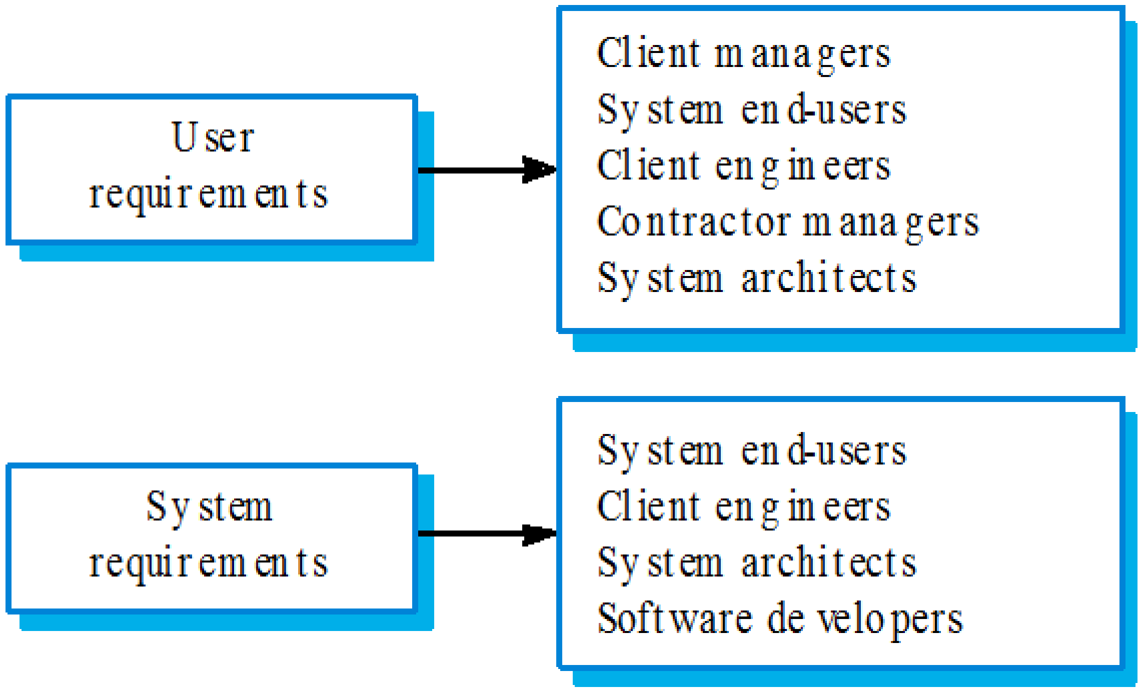
User requirement definition

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 tool which may be applied to the file.
1.3 Each external file type may be represented as 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.
1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Requirements readers



Functional and non-functional requirements

- Functional requirements
  - 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
  - constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc
- Domain requirements
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain

Functional requirements

- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail

Cont...

Normally, functional requirements includes,

- Descriptions of data to be entered into the system
- Descriptions of operations performed by each screen
- Descriptions of work-flows performed by the system
- Descriptions of system reports or other outputs
- Who can enter the data into the system.
- How the system meets applicable regulatory requirements

The LIBSYS system

- A library system that provides a single interface to a number of databases of articles in different libraries
- Users can search for, download and print these articles for personal study

Examples of functional requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it
- The system shall provide appropriate viewers for the user to read documents in the document store
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area

Sample functional requirements for a HR system

- System should allocate an id for each employee
- System should track salary details for each employee
- Personal leave details should be recorded
- Employee training details should be tracked
- System should keep track on EPF and tax reductions from the employees
- At a certain time, HR manager should be able to view available and already used leave for each employee

Requirements imprecision

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways by developers and users
- Consider the term ‘appropriate viewers’
  - User intention - special purpose viewer for each different document type
  - Developer interpretation - Provide a text viewer that shows the contents of the document

Requirements completeness and consistency

- Requirements should be both complete and consistent
- Completeness
  - They should include descriptions of all facilities required
- Consistency
  - There should be no conflicts or contradictions in the descriptions of the system facilities
- In practice, it is impossible to produce a complete and consistent requirements document

