Systems Analysis and Design

Chapter 4

Requirements Engineering

The slide is mainly adopted from:

- J. S. Valacich, J. George, Modern Systems Analysis and Design. 8th Edition, Pearson 2017.
- I. Sommerville. Software Engineering. 10th Edition, Pearson, 2016

Topics covered

- Functional and non-functional requirements
- Requirements engineering processes
- Requirements elicitation
- Requirements specification
- Requirements validation
- Requirements change

Performing Requirements Engineering

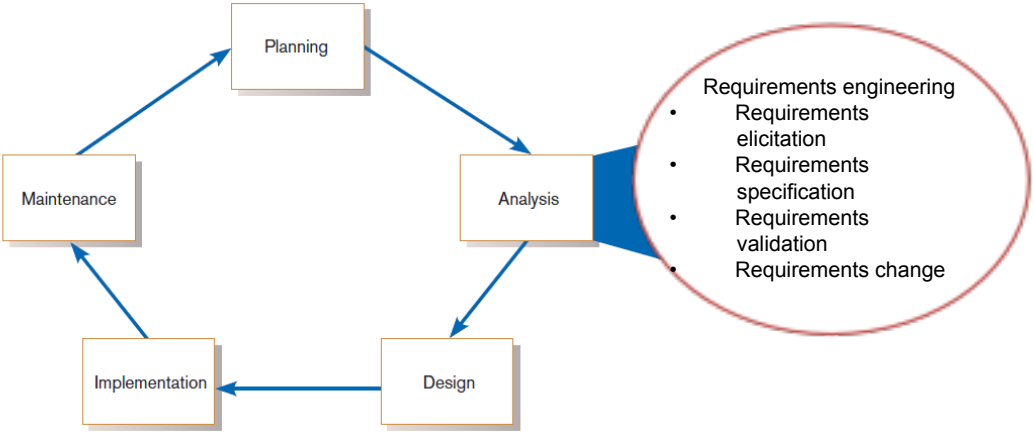


FIGURE 6-1
Systems development life cycle with analysis phase highlighted

Requirements engineering

- The process of **establishing** the **services** that a customer **requires** from a **system** and the **constraints** under which it operates and is developed.
- The system requirements are the **descriptions** of the system **services** and **constraints** that are generated during the requirements engineering 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 requirement

- User requirements
  - Statements in **natural language** 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's functions, services and operational constraints. Defines what **should** be implemented so may be part of a contract between client and contractor.

User and system requirements

User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

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.
  - May state what the system should not do.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
  - Constraints on the system from the domain of operation

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.
- Functional system requirements should describe the system services in detail.

Mentcare system: functional requirements

- A user shall be able to search the appointments lists for all clinics.
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements imprecision

- Problems arise when functional requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term ‘search’ in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

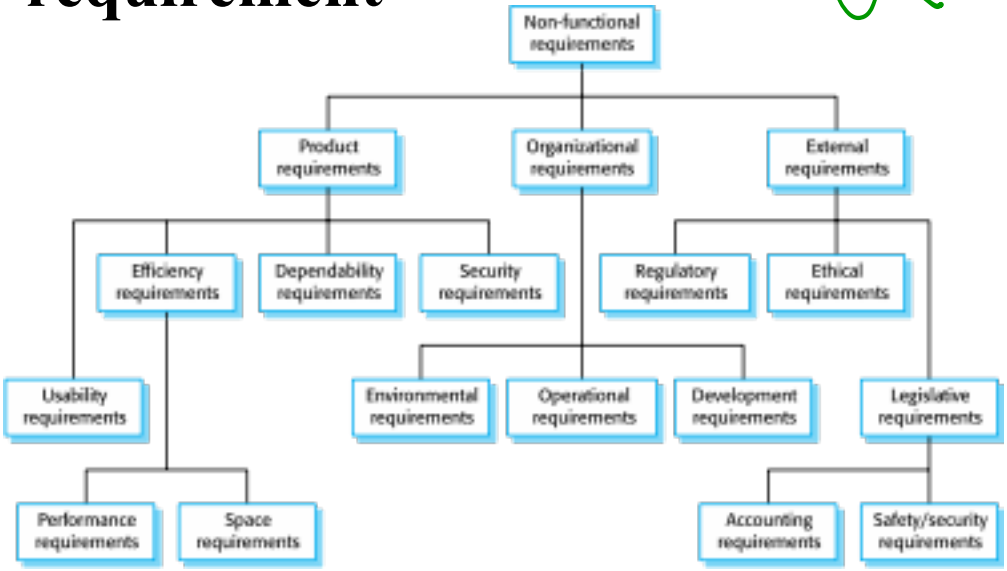
Requirements completeness and consistency

- In principle, requirements should be both complete and consistent.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements 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 IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Types of nonfunctional requirement



Non-functional requirements implementation

پیاده سازی

- Non-functional requirements may affect the **overall architecture** of a system rather than the **individual components**.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational 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.

