Advanced Software Engineering
(CS6401)

Autumn Semester (2022-2023)

Dr. Judhistir Mahapatro
Department of Computer Science and Engineering
National Institute of Technology Rourkela

Requirement Analysis and Specification

Objectives

- To introduce the concepts of user and system requirements
- To describe functional and non-functional requirements
- To explain how software requirements may be organized in a requirements document

Requirement Engineering

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

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

What is a Requirement?

- A Requirement is:
  - A capability or condition required from the system.
- What is involved in requirements analysis and specification?
  - Determine what is expected by the client from the system. (Gather and Analyze)
  - Document those in a form that is clear to the client as well as to the development team members. (Document)

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; (there must not be room left for different interpretation)
  - May be the basis for the contract itself – therefore must be defined in detail;

Understanding and specifying requirements

- **For toy problems:** understanding and specifying requirements is rather easy…
- **For industry-standard problems:** Probably the hardest, most problematic and error-prone among development tasks.
- **The task of requirements specification :**
  - **Input**: User needs that are hopefully fully understood by the users.
  - **Output**: Precise statement of what the software will do.

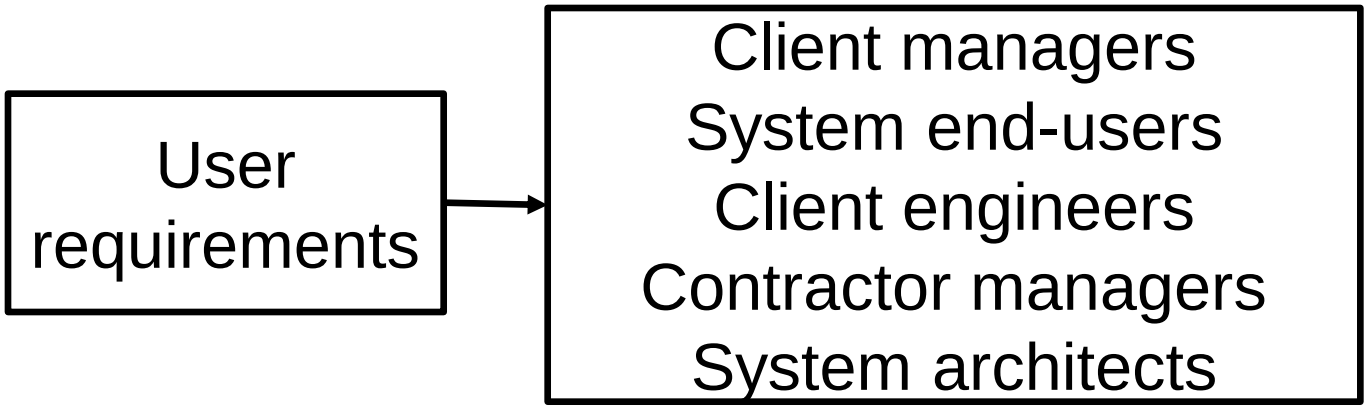
Types of requirement

- User requirement
  - 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.

