Chapter 4 - Requirements Engineering

MI Qing (Lecturer)

Telephone: 15210503242

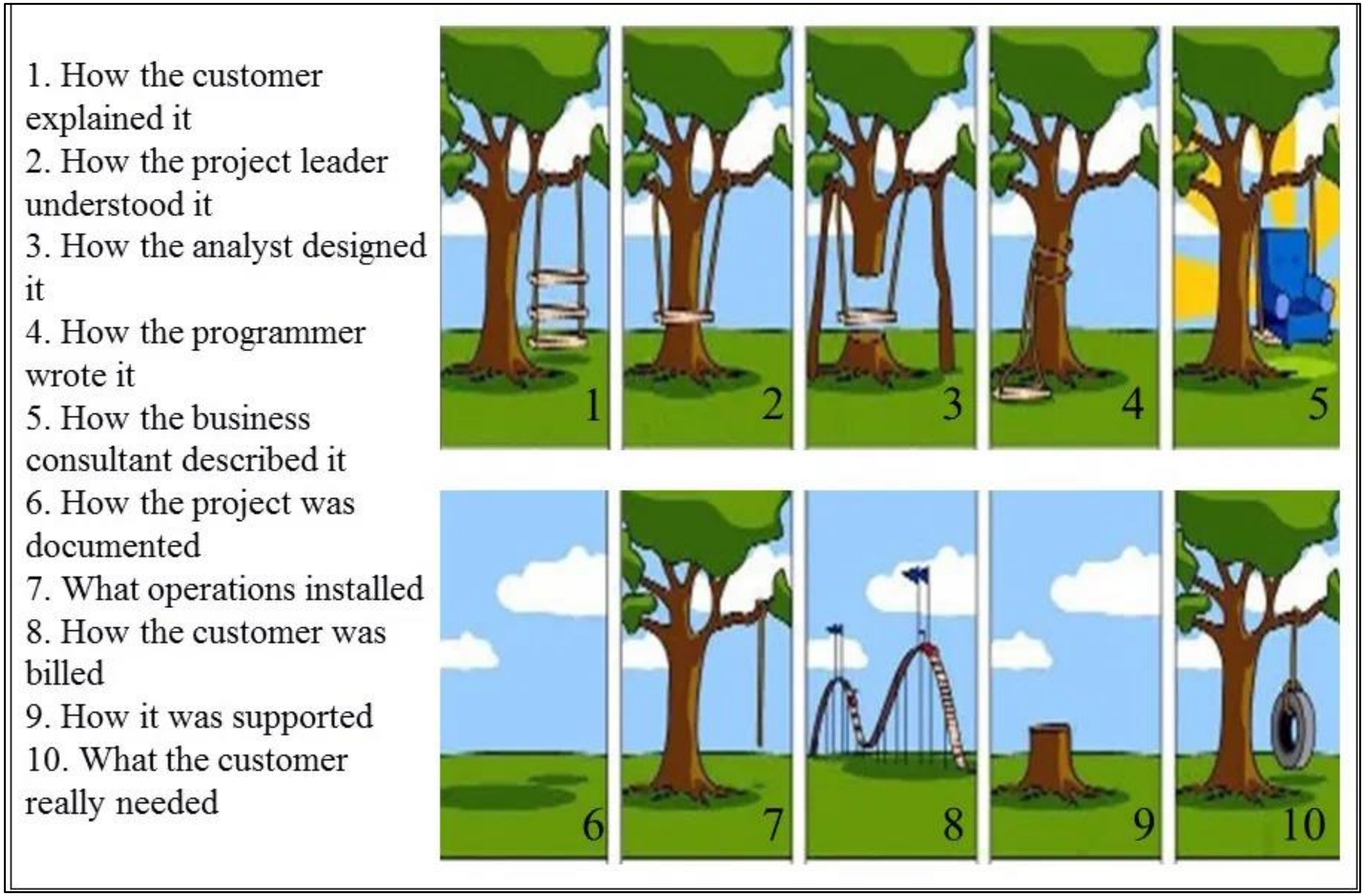
Email: miqing@bjut.edu.cn

Office: 410, Information Building

Topics Covered

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

Importance of Requirements Engineering



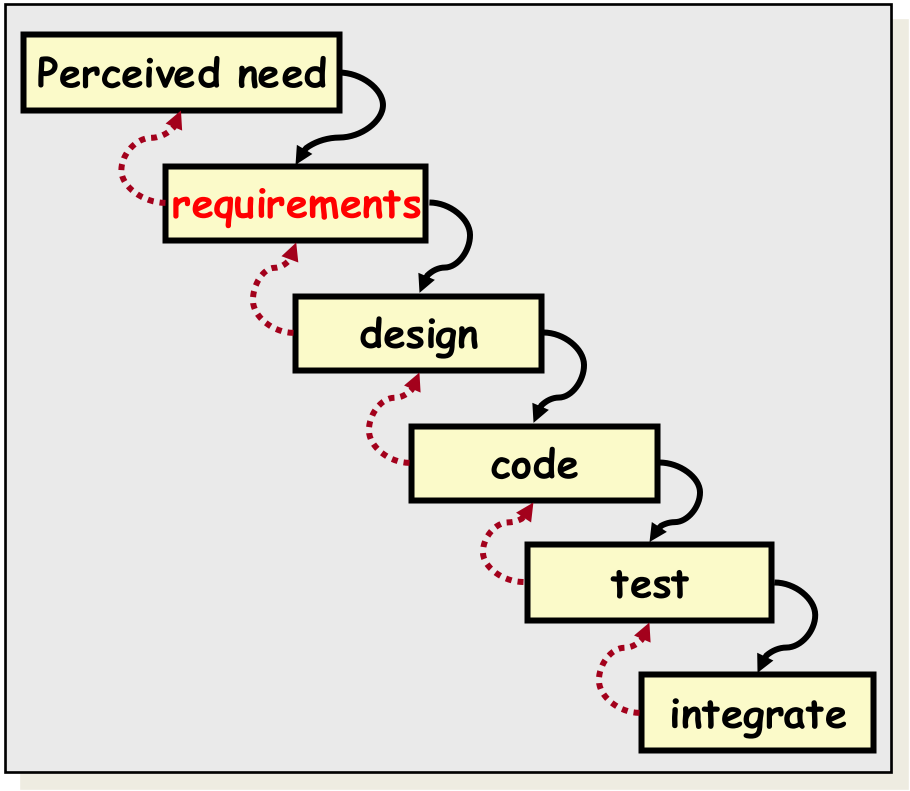
1. How the customer explained it
2. How the project leader understood it
3. How the analyst designed it
4. How the programmer wrote it
5. How the business consultant described it
6. How the project was documented
7. What operations installed
8. How the customer was billed
9. How it was supported
10. What the customer really needed

The hardest single part of building a software system is deciding precisely **what** to build.

-Fred Brooks

Requirements Engineering

- ✧ The requirements for a system are the descriptions of the services that a system should provide and the constraints on its operation.
- ✧ The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).



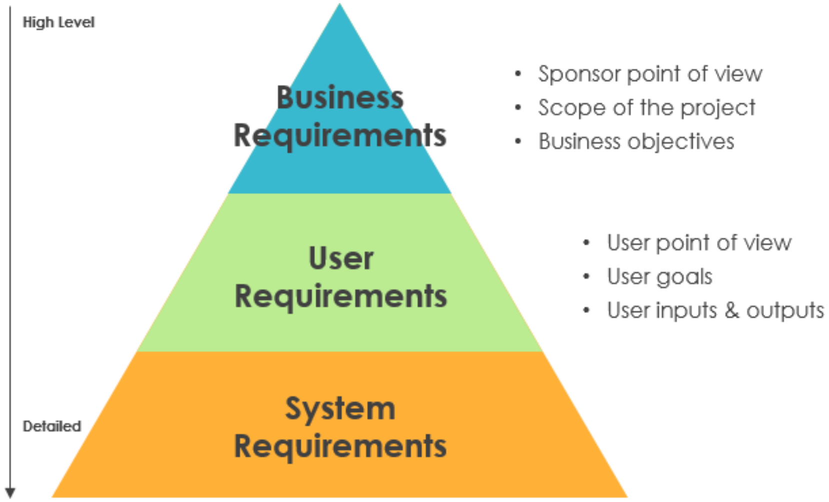
Requirements Abstraction

“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 organization’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.”

Davis, A. M. 1993. Software Requirements: Objects, Functions
and States. Englewood Cliffs, NJ: Prentice-Hall.

Types of Requirement

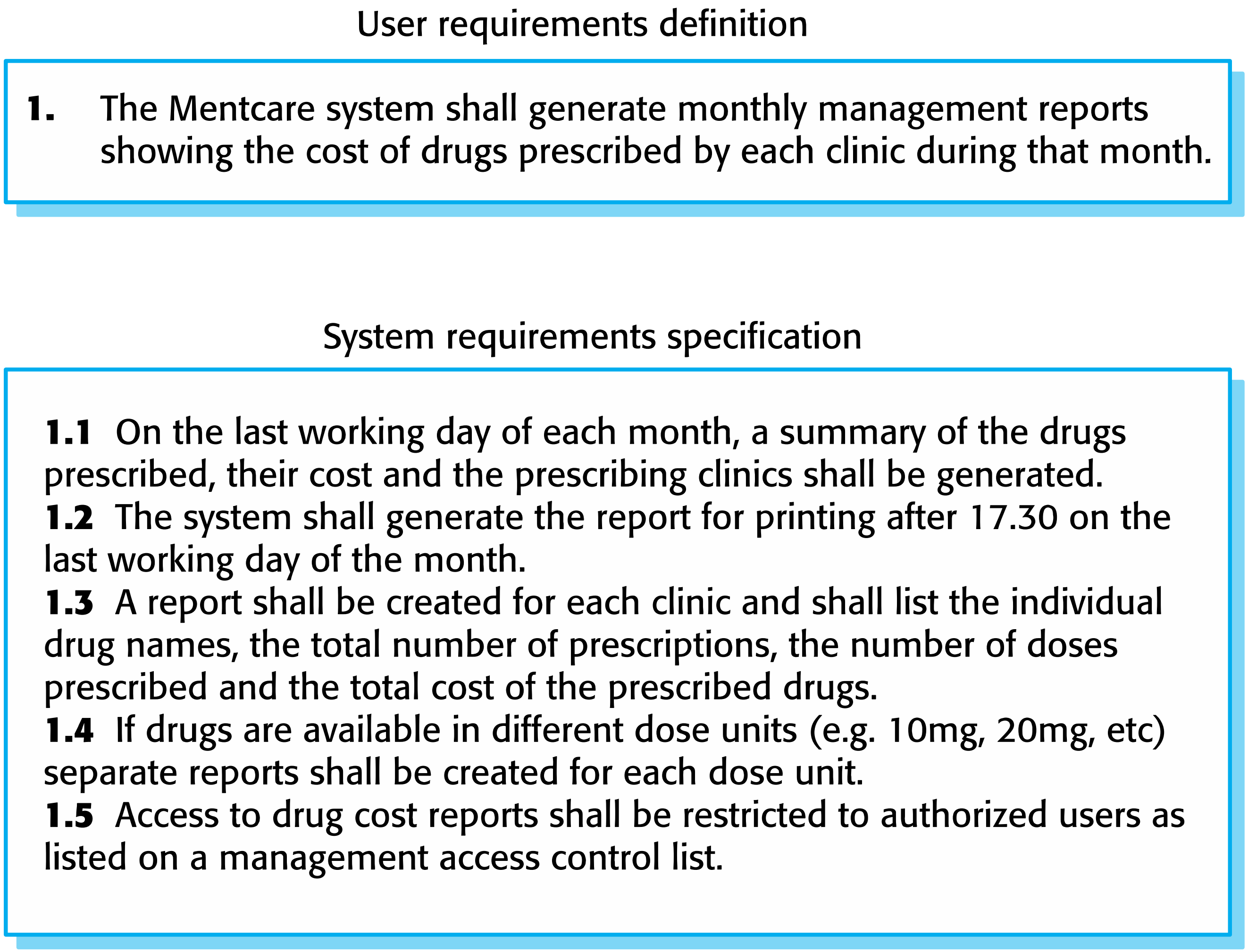
- ✧ User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints.
- ✧ System requirements
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of the contract between the system buyer and the software developers.