Examples of nonfunctional requirements in the Mentcare system

Product requirement
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Goals and requirements

- Non-functional requirements 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.

Usability requirements

- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)
- Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (Testable non-functional requirement)

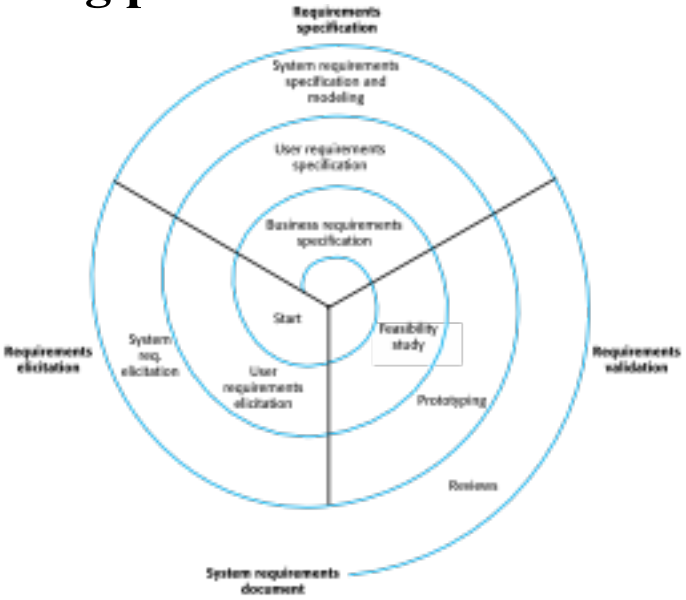
Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes 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

Requirements engineering processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- However, there are a number of generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements validation;
  - Requirements management.
- In practice, RE is an iterative activity in which these processes are interleaved.

A spiral view of the requirements engineering process



Requirements elicitation and analysis

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, ~~trade unions~~, etc. These are called *stakeholders*.

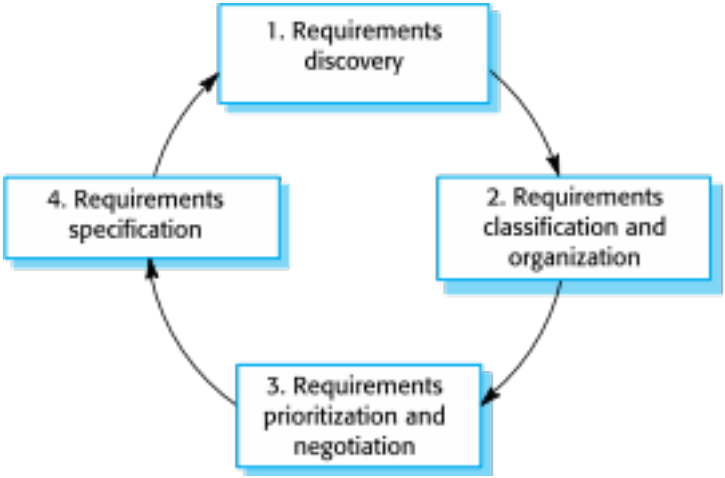
Requirements elicitation

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

Problems of requirements elicitation

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

The requirements elicitation and analysis process



Process activities

- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
  - Requirements are documented and input into the next round of the spiral.

Requirements discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Interaction is with system stakeholders from managers to external regulators.
- Systems normally have a range of stakeholders.