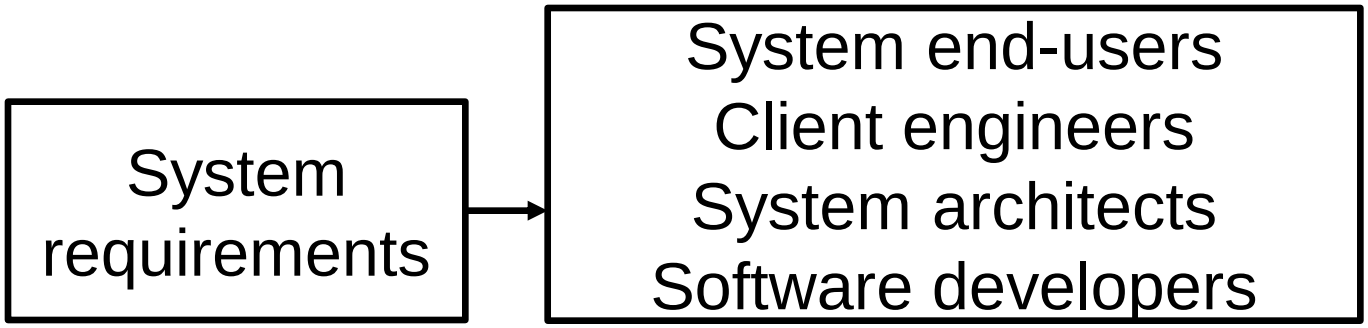
Definitions and specifications

- User Requirements Definition
  - The software must provide a means of accessing external files created by other tools
- System Requirements Specification
  - The user should have a facility to define the type of the external file.
  - Each external file should have an associated tool which can be applied to view the file
  - Each external file must be represented as an icon on the desktop.
  - When the user selects an icon representing an external file, the effect must be apply the pre-defined tool and open the file.

Requirements Audience

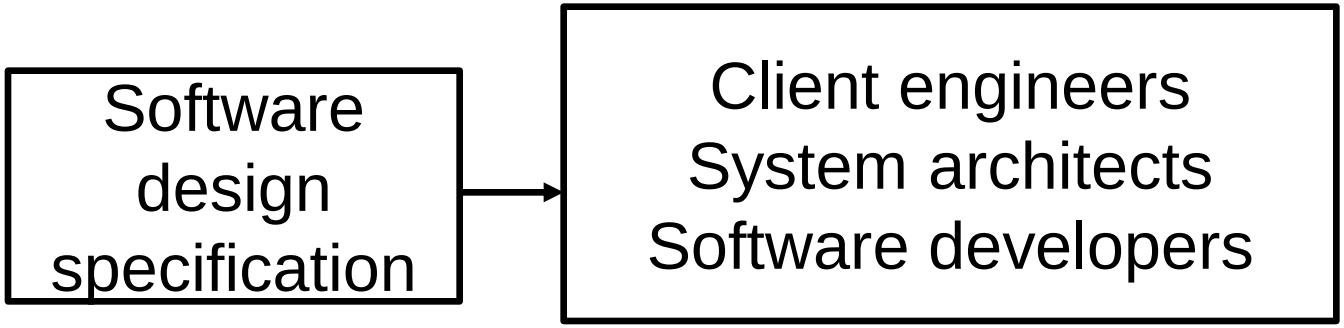


Requirements Audience (Cont..)



- It is not written by the user, not for the user but written for translating the user requirements into more precise specification.
- This translation is often done by System Architect or Business Analyst.
- It is very closer to the Software design specification

Requirements Audience (Cont..)



- Software design specification is the area defining How to build the system rather what the system expected to do

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.
  - For example, try to open a file but it does not exists so how the system reacts to such a situation

- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. (its everything, ..catch-all bucket to put in requirements that don't fit into the functional structure of the system)

Functional requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.
- The LIBSYS system
  - A library system that provides a single interface to a number of databases of articles in different libraries.
  - Users can search for, download and print these articles for personal study

Examples of functional requirements

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

Requirements imprecision

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

Requirements completeness and consistency

- In principle, requirements should be both complete and consistent.
- Complete
  - They should include descriptions of all facilities required. (specifying the requirements that what the systems exactly do and it should also specify along the same line what the system is not to do)
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document.

Non-functional requirements

- 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 CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Example: Trading System – Real Time Systems

Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g., execution speed, reliability, etc.
- Organizational requirements
  - Requirements that are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc. (ISO 9001)
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements (confirm to certain standard guidelines and rules, example, Military applications) etc. (quality guidelines)

Non-functional requirements examples

- Product requirement

  1.1 The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.

- Organizational requirement

  2.1 The system development process and deliverable documents shall conform to the process and deliverables defined in ISO 9001.

- External requirement

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

Goals and requirements

- Non-functional requirements 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.
  - User should be able to learn the system within two days of learning time and after that the user can process the (e.g., Insurance Claim) less than 30 min of time
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

Examples

- A system goal
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.