- Sponsor point of view
- Scope of the project
- Business objectives

- User point of view
- User goals
- User inputs & outputs

User and System Requirements



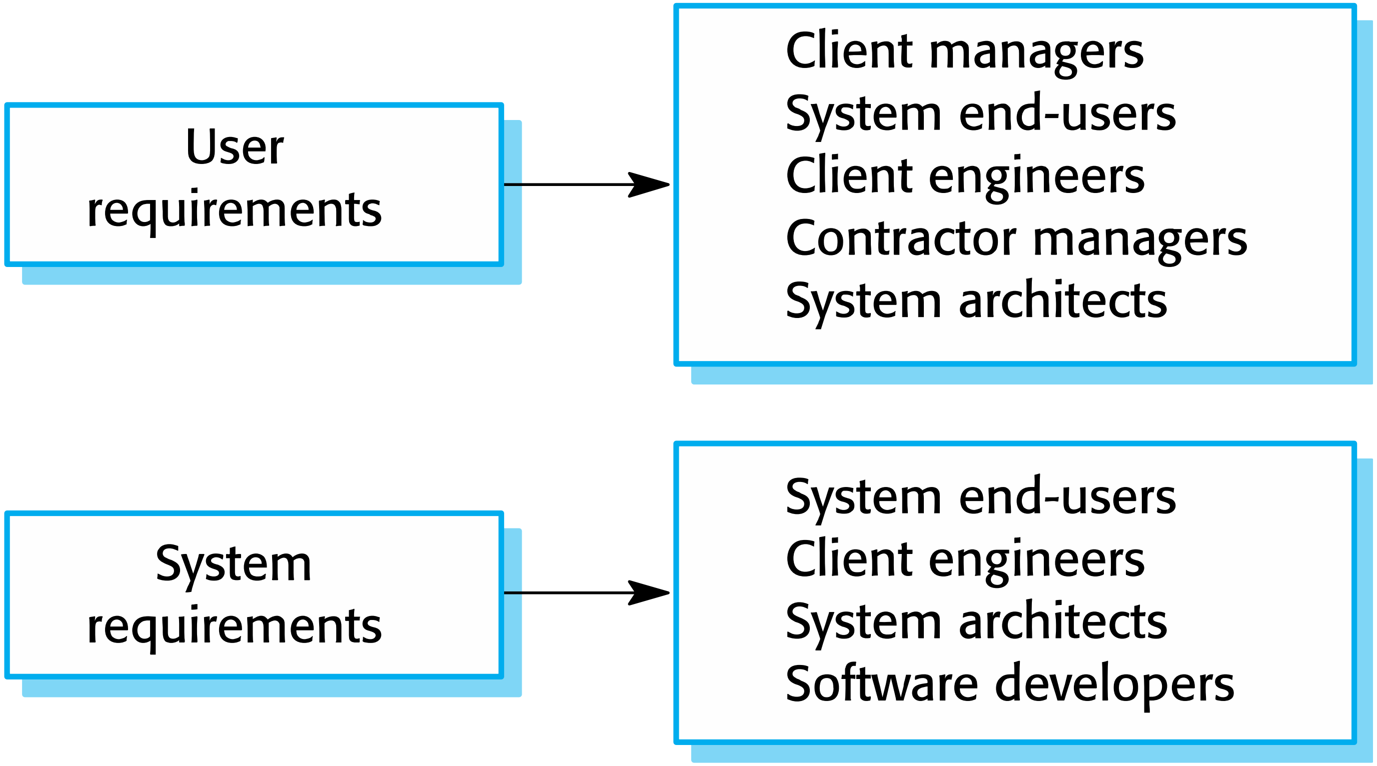
User requirements definition

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.

Readers of Different Types of Requirements Specification



User requirements

Client managers
System end-users
Client engineers
Contractor managers
System architects

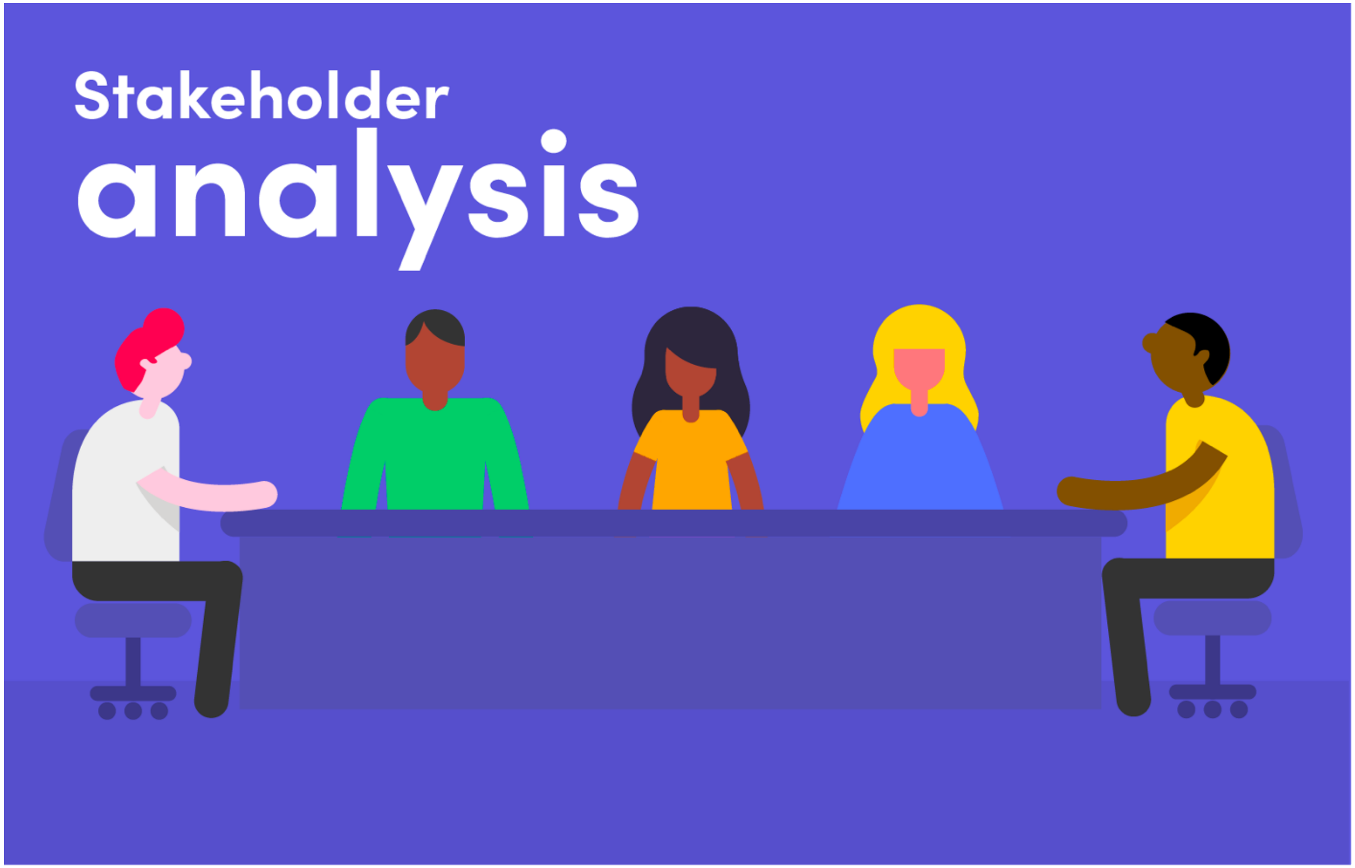
System requirements

System end-users
Client engineers
System architects
Software developers

System Stakeholders

- ✧ Any person or organization who is affected by the system in some way and so who has a legitimate interest.
- ✧ Stakeholder samples:
  - 用户：关心新系统特征和功能
  - 客户：为新系统买单的人
  - 系统分析师：想要获取正确的需求
  - 设计师：想要构造完美的系统，尽量重用已有的代码
  - 培训与用户支持人员：确保系统可用和可管理
  - 业务分析师：想确保“我们做得比竞争对手好”
  - 技术文档作者：为系统准备用户手册及其他相关文档
  - 项目经理：希望按时、按预算、按目标完成项目

Stakeholders in the Mentcare System



Functional and Non-functional Requirements

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.

Functional Requirements Examples

1. A user shall be able to search the appointments lists for all clinics.

2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

3. Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements Imprecision

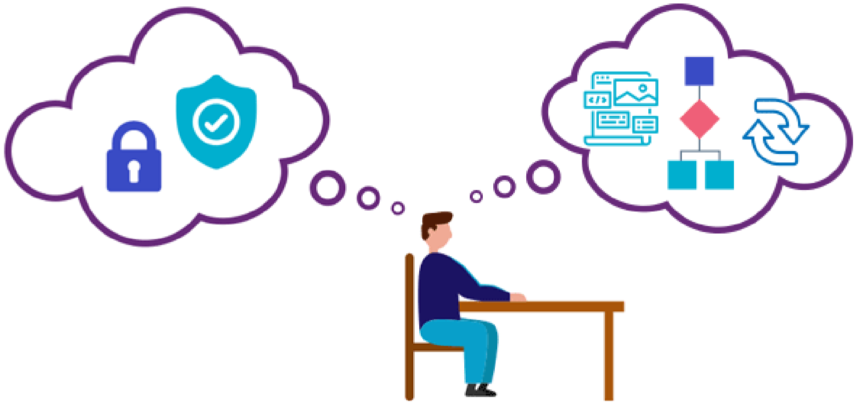
- ✧ Problems arise when functional requirements are not precisely stated. Ambiguous requirements may be interpreted in different ways by developers and users.
- ✧ Consider the term 'search' in requirement 1:
  - ▪ User intention: search for a patient name across all appointments in all clinics.
  - ▪ Developer interpretation: search for a patient name in an individual clinic. User chooses clinic then search.
- ✧ In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Non-functional Requirements

- Non-functional requirements usually specify or constrain characteristics of the system as a whole.
  - Properties such as reliability, response time and memory use.
  - Constraints such the capabilities of I/O devices or the data representations used in interfaces with other systems.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Non-Functional Requirements

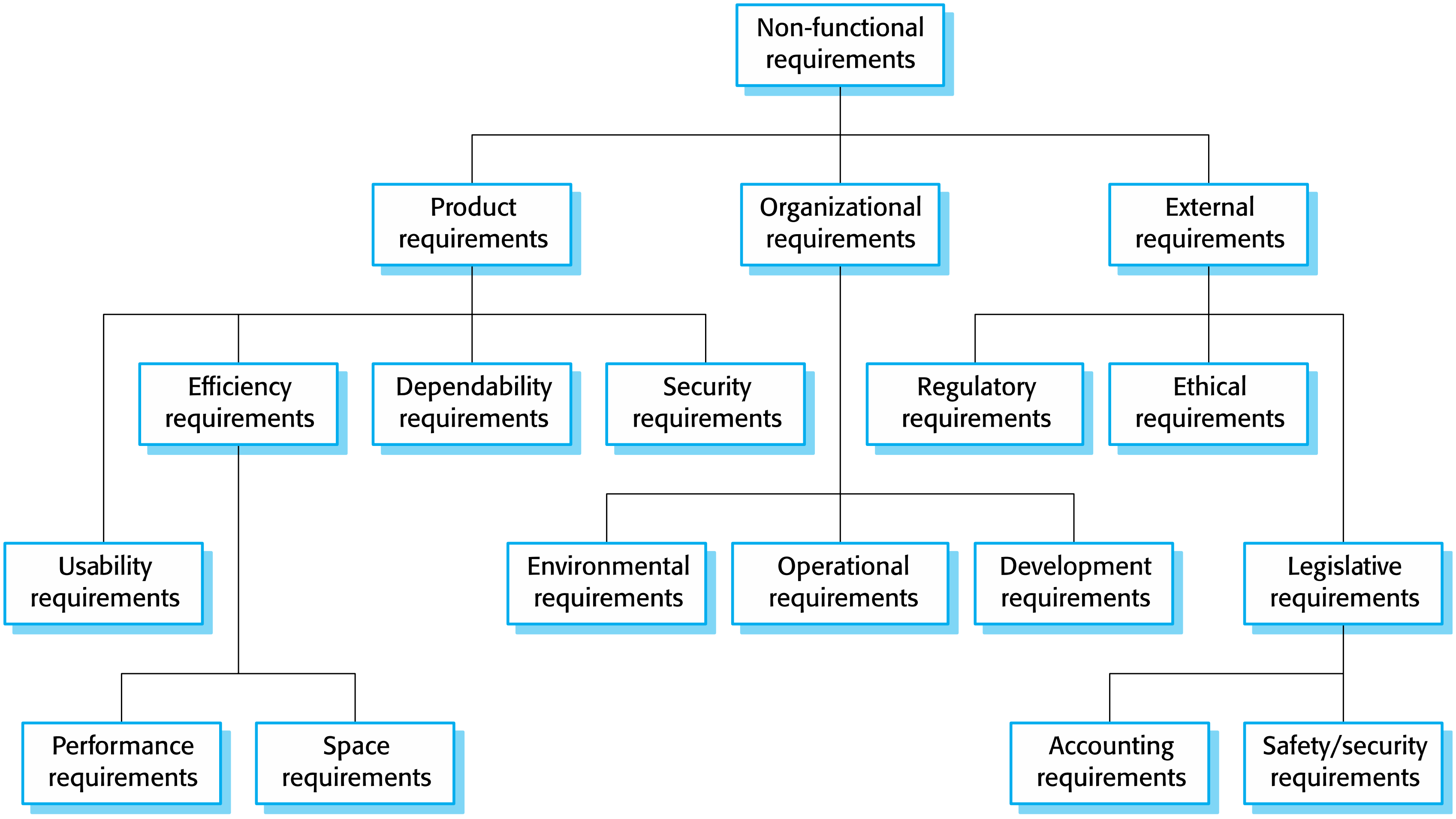
Functional Requirements



Non-functional Requirements Implementation

- 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.

Types of Non-functional Requirement



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. the programming language or process standards to be used.
- ✧ External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. legislative requirements.