The Process of Determining Requirements (Requirements discovery)

- Good Systems Analyst Characteristics:
  - Impertinence—question everything
  - Impartiality—consider all issues to find the best organizational solution
  - Relax constraints—assume anything is possible
  - Attention to details—every fact must fit
  - Reframing—challenge yourself to new ways

Deliverables and Outcomes

- Deliverables for Requirements Determination:
  - From interviews and observations
    - interview transcripts, observation notes, meeting minutes
  - From existing written documents
    - mission and strategy statements, business forms, procedure manuals, job descriptions, training manuals, system documentation, flowcharts
  - From computerized sources
    - Joint Application Design session results, CASE repositories, reports from existing systems, displays and reports from system prototype

تهیین

Methods for Determining Requirements

- Traditional Methods
  - Interviewing individuals
  - Interviewing groups
  - Observing workers
  - Studying business documents
- Contemporary Methods
  - Joint Application Design (JAD)
  - Group Support Systems
  - CASE tools
  - System prototypes

طراحی برنامه مشترک

Interviewing and Listening

- One of the primary ways analysts gather information about an information systems project using formal or informal interviews with stakeholders
- Types of interview
  - **Closed** interviews based on pre-determined list of questions
  - **Open** interviews where various issues are explored with stakeholders.
- An **interview guide** is a document for developing, planning and conducting an interview.

how to behave

Guidelines for Effective Interviewing

- Plan the interview.
  - Prepare interviewee: appointment, priming questions.
  - Prepare agenda, checklist, questions.
- Listen carefully and take notes (tape record if permitted).
- Type & Review interview notes within 48 hours.
- Seek a variety of perspectives from the interviews.
- Don't phrase a question in a way that implies a right or wrong answer.
- Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
- Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system
- Don't set expectations about the new system unless you know these will be deliverables.

Problems with interviews

- Application specialists may use **language** to describe their work that isn't **easy** for the requirements engineer to understand.
- Interviews are not good for understanding domain requirements → سخت بازسازی‌ها
  - Requirements engineers cannot understand specific **domain terminology**;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

having or showing the ability to speak

Interviewing and Listening (Cont.)

Interview Outline	
Interviewee: *Name of person being interviewed*	Interviewer: *Name of person leading interview*
Location/Medium: *Office, conference room, or phone number*	Appointment Date: Start Time: End Time:
Objectives: *What data to collect* *On what to gain agreement* *What areas to explore*	Reminders: *Background/experience of interviewee* *Known opinions of interviewee*
Agenda: Introduction Background on Project Overview of Interview Topics to Be Covered Permission to Record Topic 1 Questions Topic 2 Questions … Summary of Major Points Questions from Interviewee Closing	Approximate Time: 1 minute 2 minutes 1 minute 5 minutes 7 minutes … 2 minutes 5 minutes 1 minute

FIGURE 6-2 Typical interview guide

Interviewing and Listening (Cont.)

General Observations:
Interviewee seemed busy probably need to call in a few days for follow-up questions because he gave only short answers. PC was turned off–probably not a regular PC user.

Unresolved Issues, Topics Not Covered:
He needs to look up sales figures from 1999. He raised the issue of how to handle returned goods, but we did not have time to discuss.

Interviewee:	Date:
Questions:	Notes:
When to ask question, if conditional *Question: 1* *Have you used the current sales tracking system? If so, how often ?* *If yes, go to Question 2*	*Answer* *Yes, I ask for a report on my product line weekly.* *Observations* *Seemed anxious–may be overestimating usage frequency.*
Question: 2 *What do you like least about the system?*	*Answer* *Sales are shown in units, not dollars.* *Observations* *System can show sales in dollars, but user does not know this.*

FIGURE 6-2 Typical interview guide (cont.)

Interviewing Groups

- Drawbacks to individual interviews:
  - Contradictions and inconsistencies between interviewees
  - Follow-up discussions are time consuming
  - New interviews may reveal new questions that require additional interviews with those interviewed earlier
- Interviewing several key people together
  - Advantages
    - More effective use of time
    - Can hear agreements and disagreements at once
    - Opportunity for synergies
  - Disadvantages
    - More difficult to schedule than individual interviews

Nominal Group Technique (NGT)

- A facilitated process that supports idea generation by groups
- Process
  - Members come together as a group, but initially work separately.
  - Each person writes ideas.
  - Facilitator reads ideas out loud, and they are written on a blackboard or flipchart.
  - Group openly discusses the ideas for clarification.
  - Ideas are prioritized, combined, selected, reduced.
- Used to complement group meetings or as part of JAD effort

Directly Observing Users

- **Direct Observation**
  - Watching users do their jobs
  - Used to obtain more firsthand and objective measures of employee interaction with information systems
  - Can cause people to change their normal operating behavior
  - Time-consuming and limited time to observe

Analyzing Procedures and Other Documents

روش بررسی و تحلیل اسناد

- **Document Analysis**
  - Review of existing business documents
  - Can give a historical and "formal" view of system requirements
- Types of information to be discovered:
  - Problems with existing system
  - Opportunity to meet new need
  - Organizational direction
  - Names of key individuals
  - Values of organization
  - Special information processing circumstances
  - Reasons for current system design
  - Rules for processing data

Analyzing Procedures (Cont.)