- A verifiable non-functional requirement
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

User requirements

- Should describe functional and non-functional requirements in such a way that they are understandable by system users who don’t have detailed technical knowledge.

- User requirements are defined using natural language, tables and diagrams as these can be understood by all users.

Problems with natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up
- Requirements amalgamation
  - Several different requirements may be expressed together.

LIBSYS requirement

- LIBSYS shall provide a financial accounting system that maintains records of all payments made by users of the system. System managers may configure this system so that regular users may receive discounted rates.
  - Two requirements mixing up, first one is quit general and second one is specific.

Requirement problems

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

Guidelines for writing requirements

- Invent a standard format and use it for all requirements
- Use language 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. (user may not understand certain computer terminology like frame or plain text format in webpage)

System requirements

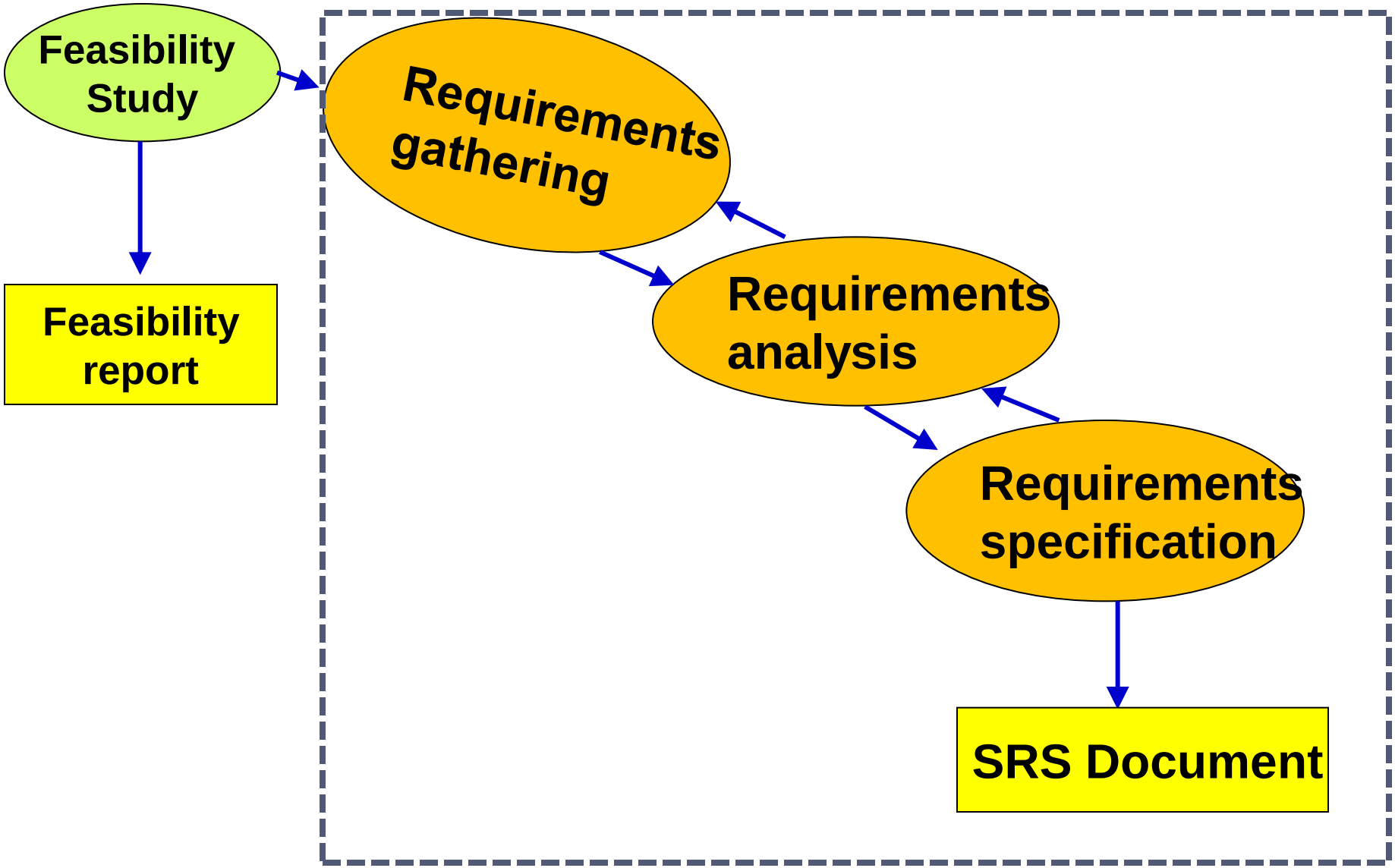
- More detailed specifications of the system functions, services and constraints than user requirements.
- They are intended to be a basis for designing the system. (contract between client and developer)
- They may be incorporated into the system.