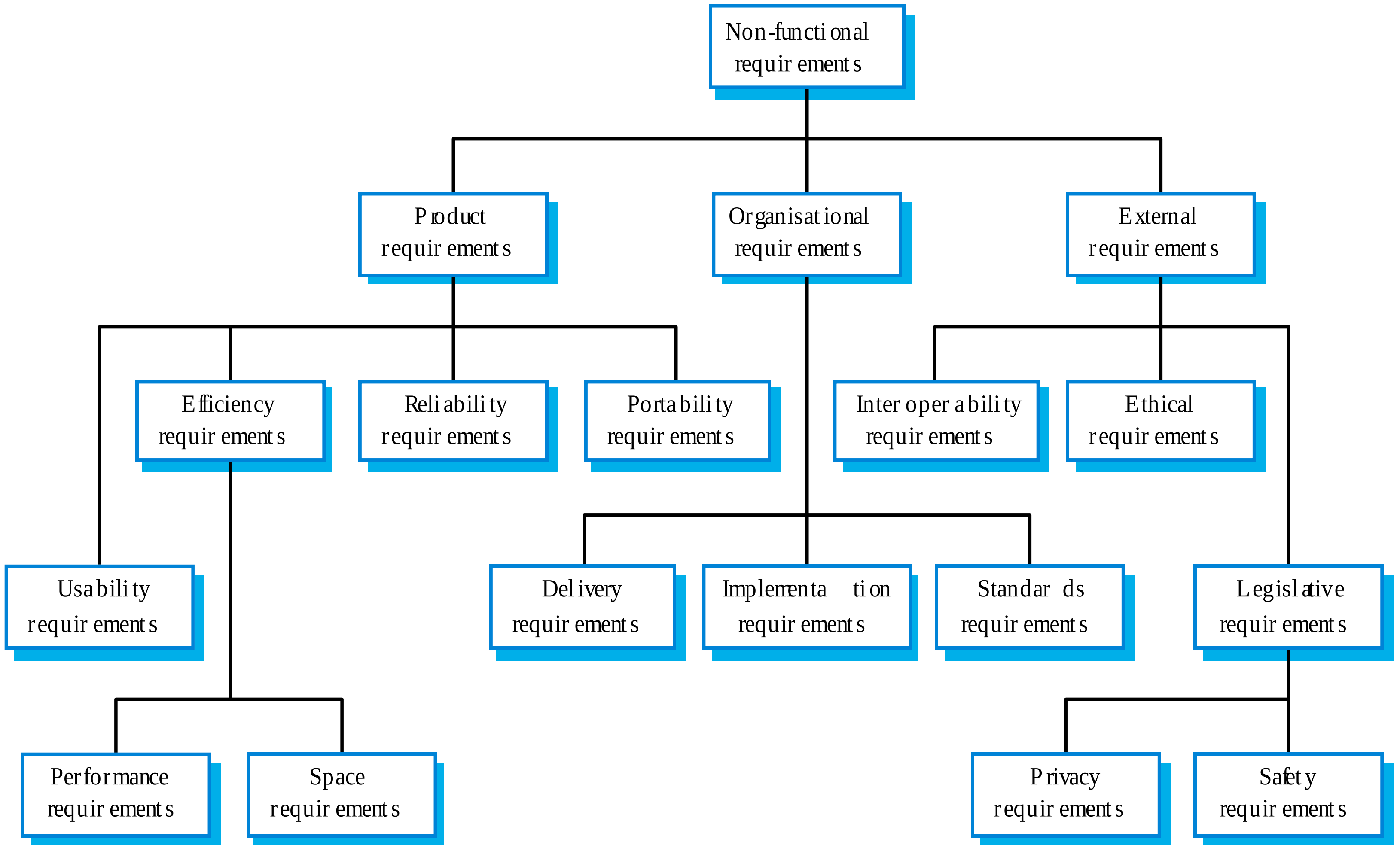
Non-functional requirements

- Define system properties and constraints e.g. reliability, response time and storage requirements.
- Process requirements may also mandate the use of a particular CASE system, programming language or development method
- More critical than functional requirements. If they are not met, the system is useless

Non-functional classifications

- Product requirements
  - Specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Outcome of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirement types



Non-functional requirements examples

- Product requirement

  8.1 The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.
- Organisational requirement

  9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
- External requirement

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

Requirements measures

Property	**Measure**
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Domain requirements

- Derived from the application domain
- Describes system characteristics and features that reflect the domain
- May be new functional requirements, constraints on existing requirements or define specific computations
- If not satisfied, the system may be unworkable

Library system domain requirements

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

Domain requirements problems

- Understandability
  - Requirements are expressed in the language of the application domain (e.g. mathematical equation);
  - Not understood by software engineers developing the system
- Implicitness
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit

User requirements

- Does not include design characteristics
- These are requirements that the user 'wants' or would 'ask for‘
- User Requirements are those that can be observed from outside the system
- Should describe functional and non-functional requirements in such a way that they are understandable by system users who don’t have detailed technical knowledge
- User requirements are defined using natural language, tables and diagrams as these can be understood by all users
- Example user requirement for a learning management system(LMS):

    LMS should facilitate the distribution of e-course materials for the students who are subscribed to the relevant courses.