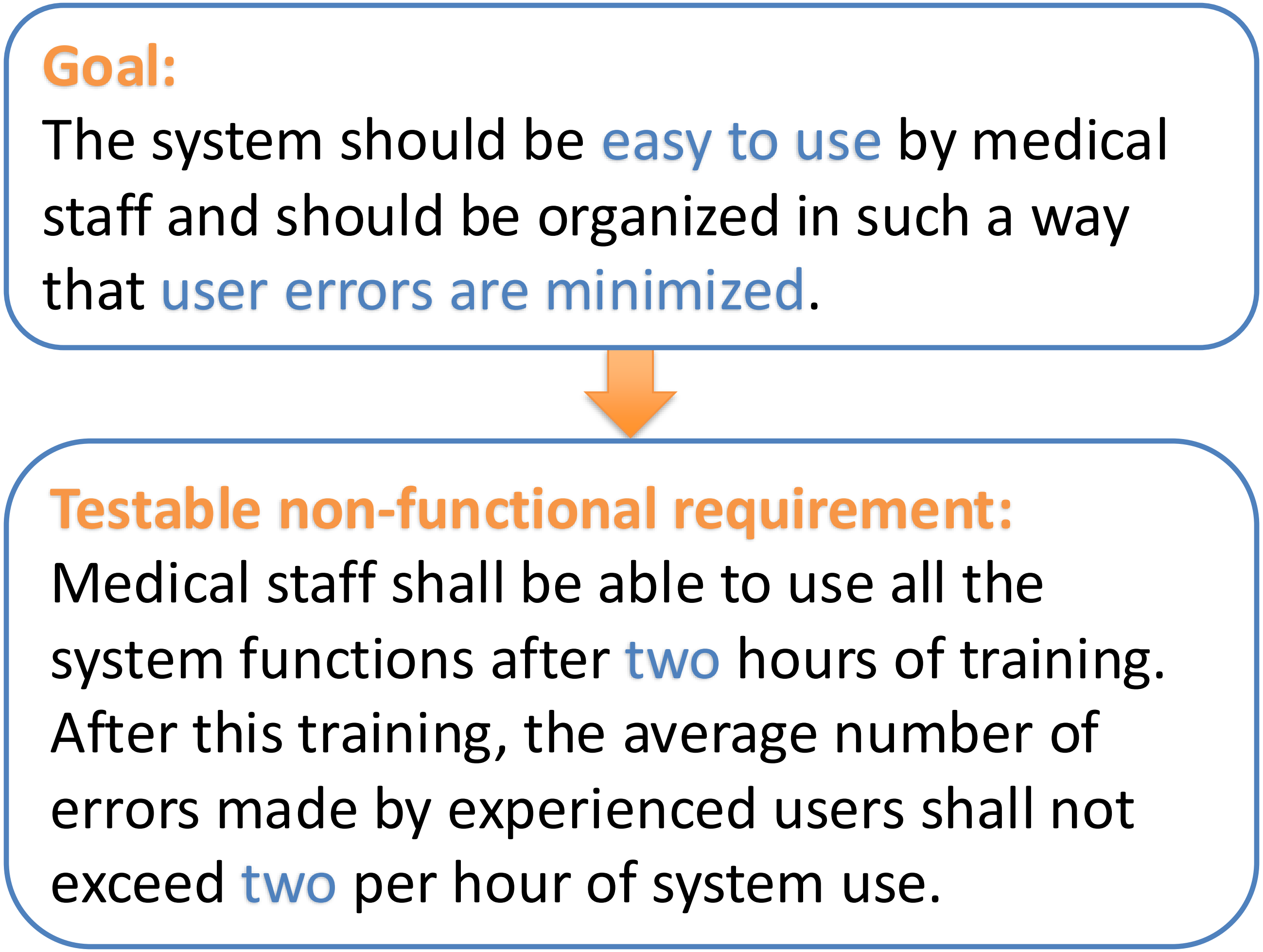
Non-functional Requirements Examples

Product requirement
The Mentcare system shall be available to all clinics during normal working hours (Mon-Fri, 08:30-17:30). 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.

Usability Requirements



Goal:

The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

Testable non-functional requirement:

Medical staff shall be able to use all the system functions after two hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use.

Metrics for Specifying Non-functional 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

Differences between Functional and Non-functional Requirements (a)

Functional	Non-functional
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies “What should the software system do?”	It places constraints on “How should the software system fulfill the functional requirements?”
Functional requirement is usually specified by User.	Non-functional requirement is usually specified by technical peoples e.g. Architect, Technical Leaders and Software Developers.
It is captured in use case.	It is captured as a quality attribute.

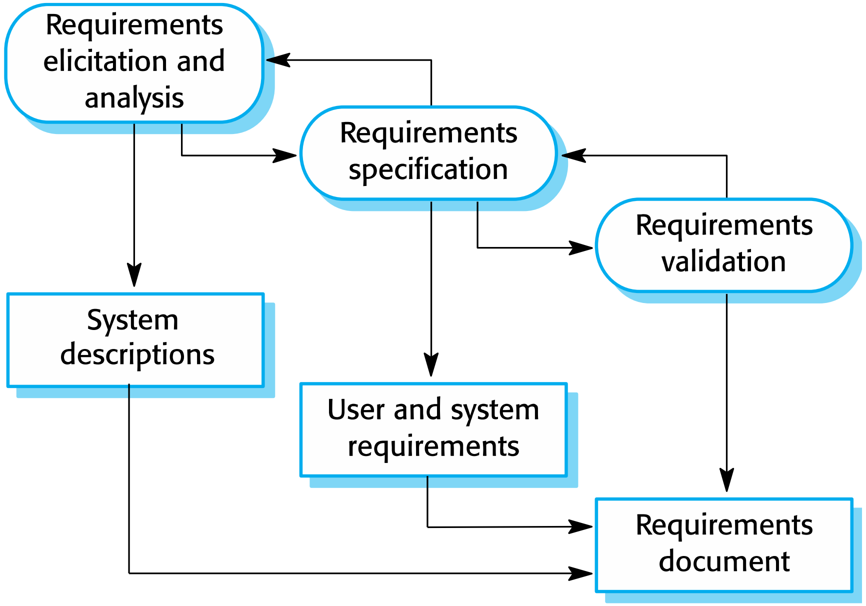
Differences between Functional and Non-functional Requirements (b)

Functional	Non-functional
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, API testing, etc.	Non-functional Testing like Performance, Stress, Usability, Security testing, etc.
Usually easy to define.	Usually more difficult to define.
Examples A verification email is sent to user whenever he/she registers for the first time on our website.	**Examples** The site should load in 3 seconds when the number of simultaneous users are > 10000.

Requirements Engineering Processes

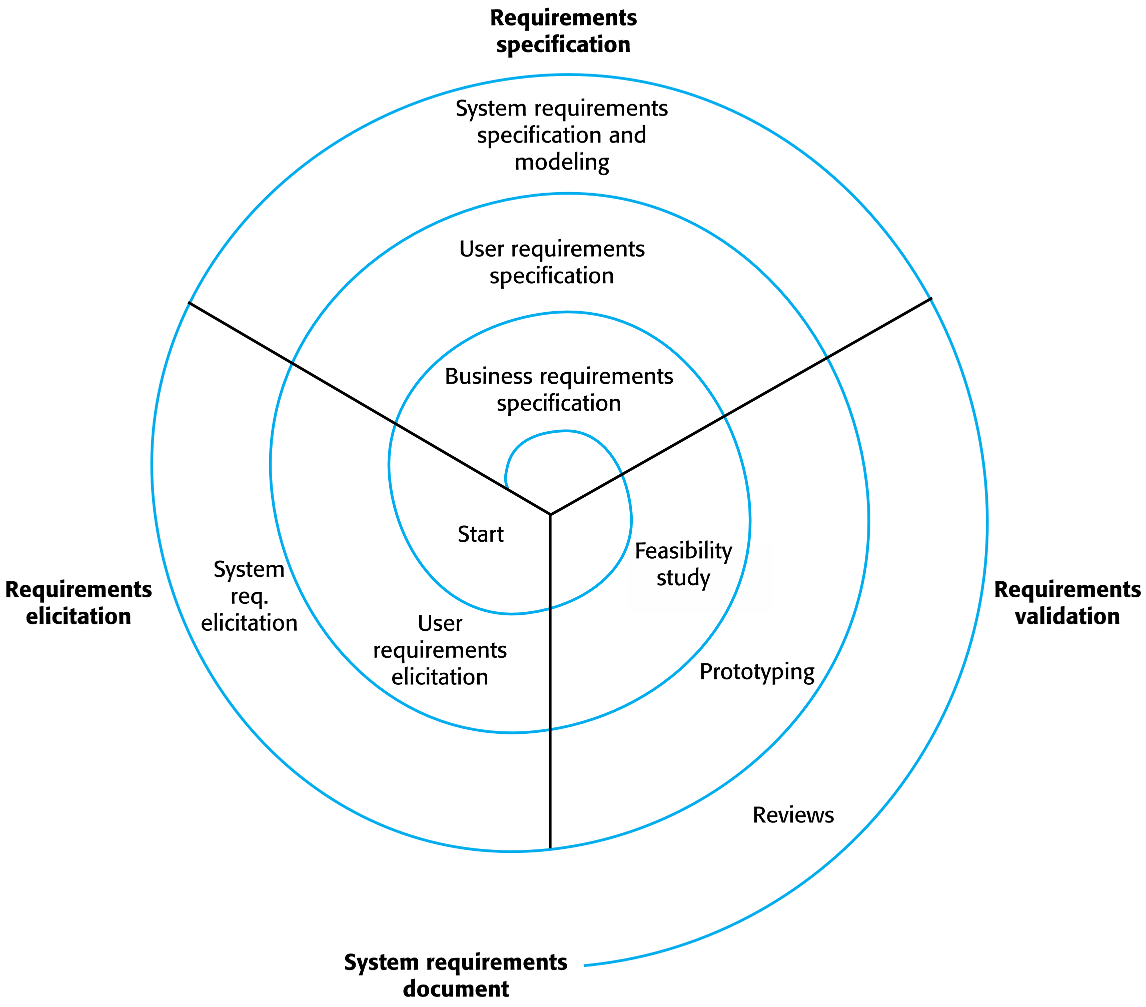
Requirements Engineering Processes

- ✧ Requirements engineering involves three key activities:
  - Discovering requirements by interacting with stakeholders (elicitation and analysis)
  - Converting these requirements into a standard form (specification)
  - Checking that the requirements actually define the system that the customer wants (validation)
- ✧ In practice, RE is an iterative activity in which these processes are interleaved.



Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

A Spiral View of the Requirements Engineering Process


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Prototyping
Reviews
Requirements elicitation
System req. elicitation
User requirements elicitation
Requirements validation
System requirements document

Requirements Elicitation

Requirements Elicitation

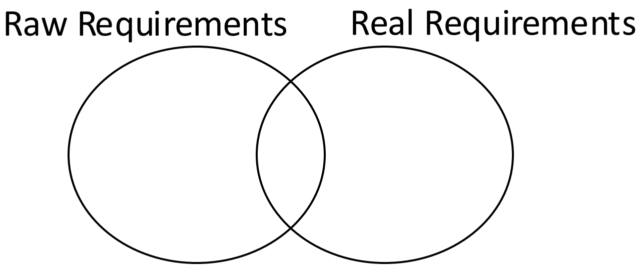
- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- 5W2H

©JKING, ModernAnalyst.Com
WHY SHOULD I CLEAN MY ROOM?
WHAT'S THE CURRENT PROCESS FOR DUSTING?
ISN'T HOMEWORK MORE IMPORTANT?
HOW DO I KNOW WHEN IT'S CLEAN?
JUST FORGET IT.

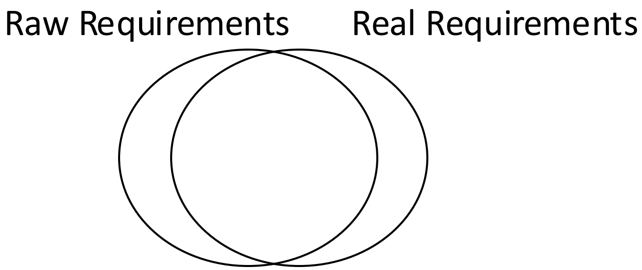
Why Define Requirements?

- Ill-defined requirements and unmanaged requirement changes lead to:
  - Disagreement with customers on requirements and product acceptance
  - Unrealistic estimates and commitments to customers
  - Schedule slippage
  - Cost overruns
  - Uncertainty regarding project closure
  - Team burnout
  - Unhappy customers
- It is critical to involve the stakeholders early in the requirements elicitation process.