Requirements and Design

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

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

Requirements Engineering Process



Requirements Analysis and Specification

- Requirements Gathering:
  - Fully understand the user requirements.
- Requirements Analysis:
  - Remove inconsistencies, anomalies, etc. from requirements.
- Requirements Specification:
  - Document requirements properly in an SRS document.

Requirements Analysis and Specification

- **Aim of this phase:**
  - understand the exact requirements of the customer,
  - document them properly.

- Consists of two distinct activities:
  - requirements gathering and analysis
  - requirements specification.

Goals of Requirements Analysis

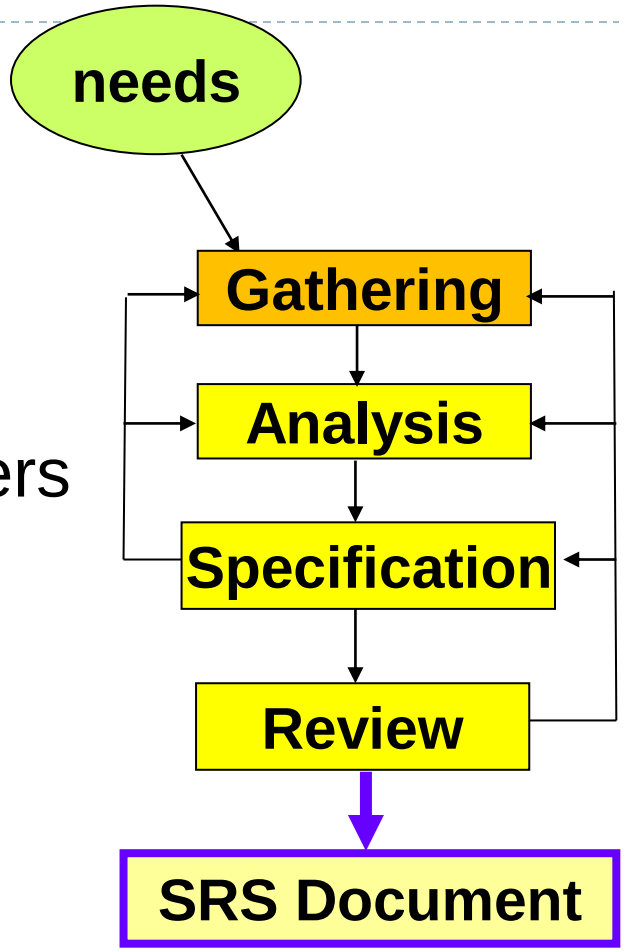
- Collect all related data from the customer:
  - Analyse the collected data to clearly understand what the customer wants,
  - Find out any inconsistencies and incompleteness in the requirements,
  - Resolve all inconsistencies (requirements contradicts) and incompleteness (parts of requirement omitted).

Requirements Gathering

- Gathering relevant data:
  - usually collected from the end-users through interviews and discussions.
  - For example, for a business accounting software:
    - interview all the accountants of the organization to find out their requirements.

How to Gather Requirements?

- Observe existing (manual) systems
- Study existing procedures
- Discuss with customer and end-users
- Input and Output analysis
- Analyse what needs to be done



Requirements Gathering Activities

- 1. Study existing documentation
- 2. Interview
- 3. Task analysis
- 4. Scenario analysis
- 5. Form analysis

Requirements Gathering (CONT.)

