Requirements and Specification

Requirements Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed

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

Requirements Engineering

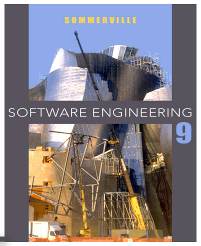
- The hardest single part of building a software system is deciding what to build
  - Cripples the process if done wrong
  - Costly to rectify later

???

- The goal of requirement engineering is to determine (pick one):
  - What software client **wants?**
  - What software client **needs?**

"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."

Software specification (or requirements engineering)



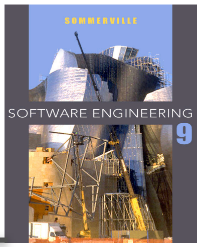
✧ The process of establishing what services are required and the constraints on the system's operation and development.

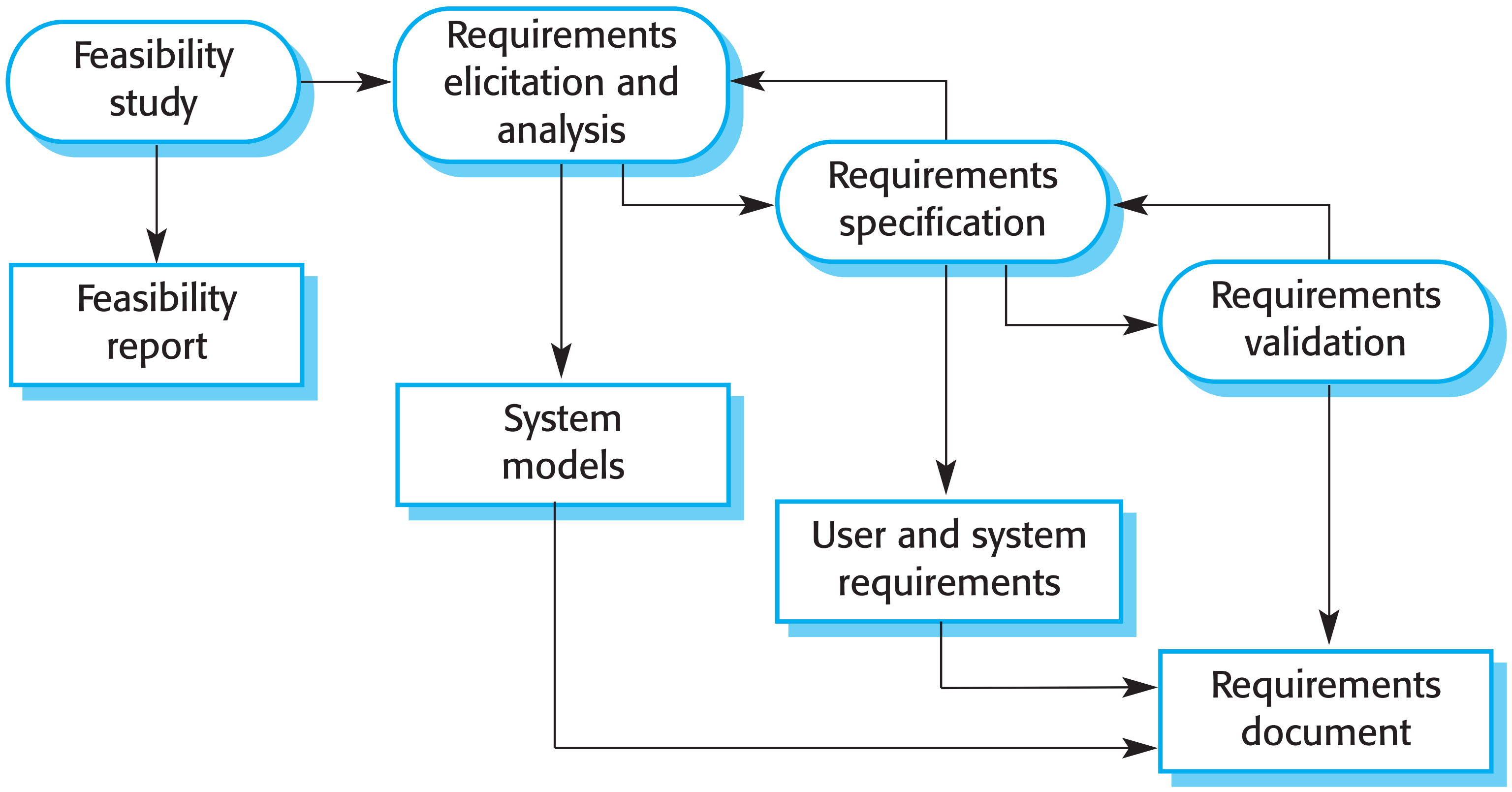
✧ Requirements engineering process

- Feasibility study
  - Is it technically and financially feasible to build the system?
- Requirements elicitation and analysis
  - What do the system stakeholders require or expect from the system?
- Requirements specification
  - Defining the requirements in detail
- Requirements validation
  - Checking the validity of the requirements

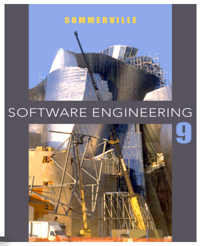
Sommerville: Ch 2.2.1, p36ff

The requirements engineering process





Outcome



- An agreed requirements document that specifies a system satisfying stakeholder requirements.
- Two level of details

  End-users and customers need a high-level statement of the requirements.