GUIDE FOR PREPARATION OF INVENTION DISCLOSURE
(See FACULTY and STAFF MANUALS for Detailed Patent Policy and Routing Procedures.)

(1) DISCLOSE ONLY ONE INVENTION PER FORM.

(2) PREPARE COMPLETE DISCLOSURE.

The disclosure of your invention is adequate for patent purposes ONLY if it enables a person skilled in the art to understand the invention.

(3) CONSIDER THE FOLLOWING IN PREPARING A COMPLETE DISCLOSURE:

(a) All essential elements of the invention, their relationship to one another, and their mode of operation.

(b) Equivalents that can be substituted for any elements.

(c) List of features believed to be new.

(d) Advantages this invention has over the prior art.

(e) Whether the invention has been built and/or tested.

FIGURE 6-3 Example of a procedure

Analyzing Procedures and Other Documents (Cont.)

روش‌های تجزیه و تحلیل

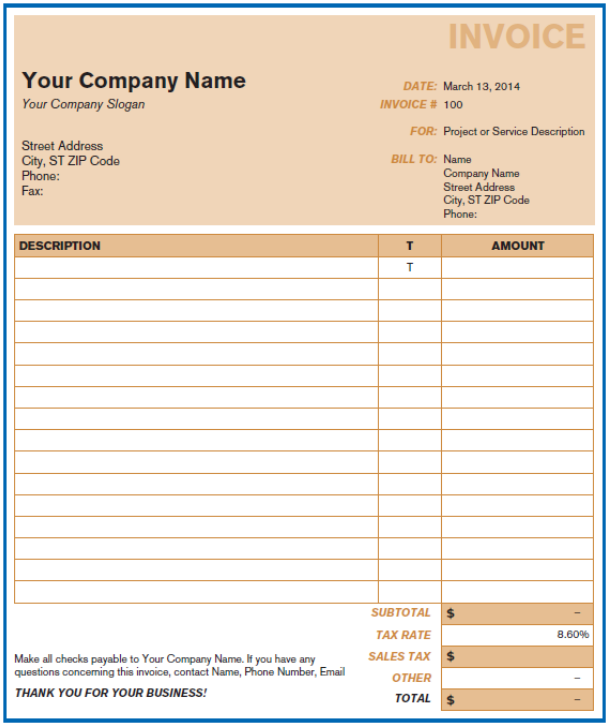
- Potential Problems with Procedure Documents:
  - May involve duplication of effort
  - May have missing procedures
  - May be out of date
  - May contradict information obtained through interviews

تناقض داشته باشد
یا برعکس عمل شده

Analyzing Procedures and Other Documents (Cont.)

- **Formal Systems**: the official way a system works as described in organizational documentation (i.e. work procedure)
- **Informal Systems**: the way a system actually works (i.e. interviews, observations)
- **Useful document: Business form**
  - Used for all types of business functions
  - Explicitly indicates what data flow in and out of a system and data necessary for the system to function
  - Gives crucial information about the nature of the organization
- **Useful document: Report**
  - Primary output of current system
  - Enables you to work backwards from the report to the data needed to generate it

Analyzing Procedures and Other Documents (Cont.)



INVOICE

Your Company Name
Your Company Slogan

Street Address
City, ST ZIP Code
Phone:
Fax:

DATE: March 13, 2014
INVOICE # 100
FOR: Project or Service Description
BILL TO: Name
Company Name
Street Address
City, ST ZIP Code
Phone:

DESCRIPTION	T	AMOUNT
	T	

	SUBTOTAL	$ –
	TAX RATE	8.60%
Make all checks payable to Your Company Name. If you have any questions concerning this invoice, contact Name, Phone Number, Email	SALES TAX	$
	OTHER	–
THANK YOU FOR YOUR BUSINESS!	TOTAL	$ –