- In the absence of a working system,
  - Lot of imagination and creativity are required.
- Interacting with the customer to gather relevant data:
  - Requires a lot of experience.

Requirements Gathering (CONT.)

- Some desirable attributes of a good requirements analyst:
  - Good interaction skills,
  - Imagination and creativity,
  - Experience…

Case Study: Automation of Office Work at CSE Dept.

- The academic, inventory, and financial information at the CSE department:
  - At present carried though manual processing by two office clerks, a store keeper, and two attendants.
- Considering the low budget he had at his disposal:
  - The HoD entrusted the work to a team of student volunteers.

Case Study: Automation of Office Work at CSE Dept.

Interview

- The team was first briefed by the HoD:
  - Concerning the specific activities to be automated.
- The analysts first discussed with the two office clerks:
  - Regarding their specific responsibilities (tasks) that were to be automated.
- The analyst also interviewed student and faculty representatives who would also use the software.

Case Study: Automation of Office Work at CSE Dept.

Task and Scenario Analysis

- For each task that a user needs the software to perform, they asked:
  - The steps through which these are to be performed.
  - The various scenarios that might arise for each task.

Form Analysis

- Also collected the different types of forms that were being used.

Case Study: Automation of Office Work at CSE Dept.

Requirements Analysis

- The analysts understood the requirements for the system from various user groups:
  - Identified inconsistencies, ambiguities, incompleteness.
- Resolved the requirements problems through discussions with users:
  - Resolved a few issues which the users were unable to resolve through discussion with the HoD.

Case Study: Automation of Office Work at CSE Dept.

Requirements Specification

- Documented the requirements in the form of an SRS document.

Analysis of the Gathered Requirements (CONT.)

- Objective: determine what the system must do to solve the problem (without describing how)
- Done by Analyst (also called Requirements Analyst)
- Produce Software Requirement Specifications (SRS) document
- Incorrect, incomplete, inconsistent, ambiguous SRS often cause for project failures and disputes

Analysis of the Gathered Requirements (CONT.)

- A very challenging task
  - Users may not know exactly what is needed or how software can bring further value to what is being done today
  - Users change their mind over time
  - They may have conflicting demands
  - They can’t differentiate between what is possible and cost-effective against that is impractical (wish-list)
  - Analyst has no or limited domain knowledge
  - Often client is different from the users

Analysis of Gathered Requirements

- Main purpose of requirement analysis:
  - Clearly understand user requirements,
  - Detect inconsistencies, ambiguities, and incompleteness.
- Incompleteness and inconsistencies:
  - Resolved through further discussions with the end-users and the customers.

Ambiguity

“When temperature becomes high, start cooler”

Do you notice any problems?

- Above what threshold we consider the temperature to be high?

Inconsistent Requirement

- Some part of the requirement:
  - contradicts some other requirement.
- **Example:**
  - One customer says turn off heater and open water shower when temperature > 100°C
  - Another customer says turn off heater and turn ON cooler when temperature > 100°C

Incomplete Requirement

- Some requirements not included:
  - Possibly due to oversight.
- **Example:**
  - The analyst has not recorded that when temperature falls below 90° C :
    - heater should be turned ON
    - water shower turned OFF.

Analysis of the Gathered Requirements

- Requirements analysis involves:
  - Obtaining a clear, in-depth understanding of the software to be developed
  - Remove all ambiguities and inconsistencies from the initial customer perception of the problem.

Analysis of the Gathered Requirements (CONT.)

- It is quite difficult to obtain:
  - A clear, in-depth understanding of the problem
    - Especially if there is no working model of the problem.

Analysis of the Gathered Requirements (CONT.)

- Experienced analysts take considerable time:
  - Clearly understand the exact requirements the customer has in his mind.

Analysis of the Gathered Requirements (CONT.)

- Experienced systems analysts know - often as a result of painful experiences ---
  - “Without a clear understanding of the problem, it is impossible to develop a satisfactory system.”