  System developers need a more detailed system specification.

User requirements

- Should describe 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.

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.

Determining Stakeholders and Needs

- Must determine stakeholders
  - Anyone who benefits from the system developed
  - E.g., who's client and who's user ?
- Try to understand what their needs are
- Reconcile different needs/points of view

Techniques for Requirement Gathering

- Interviewing
- User stories
- Straw man documents
- Prototypes

Interviewing

- One path is obvious
  - Sit down with client/user and ask questions
  - Listen to what they say, and what they don't say
- A less obvious path
  - Master-apprentice relationship
  - Have them teach you what they do
  - Go to workplace and watch them do the task
- In all types of interviews, get details
  - Ask for copies of reports, logs, emails on process
  - These may support, fill in, or contradict what the user said

Extreme Programming – User Stories

- Recall: client writes user stories
  - Using client vocabulary
- Describe usage scenarios of software
  - Title, short description
- Each user story has acceptance tests
  - Clarify the story
  - Will tell you when the customer thinks story is done

Disadvantages of Talking

- Interviews are useful, but

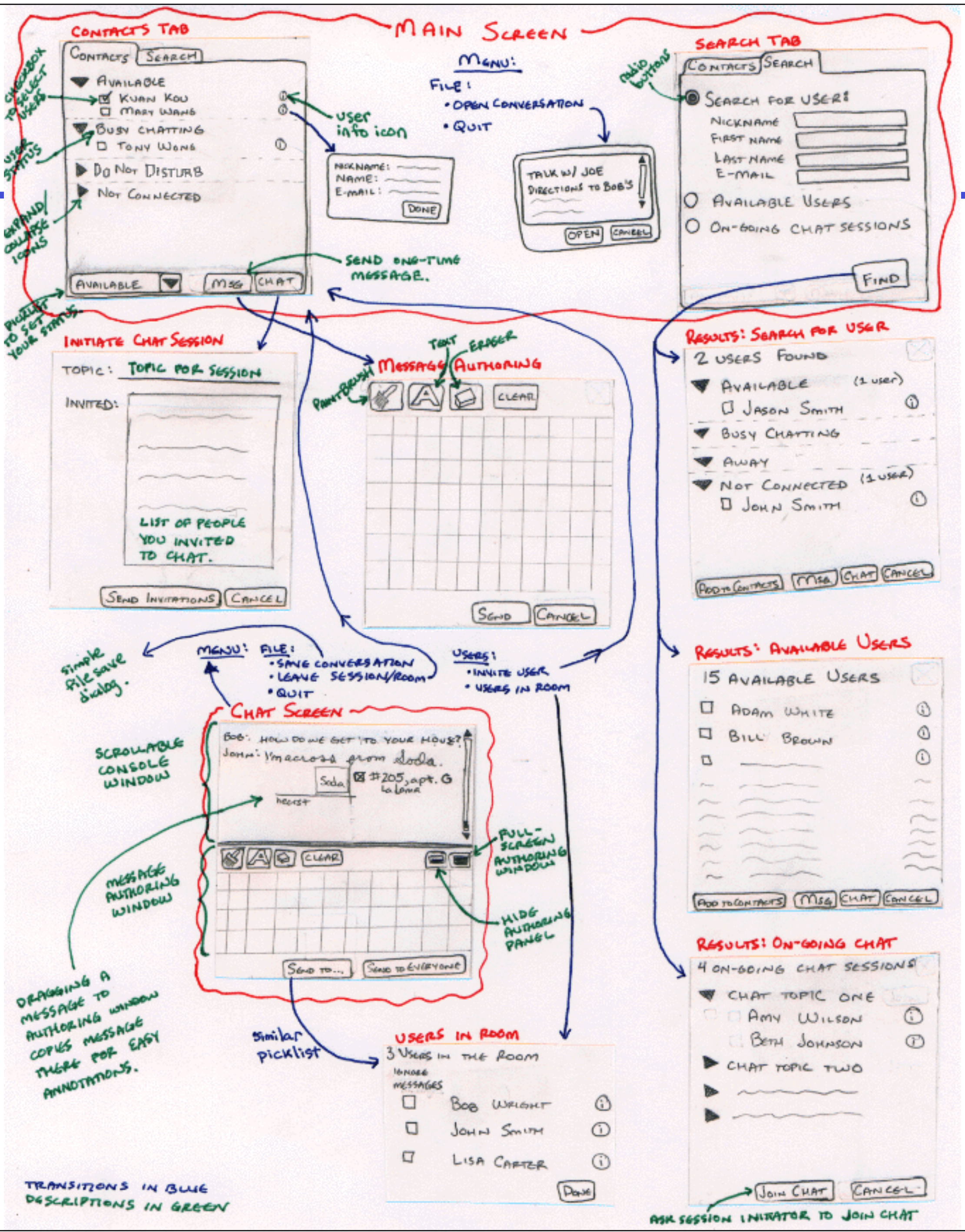
  "I know you believe you understood what you think I said, but I am not sure you realize that what you heard is not what I meant!"