FIGURE 6-4
An invoice form from Microsoft Excel

Contemporary Methods for Determining System Requirements

- **Joint Application Design (JAD)**
  - Brings together key users, managers, and systems analysts
  - Purpose: collect system requirements simultaneously from key people
  - Conducted off-site
- **Group Support Systems**
  - Facilitate sharing of ideas and voicing of opinions about system requirements
- **CASE tools**
  - Used to analyze existing systems
  - Help discover requirements to meet changing business conditions
- **System prototypes**
  - Iterative development process
  - Rudimentary working version of system is built
  - Refine understanding of system requirements in concrete terms

Joint Application Design (JAD)

- Intensive group-oriented requirements determination technique
- Team members meet in isolation for an extended period of time
- Highly focused
- Resource intensive
- Started by IBM in 1970s

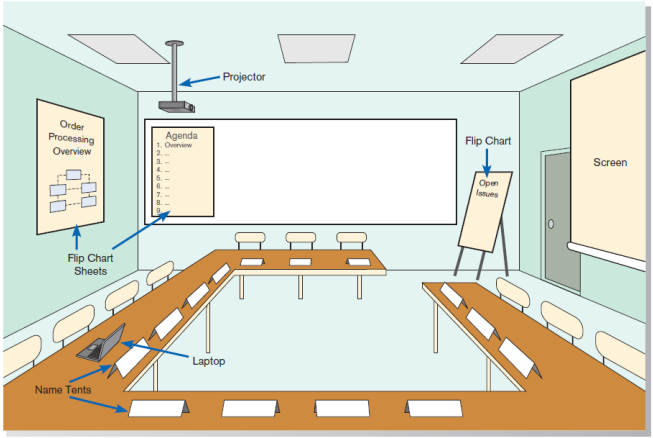


FIGURE 6-6 Illustration of the typical room layout for a JAD
Source: Based on Wood and Silver, 1995.

JAD (Cont.)

- JAD Participants:
  - **Session Leader**: facilitates group process
  - **Users:** active, speaking participants
  - **Managers**: active, speaking participants
  - **Sponsor**: high-level champion, limited participation
  - **Systems Analysts**: should mostly listen
  - **Scribe**: record session activities
  - **IS Staff**: should mostly listen
- End Result
  - Documentation detailing existing system
  - Features of proposed system

CASE Tools During JAD

- Upper CASE tools are used
- Enables analysts to enter system models directly into CASE during the JAD session
- Screen designs and prototyping can be done during JAD and shown to users

Using Prototyping During Requirements Determination

- Quickly converts requirements to working version of system
- Once the user sees requirements converted to system, will ask for modifications or will generate additional requests

Using Prototyping During Requirements Determination (Cont.)

- Most useful when:
  - User requests are not clear.
  - Few users are involved in the system.
  - Designs are complex and require concrete form.
  - There is a history of communication problems between analysts and users.
  - Tools are readily available to build prototype.
- Drawbacks
  - Tendency to avoid formal documentation
  - Difficult to adapt to more general user audience
  - Sharing data with other systems is often not considered
  - Systems Development Life Cycle (SDLC) checks are often bypassed

Radical Methods for Determining System Requirements

- **Business Process Reengineering (BPR)**: search for and implementation of radical change in business processes to achieve breakthrough improvements in products and services.
- Goals
  - Reorganize complete flow of data in major sections of an organization.
  - Eliminate unnecessary steps.
  - Combine steps.
  - Become more responsive to future change.

Disruptive Technologies

- Information technologies must be applied to radically improve business processes.
- **Disruptive technologies** are technologies that enable the breaking of long-held business rules that inhibit organizations from making radical business changes.

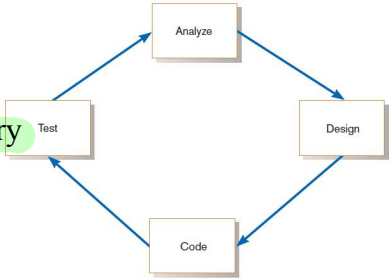
Disruptive Technologies (Cont.)

TABLE 6-6 Long-Held Organizational Rules That Are Being Eliminated through Disruptive Technologies

Rule	Disruptive Technology
Information can appear in only one place at a time.	Distributed databases allow the sharing of information.
Businesses must choose between centralization and decentralization.	Advanced telecommunications networks can support dynamic organizational structures.
Managers must make all decisions.	Decision-support tools can aid nonmanagers.
Field personnel need offices where they can receive, store, retrieve, and transmit information.	Wireless data communication and portable computers provide a "virtual" office for workers.
The best contact with a potential buyer is personal contact.	Interactive communication technologies allow complex messaging capabilities.
You have to find out where things are.	Automatic identification and tracking technology knows where things are.
Plans get revised periodically.	High-performance computing can provide real-time updating.