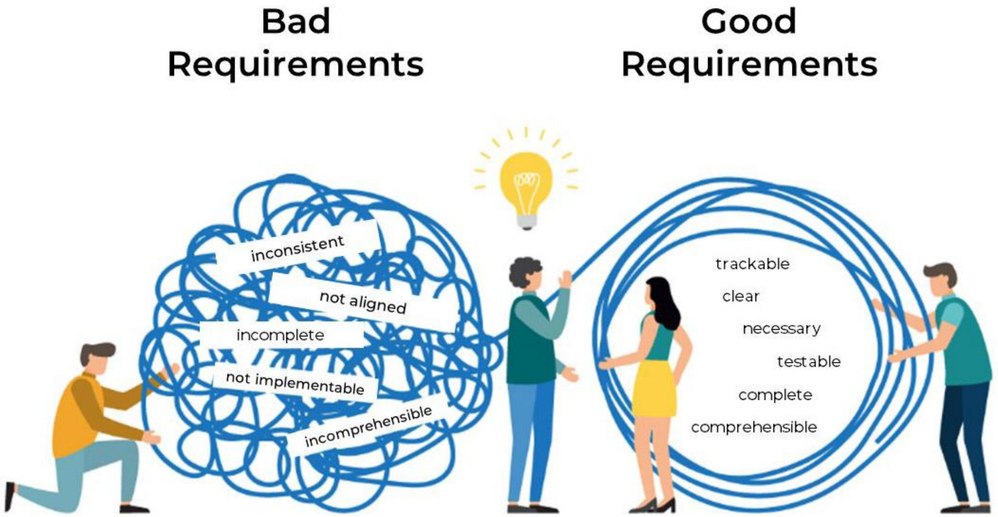
Requirements Elicitation



Without customer-oriented requirements-gathering practices



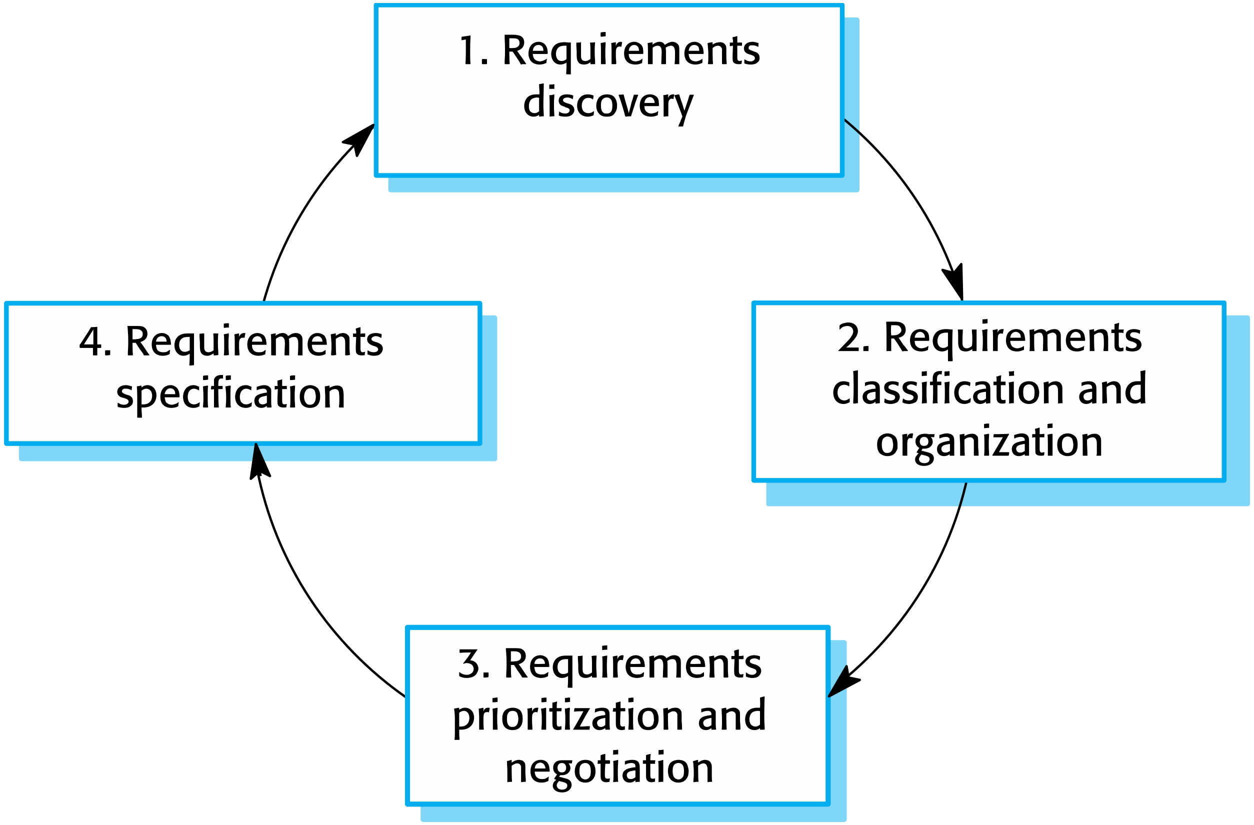
With customer-oriented requirements-gathering practices



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.

Requirements Elicitation and Analysis Process


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

Process Activities

- ✧ Requirements discovery
  - Interacting with stakeholders to discover their requirements.
- ✧ Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- ✧ Requirements prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- ✧ Requirements specification
  - Requirements are documented and input into the next round of the spiral.

“Just Enough” Elicitation

- Too much: You would spend so much time understanding the problem that no time would be left to solve it.
- Too little: You would build system before understanding problem, and would likely build the wrong system.

- Don’t ignore elicitation
- Recognize that 1 stakeholder cannot speak for all
- Maintain glossary of terms
- Use appropriate elicitation techniques
- Prepare for change

Who’s Smart?

- ✧ Don’t try to convince stakeholders that you are smart: Wrong Place to Do That.
- ✧ Instead: Take every opportunity to show you think the stakeholder is smart.

My Elevators Are Too Slow.
I Don’t Think So. I Think You Have an Elevator Throughput Problem, not a Speed Problem.
I See. Tell Me Why You Feel They Are Too Slow.

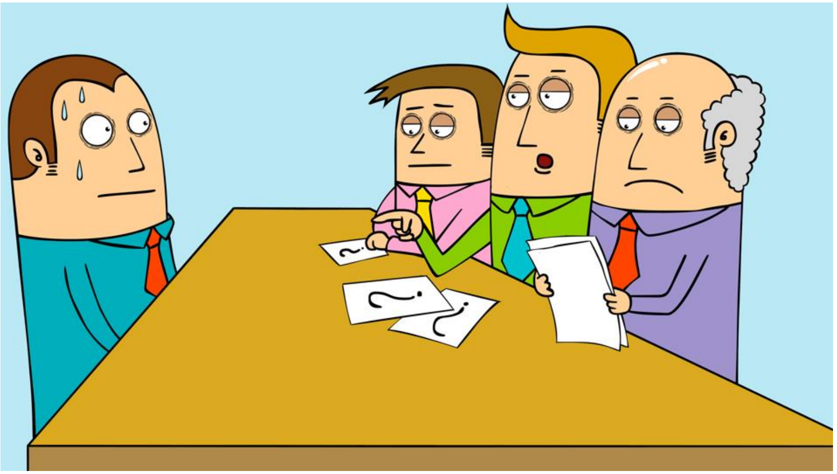
Requirements Elicitation Techniques

- Interview
  - Same Time, Same Place
  - Few People, Analyst-Driven
- Questionnaire/Survey
  - Different Time, Different Place
  - Many People, Analyst-Observer
- Focus Group
  - Same Time, Same or Different Place
  - <20 People, Analyst-Facilitated
- Observation/Ethnography
  - Same Time, Same Place
  - Analyst-Observer
- Document Review
- Brainstorming
- Prototyping/Mock-up

"To cut costs, we got rid of our business analysts and replaced them with self-serve user requirements kiosks."

Interviewing

- ✧ Formal or informal interviews with stakeholders are part of most RE processes.
- ✧ Types of interview
  - ▪ Closed interviews based on pre-determined list of questions.
  - ▪ Open interviews where various issues are explored with stakeholders.



Effective Interviewing

- Identify the interviewees in advance.
- Normally a mix of closed and open-ended interviewing.
- 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, rather than simply asking them what they want.
- The location of the interview should be predefined.
- The time limit should be described.

Interviewing Tips

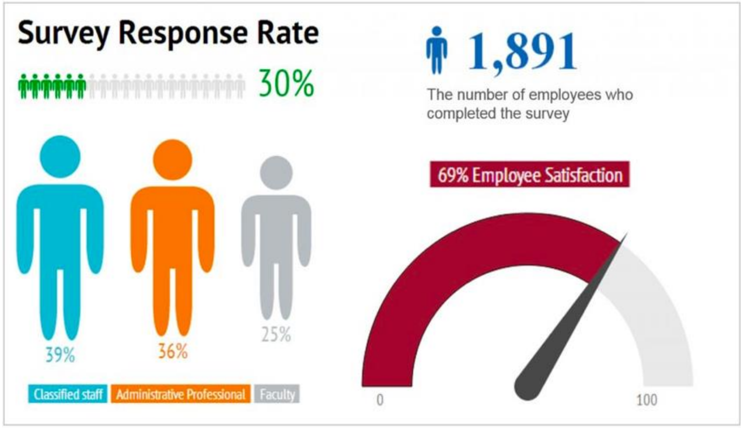
- Starting off: begin the interview with an innocuous topic to set people at ease.
  - e.g. comment on an object on the person’s desk: “What a beautiful photograph! Did you take that?”
- Ask if you can record the interview, but put tape recorder in front of person, say that they can turn it off any time.
- Ask easy questions first, perhaps personal information.
  - e.g. “How long have you worked in your present position?”
- Follow up interesting leads.
  - e.g. “Could we pursue what you just said a little further?”
- Ask open-ended questions last.
  - e.g. “Is there anything else you would like to add?”

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. Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.
- Large amount of qualitative data can be hard to analyze.
- Hard to compare different respondents.
- Interviewing is a difficult skill to master.

Questionnaire/Survey

- ✧ A set of questions is given to stakeholders to quantify their thoughts (attitudes, beliefs, characteristics).
- ✧ When do you use questionnaires?
  - Large Base of Individuals
  - Need Answers to Well-Defined Specific Issues
  - To Verify Results of Limited Interviews
  - When You Want a Specific Outcome



Appear scientific due to statistical analysis.

Questionnaire/Survey

✧ Advantages

- Easy to get data from a large audience
- Can be administered remotely
- Less time is required for the participants to respond
- Can get more accurate information as compared to interviews

✧ Disadvantages

- Simplistic (presupposed) categories provide very little context
- All the stakeholders might not participate in the surveys
- Questions may not be clear to all the participants
- Open-ended questions require more analysis

Questionnaire/Survey

- Watch for
  - Bias in sample selection
  - Small sample size (lack of statistical significance)
  - Open ended questions (very hard to analyze)
  - Leading questions (e.g. have you stopped beating your wife?)
  - Ambiguous questions

Focus Group

- A discussion during which feedback is collected on a specific subject.
- Gather (3 to 20) stakeholders in one room.
- Everybody shares ideas out loud.
- When conduct group session?
  - When many people each knows a (small) part of the whole
  - When problem needs interaction to optimize solution
  - When you can get them all together
  - Anonymity necessary? Use a Tool
  - Distributed? Use a Tool

Focus Group

✧ Advantages

- More natural interaction between people than formal interview
- Can gauge reaction to stimulus materials (e.g. mock-ups)

✧ Disadvantages

- May create unnatural groups (uncomfortable for participants)
- Danger of groupthink
- Requires a highly trained facilitator

✧ Watch for

- Sample bias
- Dominance and submission

The Keys to Successful Focus Groups
Participants contribute equally
?
Participants feel comfortable to interact openly

Observation/Ethnography

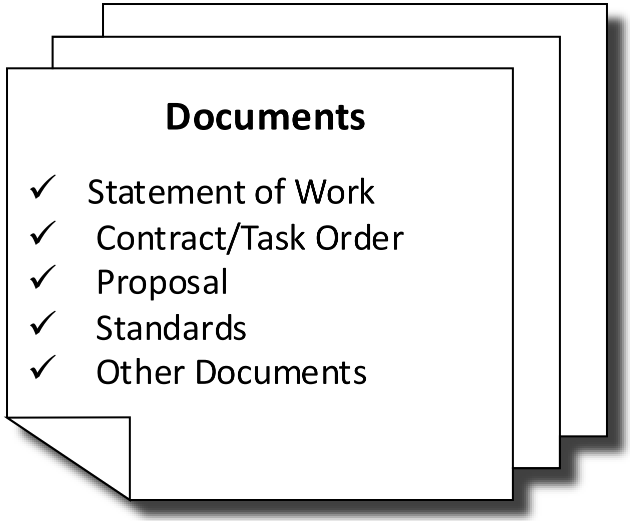
- A social scientist spends a considerable time observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Scope of Observation/Ethnography

- Requirements derived from the way in which people actually work, rather than the way in which business process definitions say they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.
  - Awareness of what other people are doing leads to changes in the ways in which we do things.
- Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Document Review

- ✧ Gather information by reviewing/examining the available materials that describe the business environment.
- ✧ Document analysis includes reviewing the business plans, technical documents, problem reports, existing requirement documents, etc.
- ✧ Purpose
  - ▪ Understand what needs to be done
  - ▪ Use as basis for further study, business process definition and interviews



Documents

- ✓ Statement of Work
- ✓ Contract/Task Order
- ✓ Proposal
- ✓ Standards
- ✓ Other Documents