- Users/clients may
  - Not have the vocabulary to tell you what they need
  - Not know enough about computer science to understand what is possible
    - Or impossible
  - Sometimes may lead to restricted functionality

- Good idea to gather requirements in other ways, too

Strawman

- Sketch the product for the user/client
  - Storyboards
  - Flowcharts
  - HTML mock-ups
  - Illustrate major events/interfaces/actions
- Anything to convey ideas without writing code!


MAIN SCREEN
CONTACTS TAB
CONTACTS
SEARCH
AVAILABLE
KUAN KOU
MARY WANG
BUSY CHATTING
TONY WONG
DO NOT DISTURB
NOT CONNECTED
AVAILABLE
MSG
CHAT
CHECKBOX TO SELECT USERS
USER STATUS
EXPAND/COLLAPSE ICONS
PICKLIST TO SET YOUR STATUS.
user info icon
NICKNAME:
NAME:
E-MAIL:
DONE
SEND ONE-TIME MESSAGE.
MENU:
FILE:
• OPEN CONVERSATION
• QUIT
TALK W/ JOE
DIRECTIONS TO BOB'S
OPEN
CANCEL
SEARCH TAB
radio buttons
SEARCH FOR USER:
NICKNAME
FIRST NAME
LAST NAME
E-MAIL
AVAILABLE USERS
ON-GOING CHAT SESSIONS
FIND
INITIATE CHAT SESSION
TOPIC: TOPIC FOR SESSION
INVITED:
LIST OF PEOPLE YOU INVITED TO CHAT.
SEND INVITATIONS
CANCEL
MESSAGE AUTHORING
PAINTBRUSH
TEXT
ERASER
CLEAR
SEND
CANCEL
RESULTS: SEARCH FOR USER
2 USERS FOUND
AVAILABLE (1 user)
JASON SMITH
BUSY CHATTING
AWAY
NOT CONNECTED (1 USER)
JOHN SMITH
ADD TO CONTACTS
MSG
CHAT
CANCEL
simple file save dialog.
MENU: FILE:
• SAVE CONVERSATION
• LEAVE SESSION/ROOM
• QUIT
USERS:
• INVITE USER
• USERS IN ROOM
RESULTS: AVAILABLE USERS
15 AVAILABLE USERS
ADAM WHITE
BILL BROWN
ADD TO CONTACTS
MSG
CHAT
CANCEL
CHAT SCREEN
BOB: HOW DO WE GET TO YOUR HOUSE?
JOHN: I'm across from Soda.
Soda
#205, apt. G
La Loma
Hearst
SCROLLABLE CONSOLE WINDOW
MESSAGE AUTHORING WINDOW
CLEAR
FULL-SCREEN AUTHORING WINDOW
HIDE AUTHORING PANEL
SEND TO...
SEND TO EVERYONE
DRAGGING A MESSAGE TO AUTHORING WINDOW COPIES MESSAGE THERE FOR EASY ANNOTATIONS.
similar picklist
USERS IN ROOM
3 USERS IN THE ROOM
IGNORE MESSAGES
BOB WRIGHT
JOHN SMITH
LISA CARTER
DONE
RESULTS: ON-GOING CHAT
4 ON-GOING CHAT SESSIONS
CHAT TOPIC ONE
AMY WILSON
BETH JOHNSON
CHAT TOPIC TWO
JOIN CHAT
CANCEL
ASK SESSION INITIATOR TO JOIN CHAT
TRANSITIONS IN BLUE
DESCRIPTIONS IN GREEN