Agile methods and requirements

- Many agile methods argue that producing detailed system requirements is a **waste of time** as requirements change so quickly.
- The requirements document is therefore always **out of date**.
- Agile methods usually use incremental requirements engineering and may express requirements as 'user stories'.
- Replace traditional SDLC waterfall with **iterative analyze–design–code–test** cycle.
- Agile usage-centered design (Focuses on **user goals**, **roles**, and **tasks**)
- This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. **critical systems**) or systems developed by several teams.



Agile Usage-Centered Design Steps

- Gather group of programmers, analysts, users, testers, facilitator.
- Document complaints of current system.
- Determine important user roles.
- Determine, prioritize, and describe tasks for each user role.
- Group similar tasks into interaction contexts.
- Associate each interaction context with a user interface for the system, and prototype the interaction context.
- Step through and modify the prototype.

Requirements specification

- The process of writing down the user and system requirements in a requirements document.
- User requirements have to be understandable by end-users and customers who do not have a technical background.
- System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
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 now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

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 architecture to satisfy non-functional requirements may be a domain requirement.
  - This may be the consequence of a regulatory requirement.

Natural language specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
- Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Guidelines for writing requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way. 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**.
- Include an explanation (rationale) of why a requirement is necessary.

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 ارغام سازی ها
  - Several different requirements may be expressed together.

Example requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. ***(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)***

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. ***(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)***

Structured specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

unable to bend

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.
- Information about the information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

A structured specification of a requirement for an insulin pump

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.

Requirements

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

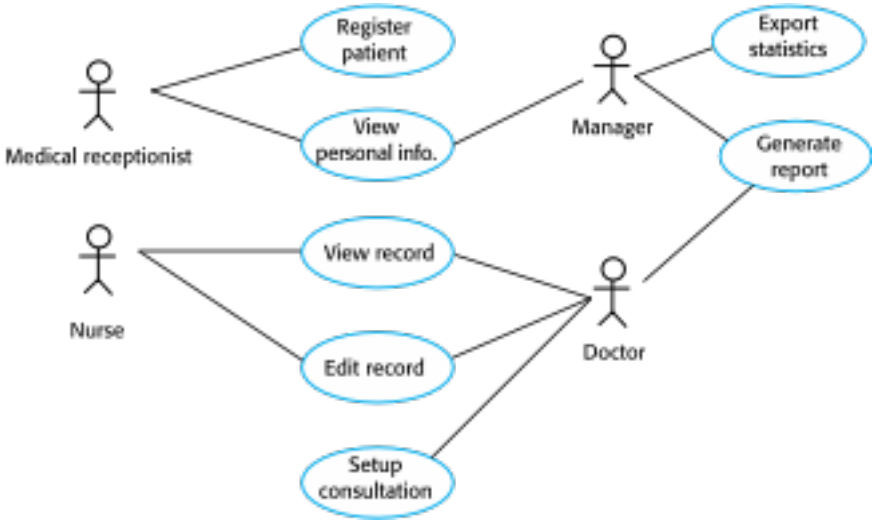
- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

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

Use cases

- **Use-cases** are a kind of **scenario** that are included in the UML.
- Use cases 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.
- High-level graphical model supplemented by more detailed tabular description.
- UML **sequence diagrams** may be used to add detail to use-cases by showing the sequence of event processing in the system.

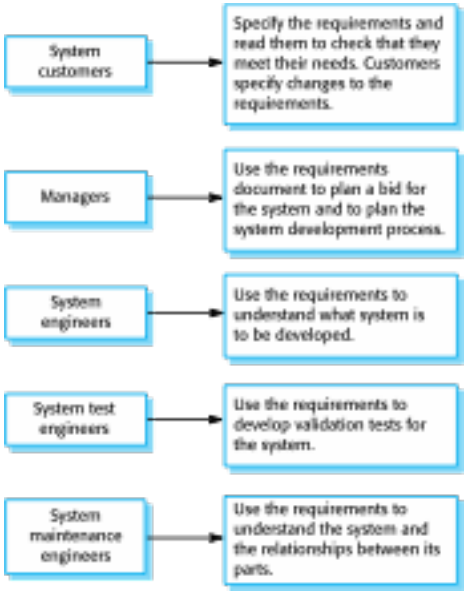
Use cases for the Mentcare system



The software requirements document

- The software 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.