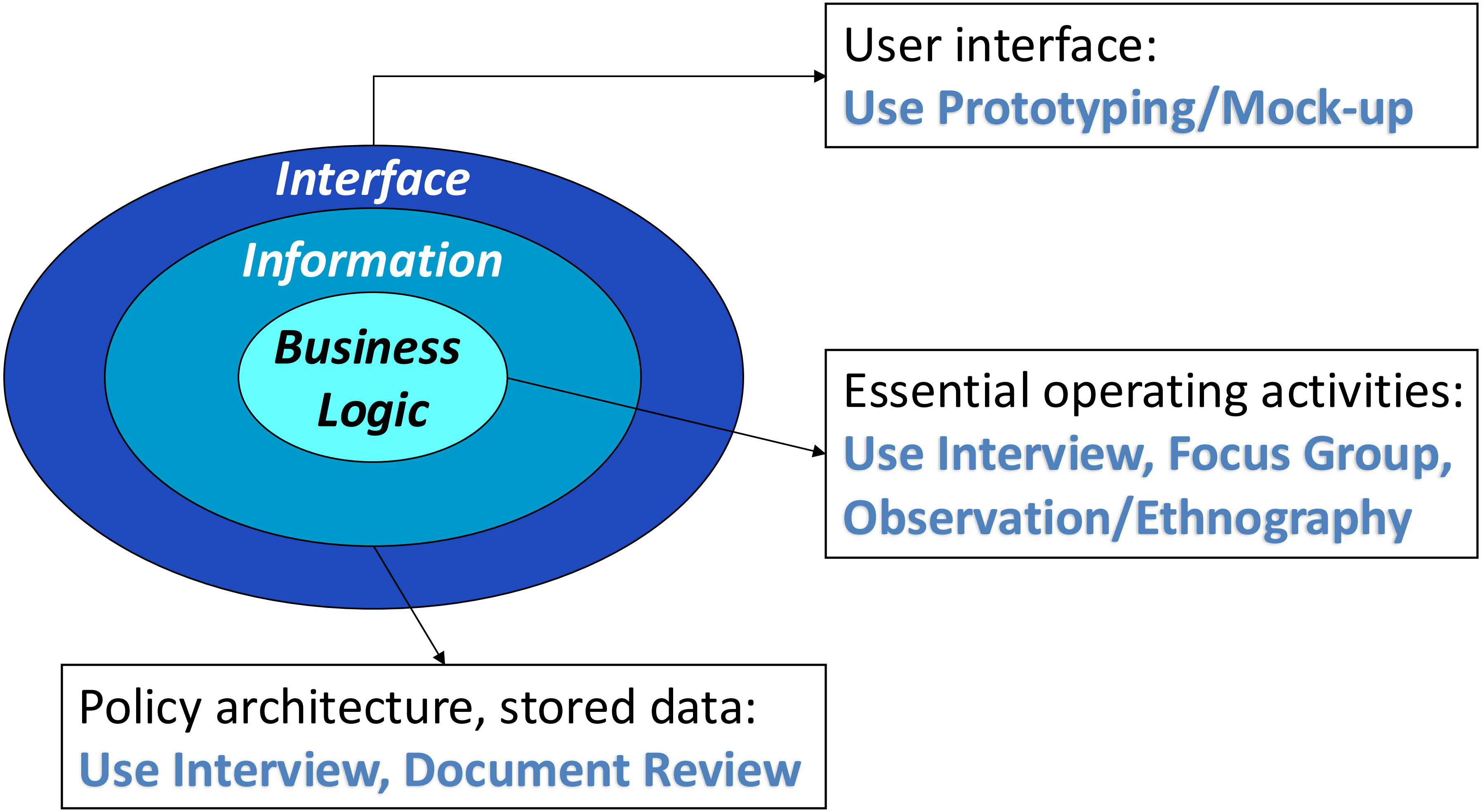
Brainstorming

- ✧ Purpose
  - Use the group effect to generate new ideas for the product/system
  - Useful when there are many unstated or derived requirements
- ✧ Guidelines
  - Use in a facilitated workshop
  - All ideas are good: do not evaluate, debate or criticize
  - Do not be bounded by what is possible
  - Attempt to produce lots of ideas (novel ideas)
  - Use random words to seed the session
  - Piggyback on others’ ideas

Prototyping/Mock-up

- Purpose
  - Clarify requirements that are ambiguous or uncertain
  - Simplify requirements documentation and acceptance needs
  - Provide early feedback to customer and end user
- Guidelines
  - A useful communication tool
  - Use to validate requirements
  - Use to assess alternative user interfaces
  - Help users visualize essential functionality

Technique Selection



User interface:
Use Prototyping/Mock-up

Essential operating activities:
Use Interview, Focus Group, Observation/Ethnography

Policy architecture, stored data:
Use Interview, Document Review

Requirements Specification

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.
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.

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.

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.)*

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.

Structured Specifications

- ✧ A way of writing system requirements where requirements are written in a standard way rather than as free-form text.
- ✧ This approach maintains most of the expressiveness and understandability of natural language but ensures that some uniformity is imposed on the specification.
- ✧ 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.

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.

A Structured Specification of a Requirement for an Insulin Pump

Insulin Pump/Control Software/SRS/3.3.2	
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 of Computation for an Insulin Pump

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

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

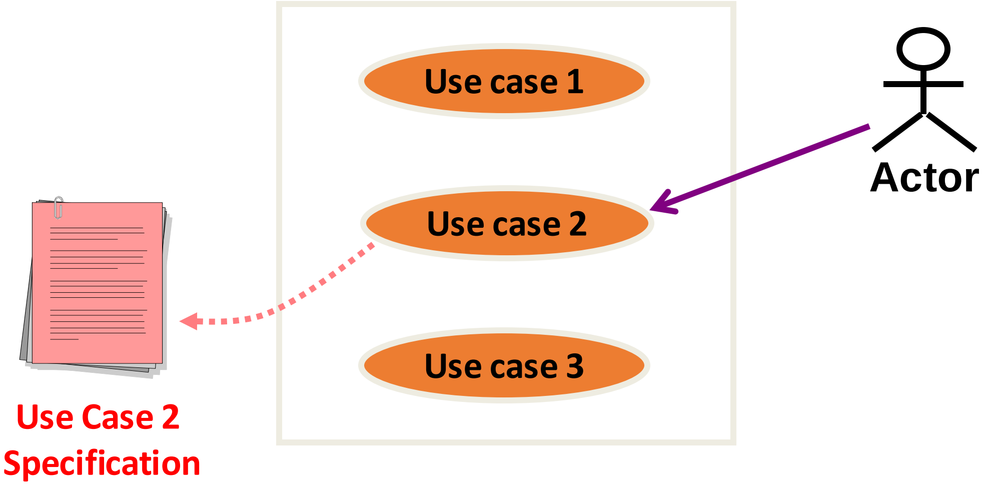
Sugar level increasing and rate of increase stable or increasing
((r2 – r1) ≥ (r1 – r0))
	CompDose = round ((r2 – r1)/4)
	If rounded result = 0 then
	CompDose = MinimumDose

Use Cases

- ✧ Use cases are a way of describing interactions between users and a system using a graphical model and structured text. A textual description or several graphical models such as the UML sequence chart can be added as additional information.
- ✧ Each use case:
  - Shows the system functionality an actor uses.
  - Models a dialog between the system and actors.
  - Defines a sequence of actions performed by a system that yields an observable result of value to an actor.

Use Cases

- Defines clear boundaries of a system.
- Captures the desired behavior of the system.
- Identifies who or what interacts with the system.



Actor

Someone/something outside the system, acting in a role that interacts with the system

Use case

Represents something of value that the system does for its actors

Identify Actors

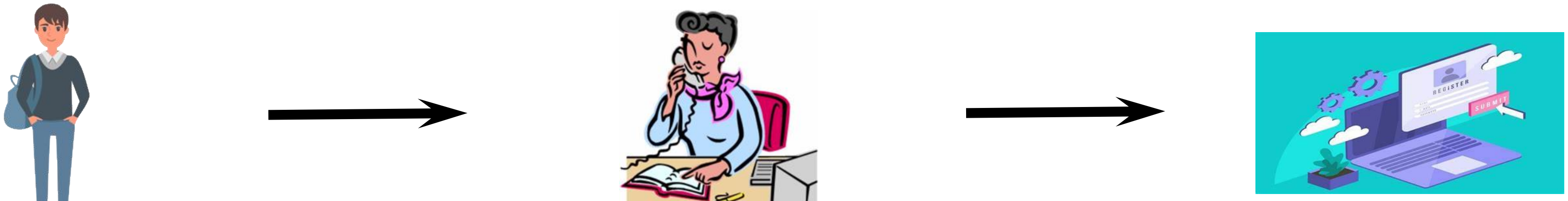
✧ Who/what uses the system?

✧ Who/what gets information from this system?

✧ Who/what provides information to the system?

✧ Who/what supports and maintains the system?

✧ What other systems use this system?



Student

Registrar

Registration System

Who is pressing the keys (interacting with the system)?

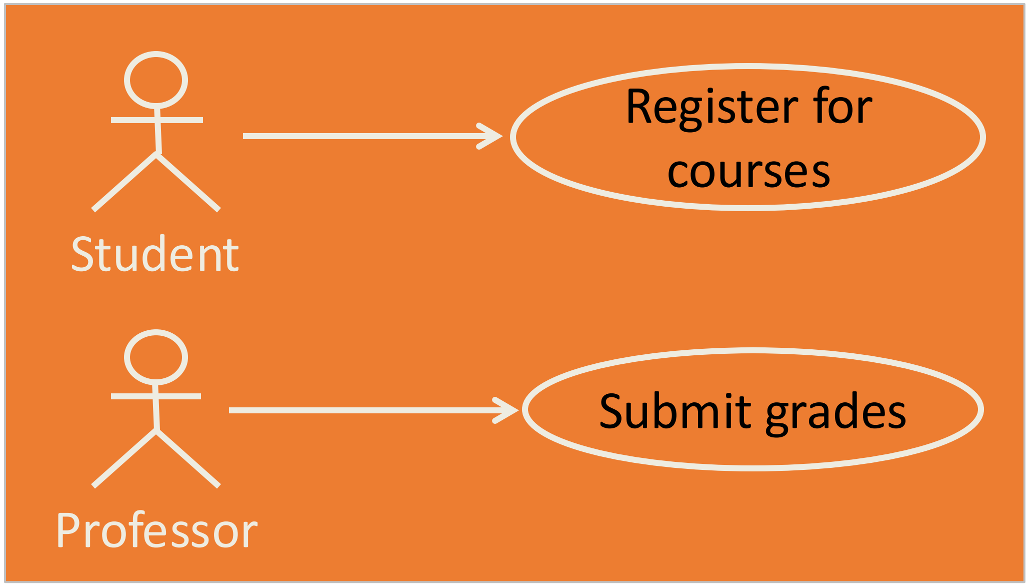
Actors and Roles

Charlie: is employed as a math professor and is an economics undergraduate.

Jodie: is a science undergraduate.

Charlie and Jodie both act as a Student.

Charlie also acts as a Professor.



Checkpoints for Actors

✧ Have you found all the actors? Have you accounted for and modeled all roles in the system's environment?

✧ Is each actor involved with at least one use case?

✧ Can you name at least two people who would be able to perform as a particular actor?

✧ Do any actors play similar roles in relation to the system? If so, merge them into a single actor.

Exercise 1

✧ Stakeholder? Primary actor? Supporting actor? System?

- ATM
- 顾客
- 银行卡
- 银行
- 银行董事
- 打印机
- 服务维护人员
- 银行中央计算机系统
- 银行职员
- 银行抢劫犯

- Primary Actors are those TO WHOM System provides services.