Analysis of the Gathered Requirements

- Several things about the project should be clearly understood:
  - What is the problem?
  - What are the possible solutions to the problem?
  - What complexities might arise while solving the problem?

Analysis of the Gathered Requirements

- Some anomalies and inconsistencies can be very subtle:
  - Escape even most experienced eyes.
  - If a formal specification of the system is constructed,
    - Many of the subtle anomalies and inconsistencies get detected.

Analysis of the Gathered Requirements (CONT.)

- After collecting all data regarding the system to be developed,
  - Remove all inconsistencies and anomalies from the requirements,
  - Systematically organize requirements into a Software Requirements Specification (SRS) document.

Analysis of the Gathered Requirements (CONT.)

- The data you initially collect from the users:
  - would usually contain several contradictions and ambiguities
    - each user typically has only a partial and incomplete view of the system.

Analysis of the Gathered Requirements (CONT.)

- Ambiguities and contradictions:
  - must be identified
  - resolved by discussions with the customers.
- Next, requirements are organized:
  - into a Software Requirements Specification (SRS) document.

Analysis Process

- Interviewing clients and users essential to understand their needs from the system

- Often existing documents and current mode of operations can be studied (they may have documents explaining procedures)

Analysis Process (Cont..)

- Long process: needs to be organized systematically
  - Interviewing, correlating, Identifying gaps, and iterating again for more details
  - Focus on what gets done or needs to be done
  - Focus on business entities, their interactions, business events, …
- Identify users and important business entities
- Get functional (domain) knowledge

Analysis Process (Cont..)

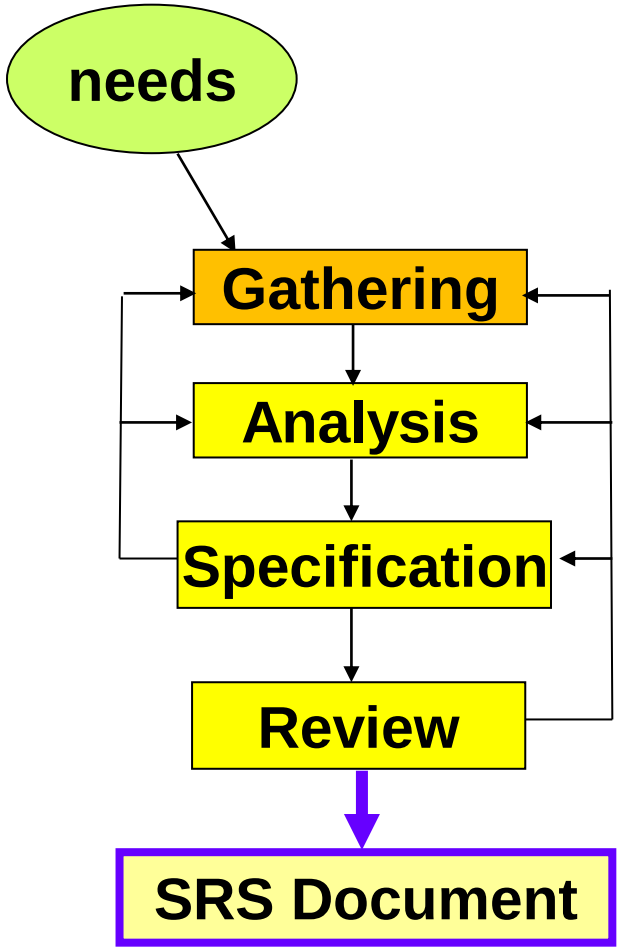
- Interview users or get details through questionnaires
- Examine existing system: study existing forms, outputs, records kept (files, ledgers, computerized systems)
- Often goes outside in: what outputs needed, which inputs provide data, what processing done, what records kept, how records updated, (i.e., go inwards from system boundaries)

Interviews

- Identify users, their roles and plan interviews in proper order to collect details progressively and systematically
- Conducting interviews is an art
  - Workout scope, durations, purpose
  - Keep records and verify/confirm details it with the user
  - Needs to sometimes ‘prompt’ users in visualizing requirements
- Need good communication skills, domain knowledge, patience, …