Rapid Prototyping

- Write a prototype
  - Major functionality, superficially implemented
  - Falls down on moderate-to-extreme examples
    - No investment in scaling, error handling, etc.

- Show prototype to users/clients
  - Users have a real system – more reliable feedback
  - Refine requirements
  - But, significant investment

Pitfalls of Rapid Prototyping

- Needs to be done quickly
  - Remember, this is just the requirements phase!
  - Danger of spending too long refining prototype
- The prototype becomes the product
  - Prototype deliberately not thoroughly thought-out
  - Product will inherit the sub-optimal architecture
- Prototype serves as the spec
  - Prototype is incomplete, maybe even contradictory
- When done well, extremely useful

Summary of Requirements Gathering

- Find out what users/clients need
  - Not necessarily what they say they want

- Use
  - Interviews
  - User stories
  - Straw man documents
  - Rapid prototyping

  - *As* appropriate . . .

Requirement vs. Specification

- User Requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System Specifications
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.
- The distinction is often glossed over
  - Sommerville sees this as two levels of detail in the requirements document, the system requirements are a "functional specification" Ch 4, p83.

Specifications

- Describe the functionality of the product
  - Precisely
  - Covering all the circumstances
- Move from the finite to the infinite
  - Finite examples (requirements) to infinite set of possible computations
  - This is not easy

Specifications: theory & practice

- In principle, specifications should be:
  - Unambiguous:
    - Only one way to interpret the spec
  - Complete
    - Include descriptions of all facilities required.
  - Consistent
    - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is almost impossible to produce a complete and consistent requirements document.

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 include 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.