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

LIBSYS requirement

4..5 LIBSYS shall provide a financial accounting system that maintains records of all payments made by users of the system. System managers may configure this system so that regular users may receive discounted rates.

Requirement problems

- Database requirements includes both conceptual and detailed information
  - Describes the concept of a financial accounting system that is to be included in LIBSYS
  - However, it also includes the detail that managers can configure this system - this is unnecessary at this level

Guidelines for writing requirements

- Invent a standard format and use it for all requirements
- Use language consistently
  - Use shall for mandatory requirements
  - Should for desirable requirements
- Use text highlighting to identify key parts of the requirement
- Avoid the use of computer jargon

System requirements

- More detailed specifications of system functions, services and constraints than user requirements
- Intended to be a basis for designing the system
- May be incorporated into the development contract
- May be defined or illustrated using system models
- Example system requirements for the learning management system (LMS)
  - Upon visiting the site students should be asked for a user name and password
  - After validation, students should be presented with the course materials for the registered courses
  - Lecturers should be able to upload course materials through a special admin interface

Problems with NL specification

- 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
- Lack of modularisation
  - NL structures are inadequate to structure system requirements

Alternatives to NL specification

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used .
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don’t understand formal specifications and are reluctant to accept it as a system contract.

Structured language specifications

- The freedom of the requirements writer is limited by a predefined template for requirements
- All requirements are written in a standard way
- The terminology used in the description may be limited
- The advantage is that the most of the expressiveness of natural language is maintained but a degree of uniformity is imposed on the specification

Form-based specification

Insulin Pump/Control Software/SRS/3.3.2

Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered
Destination	Main control loop
Action:	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin..
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side-effects	None

Tabular specification

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2-r1)<(r1-r0))	CompDose = 0

Sugar level increasing and rate of increase stable or increasing. ((r2-r1) □ (r1-r0))	CompDose = round ((r2-r1)/4)
	If rounded result = 0 then
	CompDose = MinimumDose

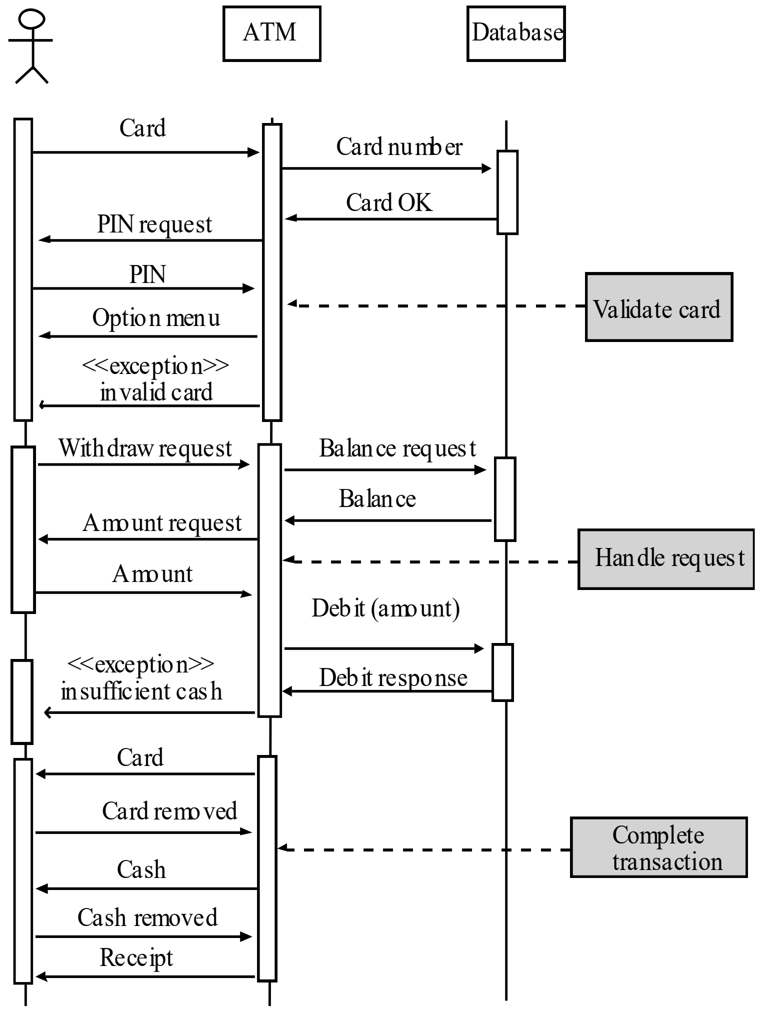
Graphical models

- Most useful when you need to show how state changes or
- To describe a sequence of actions

Sequence diagrams

- These show the sequence of events that take place during some user interaction with a system.
- You read them from top to bottom to see the order of the actions that take place.
- Cash withdrawal from an ATM
  - Validate card;
  - Handle request;
  - Complete transaction.