Software Requirements Specification

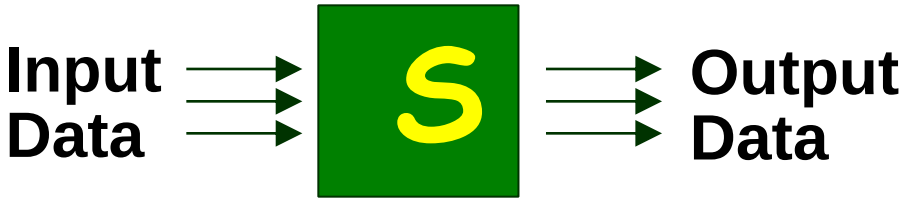
- Main aim:
  - Systematically organize the requirements arrived during requirements analysis.
  - Document requirements properly.



SRS Document

- As already pointed out--- useful in various contexts:
  - Statement of user needs
  - Contract document
  - Reference document
  - Definition for implementation

SRS Document (CONT.)



- SRS document is known as black-box specification:
  - The system is considered as a black box whose internal details are not known.
  - Only its visible external (i.e., input/output) behaviour is documented.

SRS Document (CONT.)

- SRS document concentrates on:
  - What needs to be done in terms of input-output behaviour
  - Carefully avoids the solution (“how to do”) aspects.

SRS Document (CONT.)

- The requirements at this stage:
  - Written using end-user terminology.
- If necessary:
  - Later a formal requirement specification may be developed from it.

SRS Document (CONT.)

- SRS is basis for subsequent design an implementation
- First and most important baseline
  - Defines contract with users
  - Basis for validation and acceptance
- Cost increases rapidly after this step;
  - Defects not captured here become 2 to 25 times more costly to remove later (**defects which entered in the design and implementation which will be more costlier to remove it**)

SRS Document (CONT.)

- It identifies all functional (inputs, outputs, processing) and performance requirements, and also other important constraints (legal, social, operational)
- Should be adequately detailed so that
  - Users can visualize what they will get
  - Design and implementation can be carried out

SRS Document (CONT.)

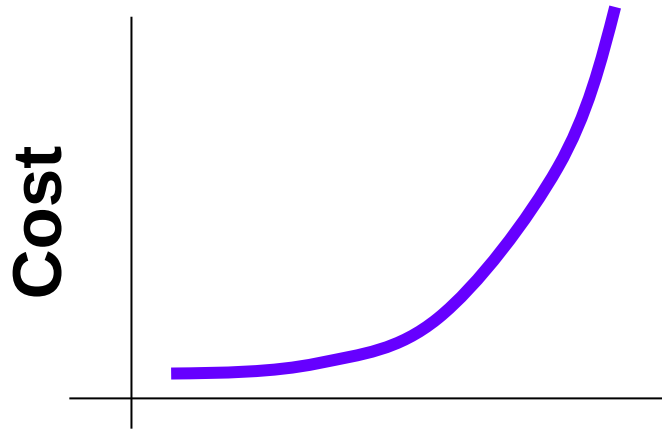
- Covers what and what at business level; e.g.,
  - What calculate take-home pay
  - How: procedure (allowances, deductions, taxes etc.)

Need for SRS...

- **Good SRS reduces development cost:**
  - Requirement errors are expensive to fix later
  - Requirement changes cost a lot (typically 40% of requirements change later)
  - Good SRS can minimize requirement changes and errors
  - Substantial savings --- effort spent during requirement saves multiple times that effort

Need for SRS…

- **An Example:**
  - Cost of fixing errors in requirement, design, coding, acceptance testing and operation increases exponentially



What are the Uses of an SRS Document?

- Establishes the basis for agreement between the customers and the suppliers
- Forms the starting point for development.
- Provide a basis for estimating costs and schedules.
- Provide a basis for validation and verification.
- Provide a basis for user manual preparation.
- Serves as a basis for later enhancements.