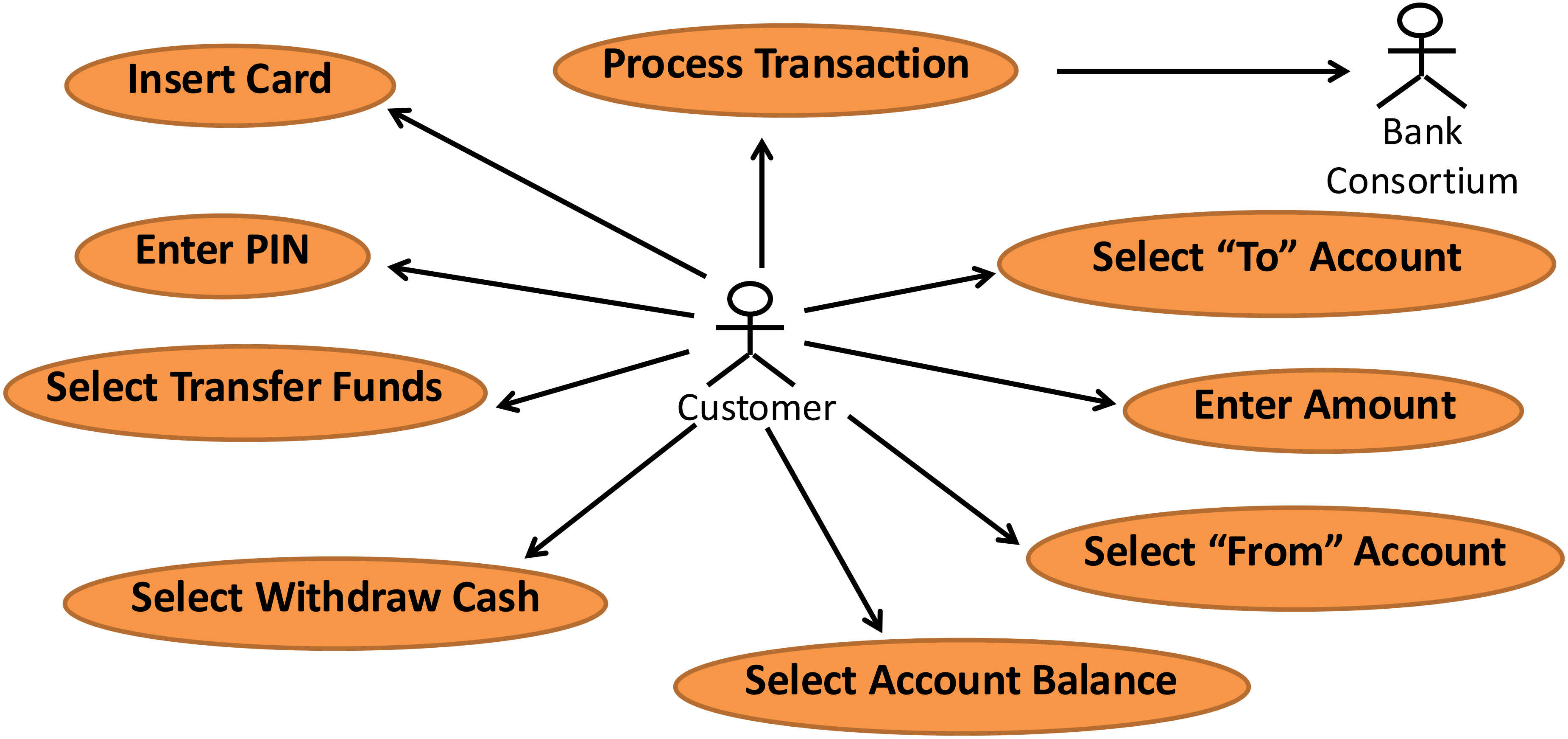
- Supporting Actors are THOSE WHO HELP THE System to provide the services.

Identify Use Cases

✧ What are the goals of each actor?

- Why does the actor want to use the system?
- Will the actor create, store, change, remove, or read data in the system? If so, why?
- Will the actor need to inform the system about external events or changes?
- Will the actor need to be informed about certain occurrences in the system?

Avoid Functional Decomposition

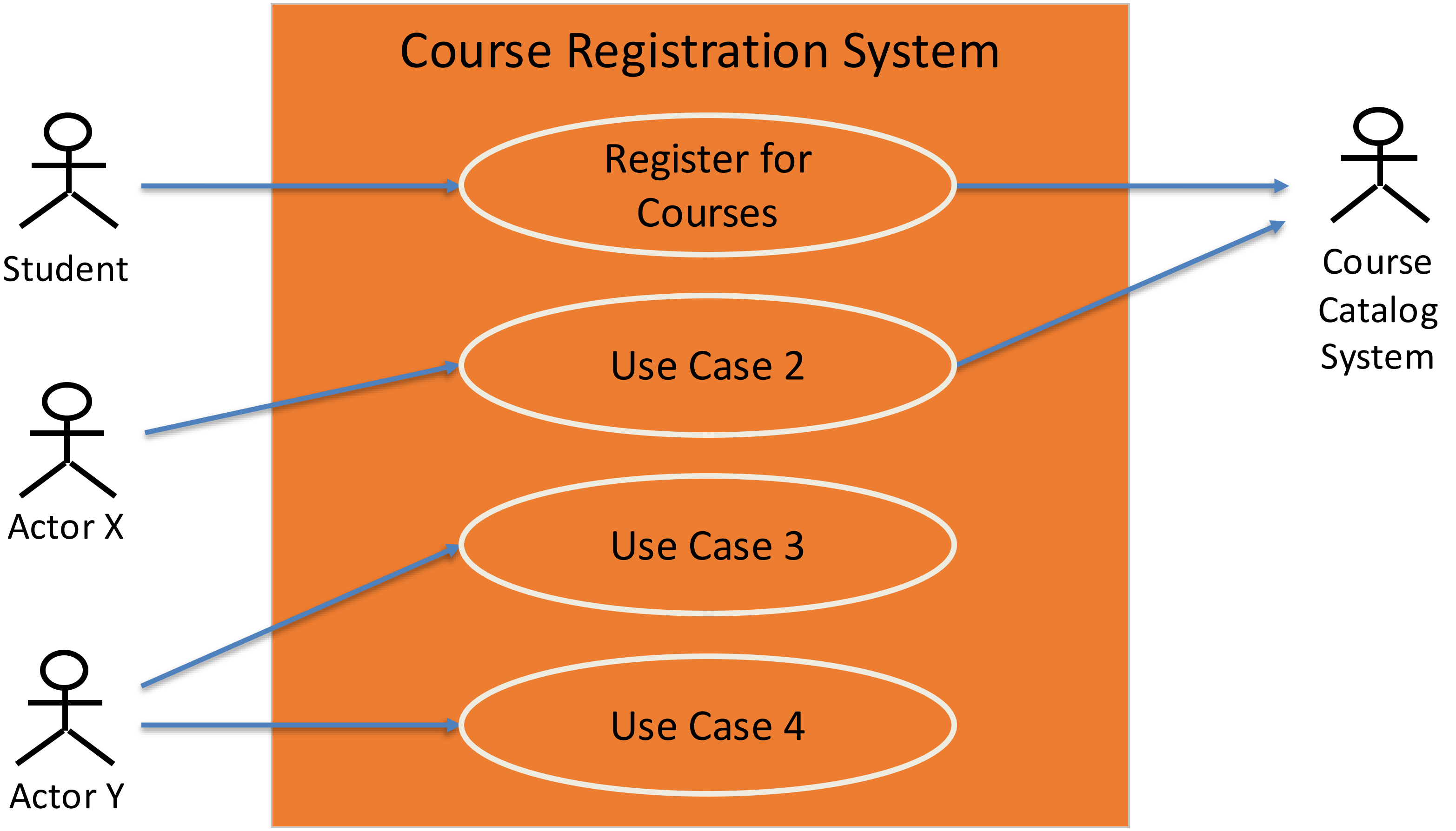


Checkpoints for Use Cases

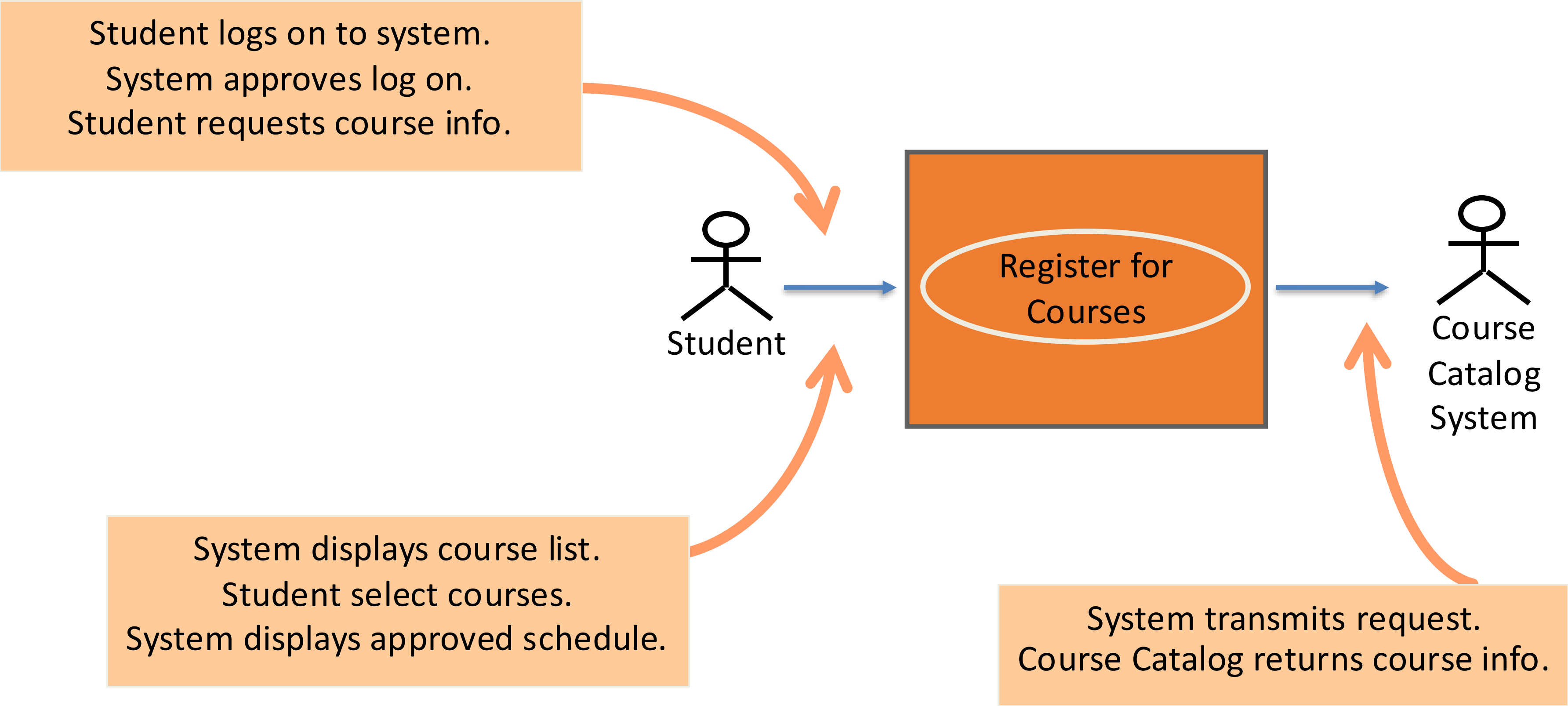
- ✧ The use case model presents the behavior of the system.
  - It is easy to understand what the system does by reviewing the model.
- ✧ All use cases have been identified.
  - The use cases collectively account for all required behavior.
- ✧ All use cases can be justified by tracing them back to a functional requirement.
- ✧ All CRUD use cases have been removed.
  - Create, Retrieve, Update, Delete

Exercise 2

✧ Online course registration system



Each Association is a Whole Dialog

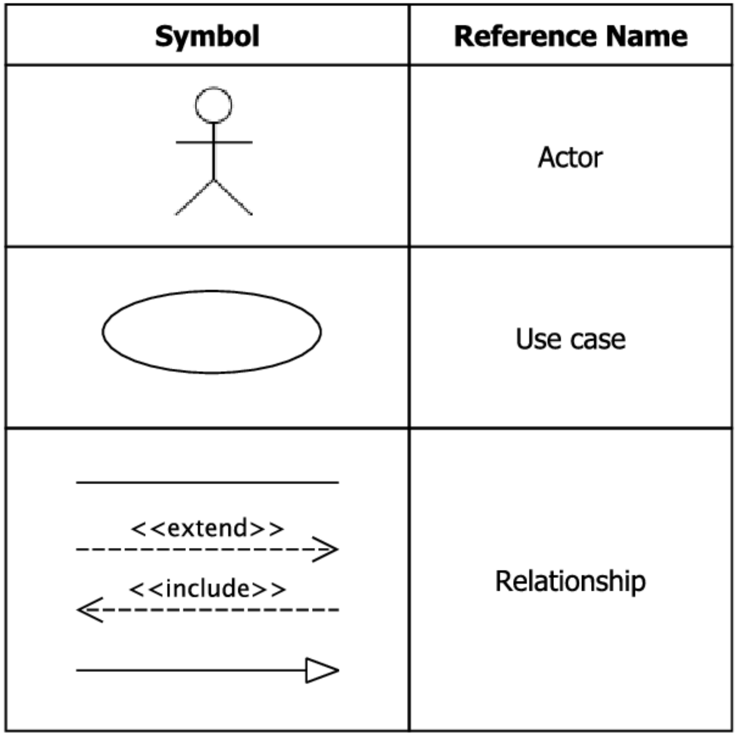


Student logs on to system.
System approves log on.
Student requests course info.

System displays course list.
Student select courses.
System displays approved schedule.

System transmits request.
Course Catalog returns course info.