Users of a requirements document



Requirements document variability

- Information in requirements document depends on type of system and the approach to development used.
- Systems developed incrementally will, typically, have less detail in the requirements document.
- Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The structure of a requirements document

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking

- **Validity**. Does the system provide the functions which best support the customer's needs?
- **Consistency**. Are there any requirements conflicts?
- **Completeness**. Are all functions required by the customer included?
- **Realism**. Can the requirements be implemented given available budget and technology
- **Verifiability**. Can the requirements be checked?

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

Requirements reviews

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Review checks

- Verifiability
  - Is the requirement realistically testable?
- Comprehensibility
  - Is the requirement properly understood?
- Traceability
  - Is the origin of the requirement clearly stated?
- Adaptability
  - Can the requirement be changed without a large impact on other requirements?

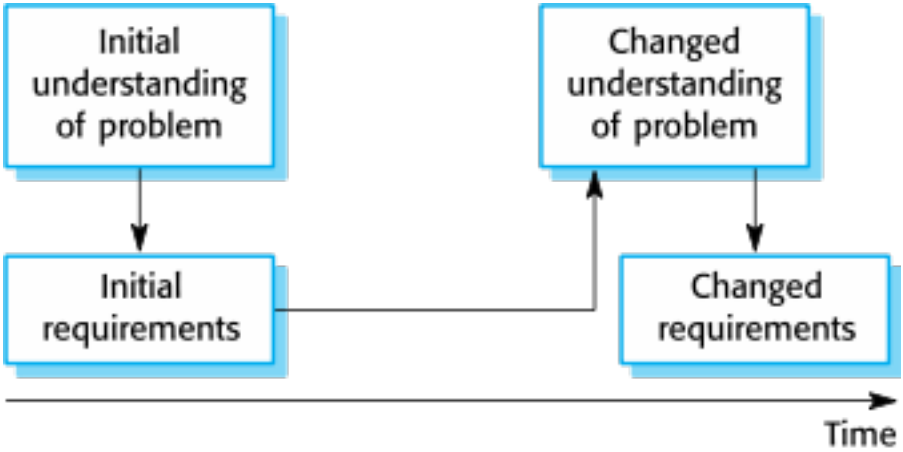
Changing requirements

- The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.
- The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose (forced to be accepted) requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing requirements

- Large systems usually have a **diverse** user community, with **many** users having **different** requirements and **priorities** that may be **conflicting** or contradictory.
  - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements evolution



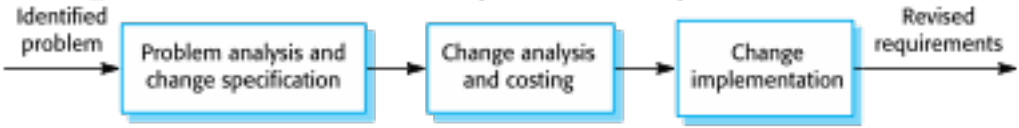
Requirements management

- Requirements management is the process of managing **changing requirements** during the requirements engineering process and system development.
- **New requirements** emerge as a system is being developed and after it has gone into use.
- You need to keep **track** of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.

Requirements management planning

- Establishes the level of requirements management detail that is required.
- Requirements management decisions:
  - ***Requirements identification*** Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - ***A change management process*** This is the set of activities that assess the impact and cost of changes.
  - ***Traceability policies*** These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.
  - ***Tool support*** Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements change management



- Deciding if a requirements change should be accepted
  - *Problem analysis and change specification*
    - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
  - *Change analysis and costing*
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
  - Change implementation
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Summary

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.
- The requirements engineering process is an iterative process that includes requirements elicitation, specification and validation.
- Requirements elicitation is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.

Summary (cont.)

- You learned how to Participate in and help plan Joint Application Design sessions.
- You can use a range of techniques for requirements elicitation including interviews and ethnography. User stories and scenarios may be used to facilitate discussions.
- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.
- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.

References

- **Chapter 4.** I. Sommerville. **Software Engineering**. 10th Edition, Pearson, 2016.

- **Chapter 6,** J. S. Valacich, J. George, **Modern Systems Analysis and Design**. 8th Edition, Pearson 2017.