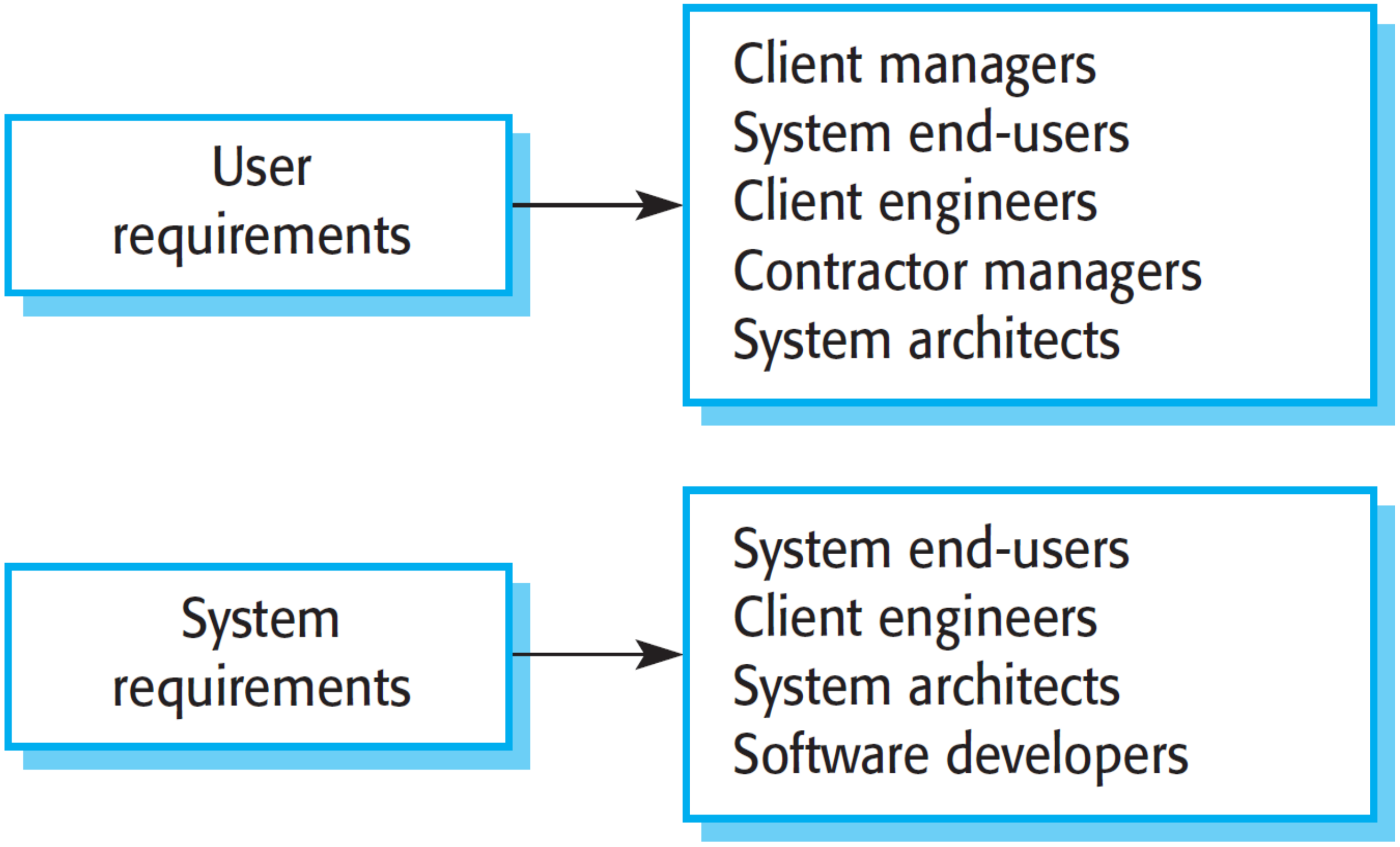
Different Views of Specifications

- Developer's
  - Specification must be detailed enough to be implementable
  - Unambiguous
  - Self-consistent

- Client's/user's
  - Specifications must be comprehensible
  - Usually means: not too technical

- Legal
  - Specification can be a contract
  - Should include acceptance criteria
    - If the software passes tests X, Y, and Z, it will be accepted

Requirements readers



Informal Specifications

- Written in natural language
  - E.g., English

- Example

  "If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%"

Problems with Informal Specs

- Informal specs of any size inevitably suffer from serious problems
  - Omissions
    - Something missing
  - Ambiguities
    - Something open to multiple interpretations
  - Contradictions
    - Spec says “do A” and “do not do A”
  - Amalgamation
    - Different requirements mixed together

These problems will be faithfully implemented in the software unless found in the spec

Informal Specifications Revisited

“If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%”

Informal Specifications Revisited

"If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%"

January: target $100K, actual $64K

February: target $120K, actual $100K

March: target $100K, actual $95,100

Comments on Informal Specification

- Informal specification is universally reviled
  - By academics
  - By “how to” authors
- Informal specification is also widely practiced
  - Why?

Why Do People Use Informal Specs?

- The common language is natural language
  - Customers can't read formal specs
  - Neither can most programmers
  - Or most managers / lawyers
  - A least-common denominator effect takes hold

- Truly formal specs are very time-consuming
  - And hard to understand
  - And overkill for most projects

Semi-Formal Specs

- Best current practice is "semi-formal" specs
  - Allows more precision than natural language where desired

- Usually a boxes-and-arrows notation
  - Must pay attention to:
  - What boxes mean
  - What arrows mean
  - Different in different systems!
- We'll see one example (UML) next time

Functional and non-functional

- 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

- Functional user requirements may be high-level statements of what the system should do but functional system specifications should describe the system services in detail.

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: 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.

Example of Requirements imprecision

- 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.