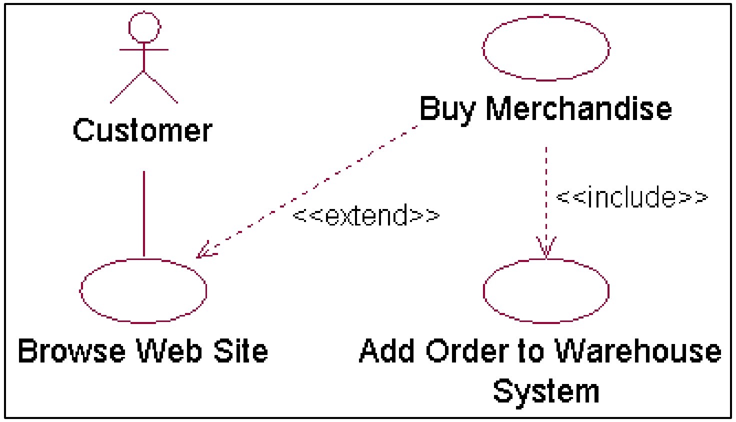
Symbols of Use Case Diagrams

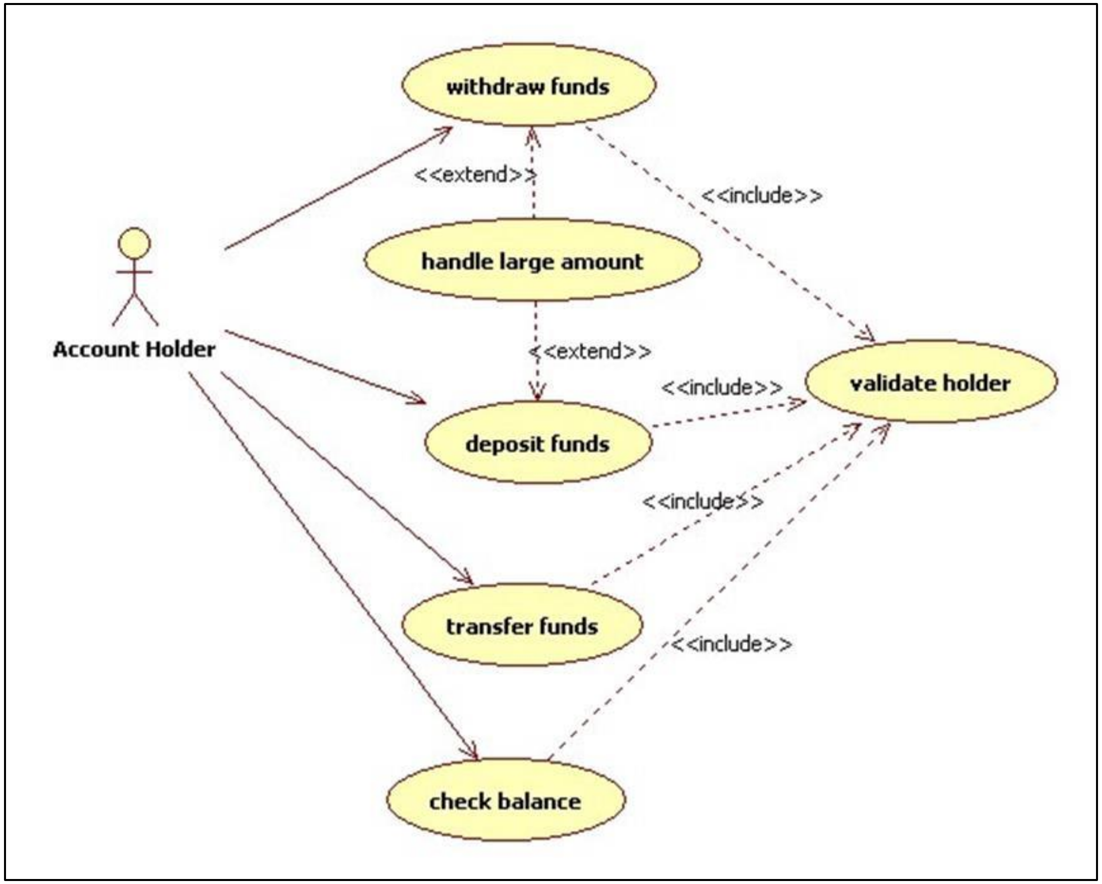


Symbol	Reference Name
	Actor
	Use case
<<extend>> <<include>>	Relationship

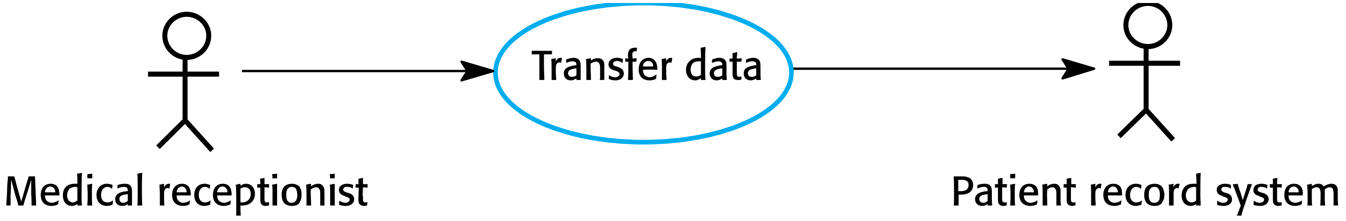
- ✧ When to use <<include>>?
  - You have a piece of behavior that is similar across many use cases
  - Break this out as a separate use case and let the other ones “include” it
- ✧ When to use <<extend>>?
  - A use case is similar to another one but does a little bit more
  - Put the normal behavior in one use case and the exceptional behavior somewhere else

Example of <<include>> and <<extend>>



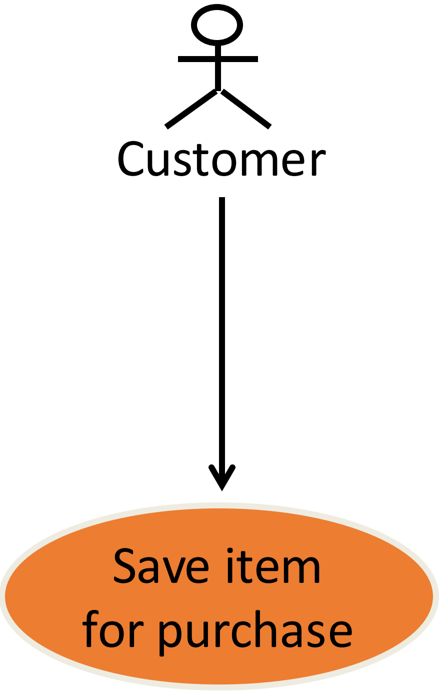


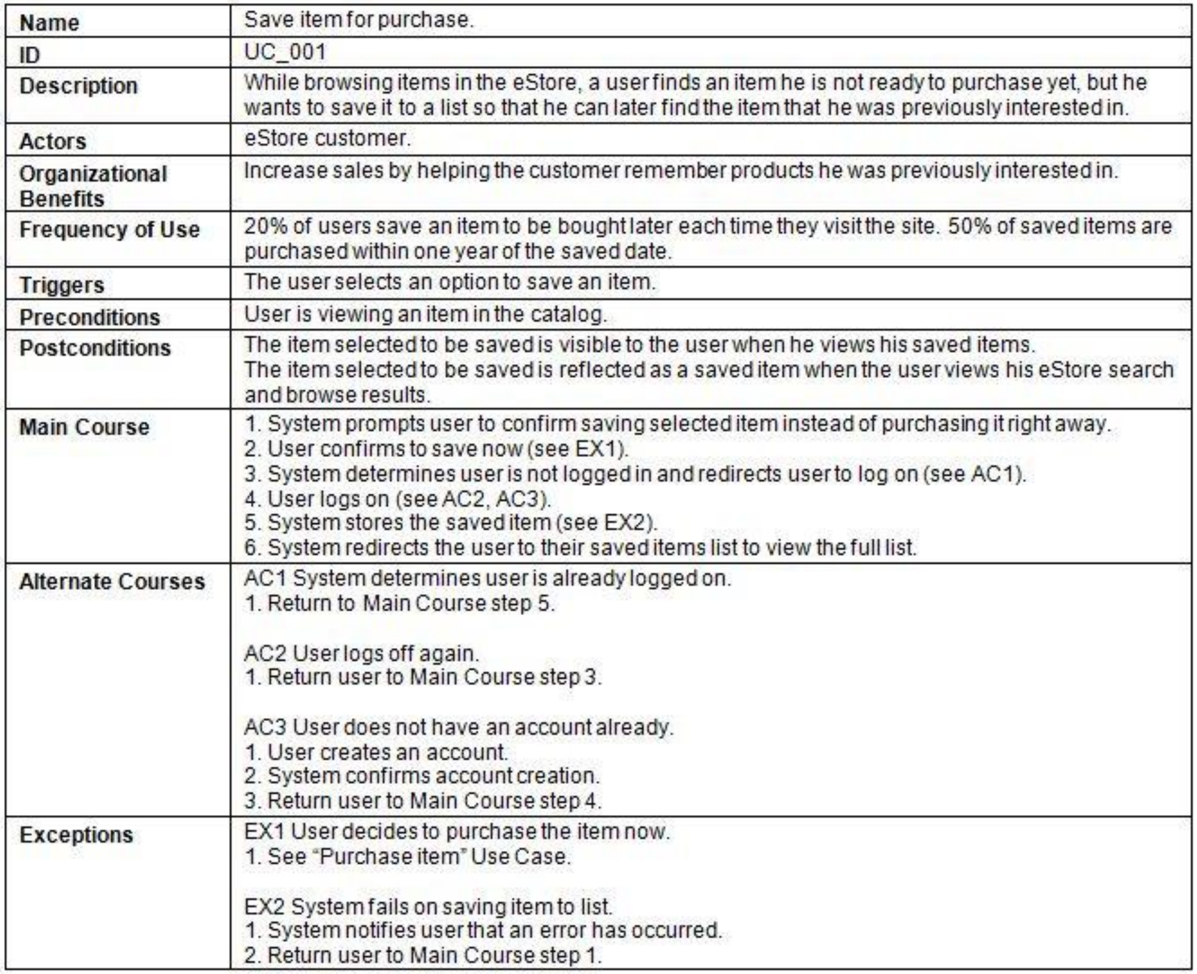
Fully Described Use Cases Example 1



Mentcare system: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Fully Described Use Cases Example 2





Name	Save item for purchase.
ID	UC_001
Description	While browsing items in the eStore, a user finds an item he is not ready to purchase yet, but he wants to save it to a list so that he can later find the item that he was previously interested in.
Actors	eStore customer.
Organizational Benefits	Increase sales by helping the customer remember products he was previously interested in.
Frequency of Use	20% of users save an item to be bought later each time they visit the site. 50% of saved items are purchased within one year of the saved date.
Triggers	The user selects an option to save an item.
Preconditions	User is viewing an item in the catalog.
Postconditions	The item selected to be saved is visible to the user when he views his saved items. The item selected to be saved is reflected as a saved item when the user views his eStore search and browse results.
Main Course	1. System prompts user to confirm saving selected item instead of purchasing it right away. 2. User confirms to save now (see EX1). 3. System determines user is not logged in and redirects user to log on (see AC1). 4. User logs on (see AC2, AC3). 5. System stores the saved item (see EX2). 6. System redirects the user to their saved items list to view the full list.
Alternate Courses	AC1 System determines user is already logged on. 1. Return to Main Course step 5. AC2 User logs off again. 1. Return user to Main Course step 3. AC3 User does not have an account already. 1. User creates an account. 2. System confirms account creation. 3. Return user to Main Course step 4.
Exceptions	EX1 User decides to purchase the item now. 1. See "Purchase item" Use Case. EX2 System fails on saving item to list. 1. System notifies user that an error has occurred. 2. Return user to Main Course step 1.

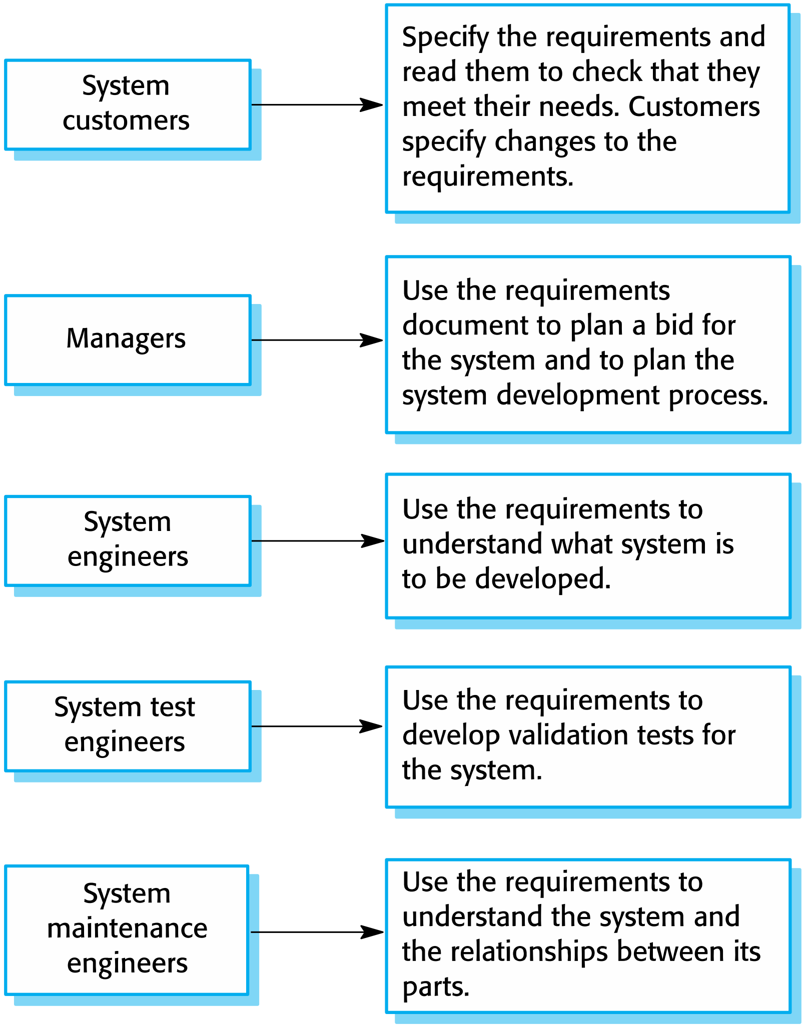
Benefits of Use Cases

- ✧ Give context for requirements.
  - ▪ Put system requirements in logical sequences.
  - ▪ Illustrate why the system is needed.
  - ▪ Help verify that all requirements are captured.
- ✧ Are easy to understand.
  - ▪ Use terminology that customers and users understand.
  - ▪ Verify stakeholder understanding.
- ✧ Facilitate agreement with stakeholder.
- ✧ Facilitate reuse: test, documentation, and design.