Sequence diagram of ATM withdrawal



Interface specification

```
interface PrintServer {

// defines an abstract printer server
// requires:          interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

        void initialize ( Printer p ) ;
        void print ( Printer p, PrintDoc d ) ;
        void displayPrintQueue ( Printer p ) ;
        void cancelPrintJob (Printer p, PrintDoc d) ;
        void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

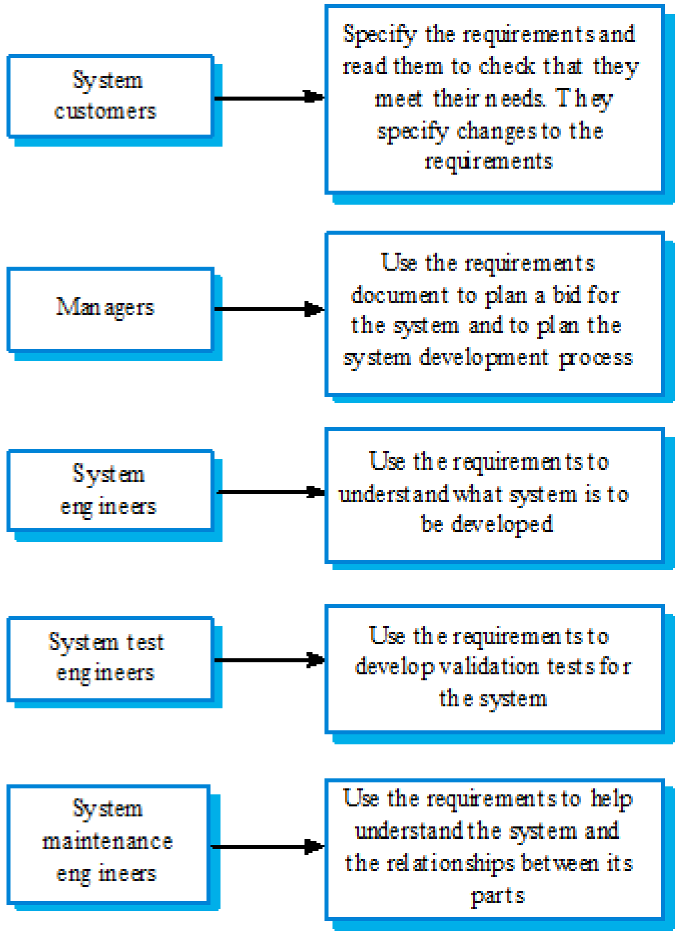
The requirements document

- Official statement of what is required of the system developers
- Should include both a definition of user requirements and a specification of the system 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

Software requirements specification (SRS)

- SRS is the official document which includes details of requirements
- SRS has multiple users including customers, managers, developers, testers and maintenance engineers
- Therefore, SRS should be written in such a way which appeals to all those stakeholders

Users of a requirements document


System customers
Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements
Managers
Use the requirements document to plan a bid for the system and to plan the system development process
System engineers
Use the requirements to understand what system is to be developed
System test engineers
Use the requirements to develop validation tests for the system
System maintenance engineers
Use the requirements to help understand the system and the relationships between its parts

IEEE requirements standard

- Defines a generic structure for a requirements document that must be instantiated for each specific system
  - Introduction
  - General description
  - Specific requirements
  - Appendices
  - Index

Requirements document structure

- Preface
- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Key points

- Requirements set out what the system should do and define constraints on its operation and implementation
- Functional requirements set out services the system should provide
- Non-functional requirements constrain the system being developed or the development process
- User requirements are high-level statements of what the system should do and they should be written using natural language, tables and diagrams

Key points

- System requirements are intended to communicate the functions that the system should provide
- A software requirements document is an agreed statement of the system requirements
- The IEEE standard is a useful starting point for defining more detailed specific requirements standards

References

- Chapter 6 of “Software Engineering”, 9/e by Ian Sommerville.
- www.software-engin.com

Questions?