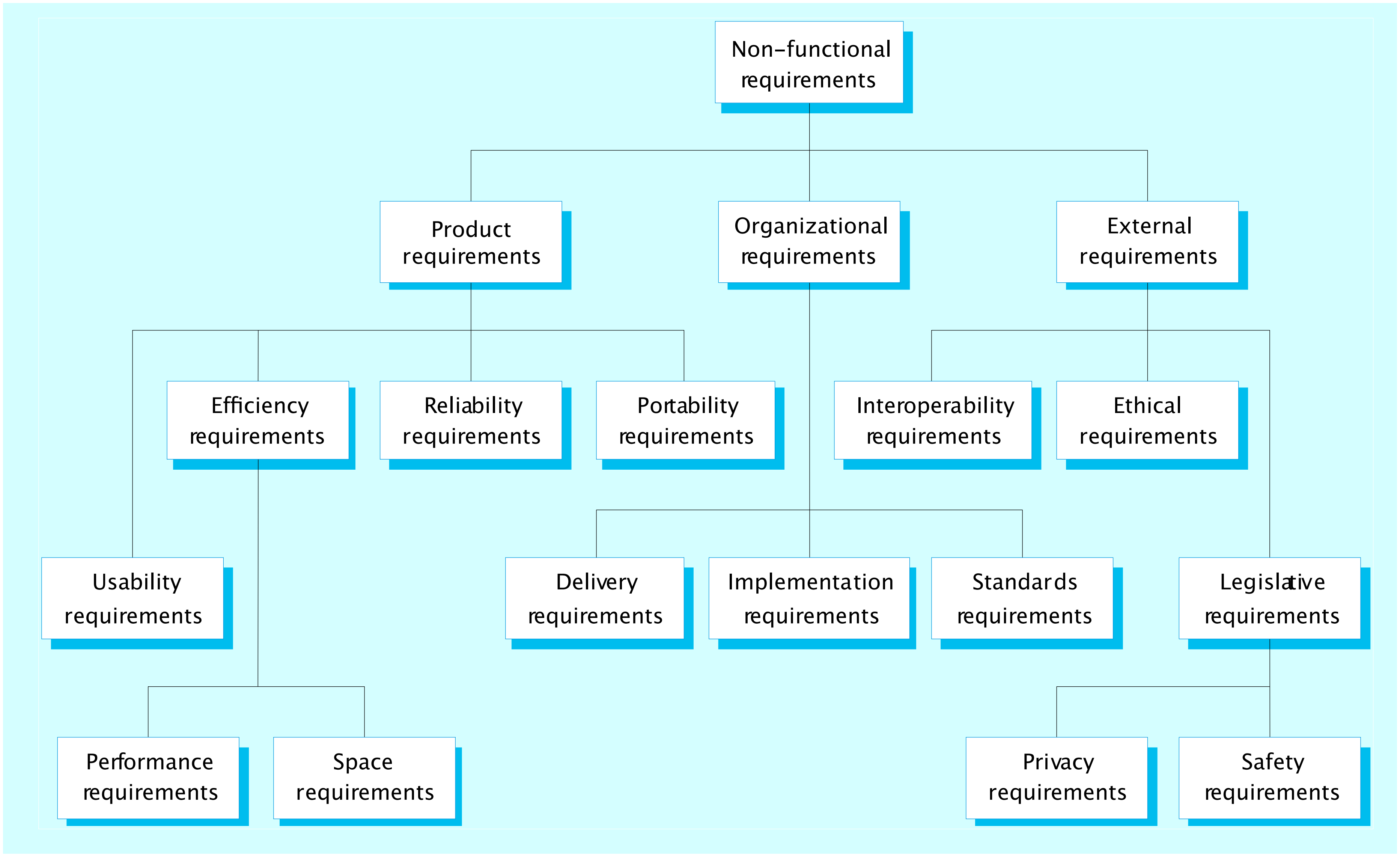
Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular process, programming language, or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organizational requirements
  - Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which 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.

- Organizational 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.

Goals and requirements

- Non-functional requirements/specifications may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as “ease of use”.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

Examples

- **A system goal**
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized
- **A verifiable non-functional requirement**
  - Experienced controllers shall 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 experienced users shall not exceed two per day.

Requirements measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K bytes Number of RAM 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

Requirements interaction

- Conflicts between different non-functional requirements are common in complex systems
- Spacecraft system
  - To minimize weight, the number of separate chips in the system should be minimized.
  - To minimize power consumption, lower power chips should be used.
  - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

(Application) Domain requirements

- Derived from the application domain
- Describe system characteristics and features that reflect the domain.
- Domain requirements
  - new functional requirements,
  - constraints on existing requirements,
  - define specific computations.
- If domain requirements are 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.