Software Requirements Document

- The software requirements document (sometimes called the software requirements specification or SRS) is an official statement of what the system developers should implement.
- 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



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.

- The diversity of possible users means that the requirements document has to be a compromise.
- It has to describe the requirements for customers, define the requirements in precise detail for developers and testers, as well as include information about future system evolution.

Requirements Document Variability

- ✧ The level of detail that you should include in a requirements document depends on the type of system that is being developed and the development process used.
  - ▪ Critical systems
  - ▪ Outsourcing
  - ▪ In-house, iterative development process
- ✧ Requirements documents standards have been designed e.g. IEEE 830-1998 standard. This standard is a generic one that can be adapted to specific uses.
  - ▪ These are mostly applicable to the requirements for large systems engineering projects

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.

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.

IEEE 830 and User Stories

- IEEE 830-style requirements statements focus on the attributes of the solution, while user stories focus on the user’s goals.
- IEEE 830 requirements specifications encourage teams to write all of the requirements statements up front rather than in an iterative manner, as with user stories.

- The product shall have a gasoline-powered engine.
- The product shall have four wheels.
- The product shall have a rubber tire mounted to each wheel.
- The product shall have a steering wheel.
- The product shall have a steel body.

- The product makes it easy and fast for me to mow my lawn.
- I am comfortable while using the product.

User Stories and Use Cases

- ✧ A user story usually encompasses a smaller scope than a use case. User stories do not include as much detail as use cases.
- ✧ User stories differ from use cases in their longevity.
  - ▪ Use cases are designed to be permanent artifacts of the development process; user stories are more transient and not intended to outlive the iteration in which they are developed.
- ✧ They are written for different purposes.
  - ▪ Use cases are written so that developers and customers can discuss them and agree to them. User stories are written to facilitate release planning and to serve as reminders to fill in requirements details with conversations.

Requirements Validation

Requirements Validation

- Requirements validation is the process of checking that requirements define the system that the customer really wants. It overlaps with elicitation and analysis, as it is concerned with finding problems with the requirements.
- The cost of fixing a requirements problem by making a system change is usually much greater than repairing design or coding errors.
  - A change to the requirements usually means that the system design and implementation must also be changed.

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
  - The reviewers (both client and contractor staff) systematically analyses SRS document to check error and ambiguity.
- ✧ Prototyping
  - The prototype of the system is presented before the end-user or customer, they experiment with the presented model and check if it meets their need. This type of model is generally used to collect feedback about the requirement of the user.
- ✧ Test-case generation
  - Requirement mentioned in SRS document should be testable. It is generally believed that if the test is difficult or impossible to design, this usually means that requirement will be difficult to implement and it should be reconsidered.

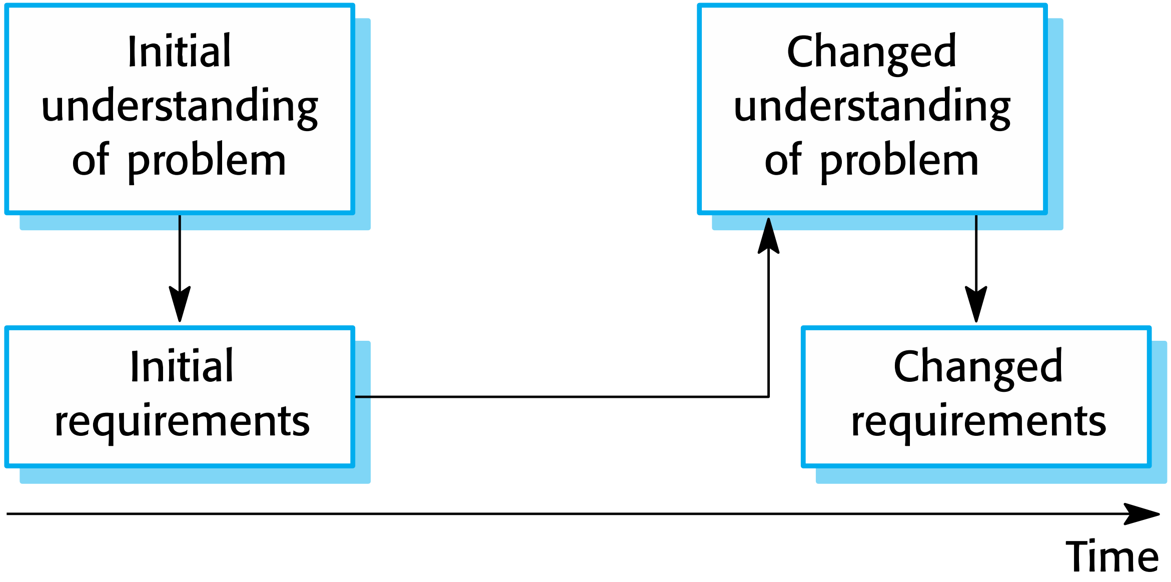
Requirements Change

Changing Requirements

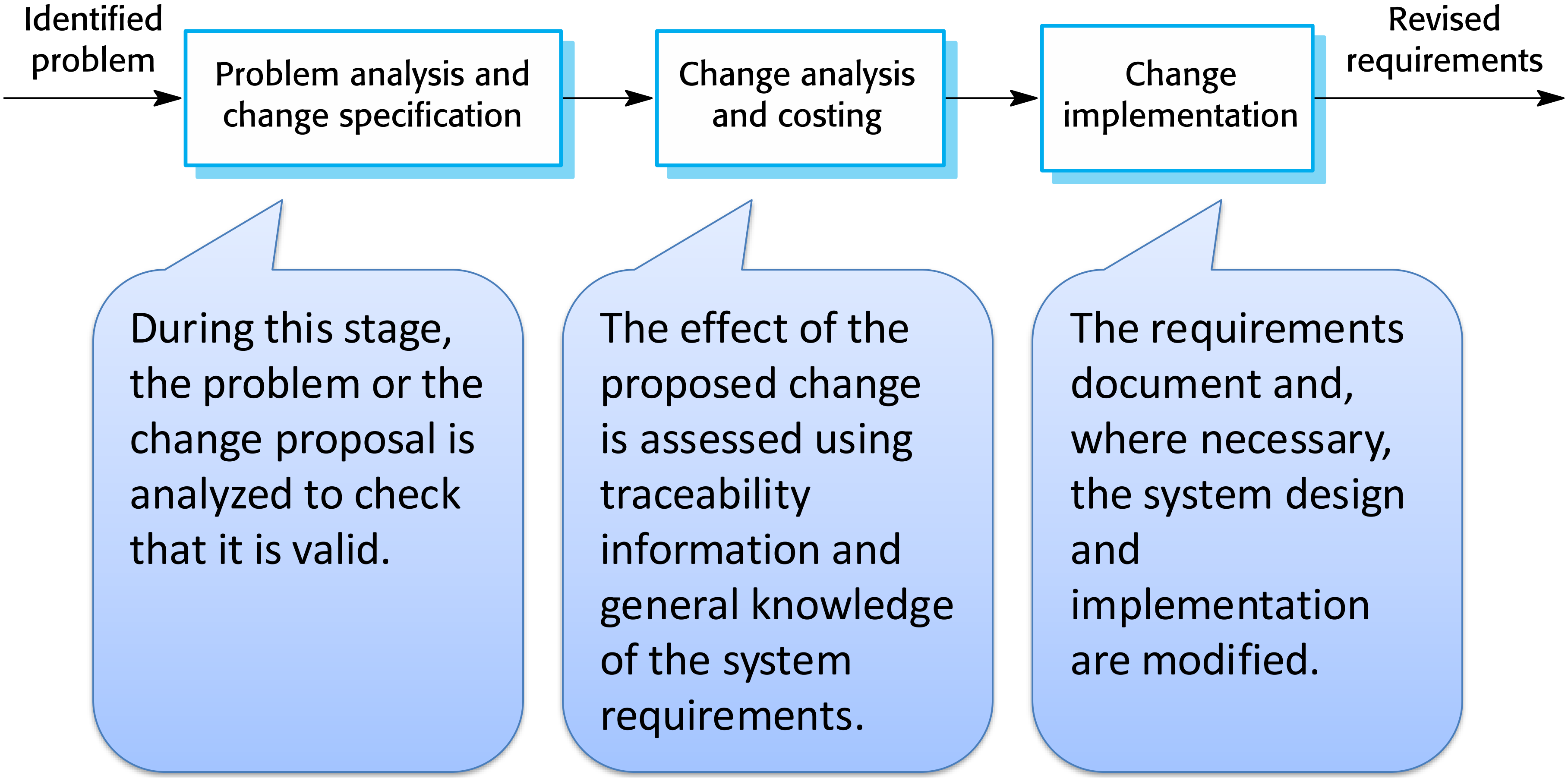
- ✧ The business and technical environment of the system always changes after installation.
  - ▪ New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation 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 requirements because of organizational and budgetary constraints. These may conflict with end-user requirements.

Changing Requirements

- ✧ Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Requirements Change Management



Identified problem

Problem analysis and change specification

Change analysis and costing

Change implementation

Revised requirements

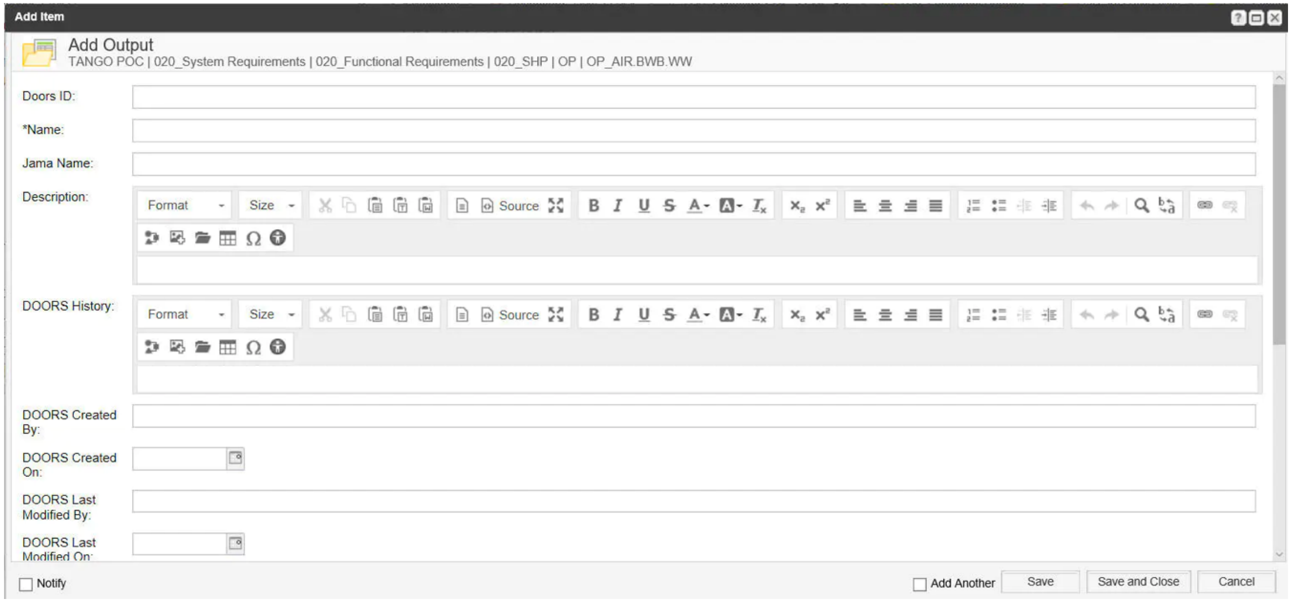
During this stage, the problem or the change proposal is analyzed to check that it is valid.

The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements.

The requirements document and, where necessary, the system design and implementation are modified.

Requirements Management

- ✧ Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- ✧ IBM Rational DOORS, DevCloud, TAPD



Add Item

Add Output

TANGO POC | 020_System Requirements | 020_Functional Requirements | 020_SHP | OP | OP_AIR.BWB.WW

Doors ID:

*Name:

Jama Name:

Description:

DOORS History:

DOORS Created By:

DOORS Created On:

DOORS Last Modified By:

DOORS Last Modified On:

Summary

Key Points

✧ 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.

Key Points

- ✧ 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 prioritisation and negotiation, requirements documentation.
- ✧ Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.

Key Points

- ✧ 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.