Train protection system

- The deceleration of the train shall be computed as:
  - $D_{train} = D_{control} + D_{gradient}$

  where $D_{gradient}$ is 9.81ms^2 * compensated gradient/alpha and where the values of 9.81ms^2 /alpha are known for different types of train.

Domain requirements problems

- Understandability
  - Requirements are expressed in the language of the application domain;
  - This is often 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.

Guidelines for writing requirements

- Invent a “standard” format and use it for all requirements.
- Use language in a consistent way. Use shall or must for mandatory requirements, should for desirable requirements
- Use text **highlighting** to identify key parts of the requirement
- Avoid the use of computer jargon
- See Reading on the Schedule

Requirements and design

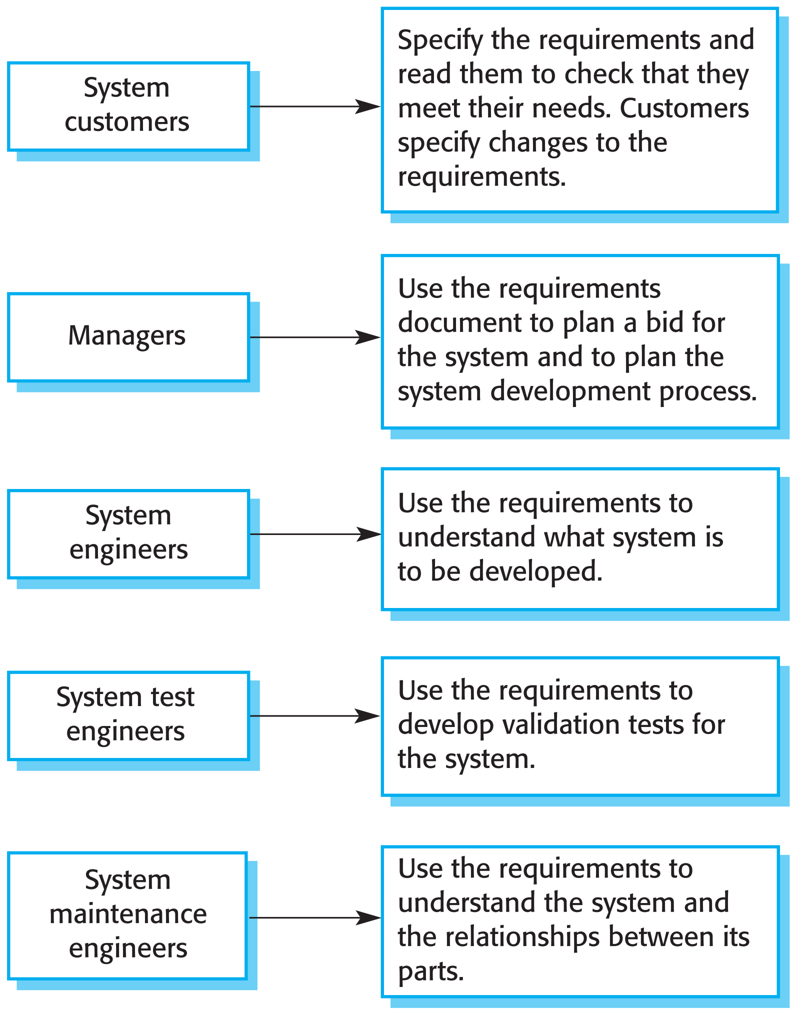
- In principle, requirements should state what the system should do and the design should describe how it does this

- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific design may be a domain requirement.

The requirements document

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

Users of a requirements document


System customers
Specify the requirements and read them to check that they meet their needs. Customers 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 understand the system and the relationships between its parts.

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 modularization
  - 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 (Ross, 1977) (Schoman and Ross, 1977). Now, use-case descriptions (Jacobsen, et al., 1993) and sequence diagrams are commonly used (Stevens and Pooley, 1999).
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 specifications

- Definition of the function or entity
- Description of inputs and where they come from
- Description of outputs and where they go to
- Indication of other entities required
- Pre and post conditions (if appropriate)
- The side effects (if any) of the function

Form-based node 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 Đthe 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

- Used to supplement natural language
- Particularly useful when you have to define a number of possible alternative courses of action

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

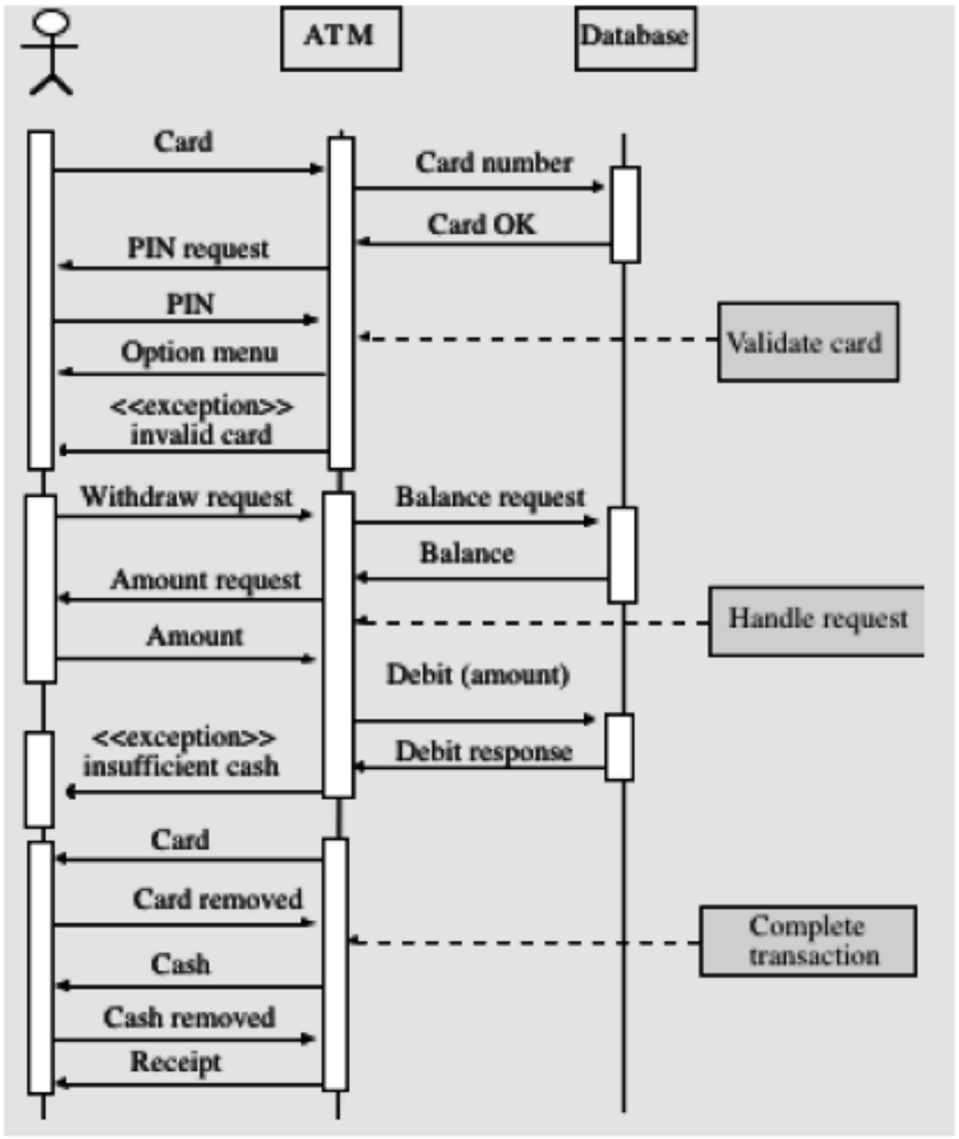
Graphical models

- Graphical models are most useful when you need to show how state changes or where you need 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



Source: Sommerville (2004)

Interface specification

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements
- Three types of interface may have to be defined
  - Procedural interfaces (i.e., APIs)
  - Data structures that are exchanged
  - Data representations (e.g., the ordering of bits)
- Formal notations are an effective technique for interface specification

Interface description

```
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

- The requirements document is the 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

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

Examples of requirements

- iTrust Medical Care Requirements Specification:
  - http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements

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
- System specifications are intended to communicate the functions that the system should provide
- A software requirements document is an agreed statement of the system requirements.

Acknowledgements

- Many slides courtesy of Rupak Majumdar