Forms A Basis for User Manual

- The SRS serves as the basis for writing User Manual for the software:
  - **User Manual: Describes the functionality from the perspective of a user --- An important document for users.**
  - Typically also describes how to carry out the required tasks with examples.

SRS Document: Stakeholders

- SRS intended for a diverse audience:
  - Customers and users use it for validation, contract, ...
  - Systems (requirements) analysts
  - Developers, programmers to implement the system
  - Testers use it to check whether requirements have been met
  - Project Managers to measure and control the project

SRS Document: Stakeholders

- Different levels of detail and formality is needed for each audience
- Different templates for requirements specifications used by companies:
  - Often variations of **IEEE 830**

Software Requirement Specification Format Document

- Baseline for the development, it's a contract document
- Based on IEEE Recommendation
- 1. **INTRODUCTION**
  - 1.1 **PURPOSE**:
    - clearly state purpose of this document (and what is covered in this document)
    - by whom and how it will be used for what purpose
  - 1.2 **SCOPE**: Overall context within which the software is being developed. What parts need to be automated.
  - 1.3 **Definitions**: Acronyms, Abbreviations as applicable
  - 1.4 **REFERENCES**: to other documents
  - 1.5 **Overview of Developer's Responsibilities**: In terms of development, installation, training, maintenance, etc.

Software Requirement Specification Format Document

- **2. GENERAL DESCRIPTION**

  2.1 **PRODUCT PERSPECTIVE**:

  - State whether it is a replacement to the existing system or it is a new software
  - relationship with other products and principle interfaces

  2.2 **PRODUCT FUNCTIONS OVERVIEW**: general overview of tasks; including data flow diagrams

  2.3 **USER CHARACTERISTICS**: who they are and what training they may need

  2.4 **GENERAL CONSTRAINTS:** about schedule, resources, cost, etc.

Software Requirement Specification Format Document

3. FUNCTIONAL REQUIREMENT

3.1 **INTRODUCTION**

3.2 **INPUTS**

3.3 **PROCESSING**

3.4 **OUTPUTS**

We give functional details, we define every function (ex: cancellation of a ticket function) by giving a brief introduction to this function, inputs, processing and outputs.

It is really the body of SRS Document.

3.5 …..(repeat similarly for each function)

Software Requirement Specification Format Document

4. External Interface Requirements

4.1 **User Interfaces**: a preliminary user manual giving commands, screen formats, outputs, error messages, etc. (logical contents of these components not the layout of the screen, output)

4.2 **Hardware Interfaces**: with existing as well as new or special purpose hardware, supported device type

4.3 **Software Interfaces**: with other software packages, operating systems, etc. (railway reservation system may be interfacing with the accounting packages so that all fund transfer may be handled)

Software Requirement Specification Format Document

5. Performance Requirements

- Capacity requirements (no of users, no of files(volume of data)), response time, throughput (in measurable terms)

  Ex: Bank Transactions – how long it takes, how many transactions will be possible over a given period
- Safety requirements – recovery after software failure, any damage or loss, handling of software and hardware failure
- Security and Privacy requirements

Software Requirement Specification Format Document

6. Design Constraints

6.1 **Standards Compliance**: software development standards as well as organizational standards (e.g., for reports- regulatory needs, auditing requirements)

6.2 **Hardware Limitations**: available machines, operating systems, storage capacities, etc.

Software Requirement Specification Format Document

7. Other Requirements

Possible future extensions

Note: All sections are not required for all projects.

- It has taken into account various aspects of the software. We can handover this SRS document to the development/design team. Then they can convert this specification into a design.

Software Requirement Specification Format Document

- SRS document needs to detailed and ensure we have collected all required data put it in the form of a document

- There should be a formal review meeting with the users and users should sign off that SRS document clearly defined what the software system needs to do.

- It is also ensured that the document contains enough design details required.

Properties of a Good SRS Document

- **It should be concise**
  - at the same time should not be ambiguous.
  - Verbose and irrelevant description reduces readability and increases the possibility of errors.
- **It should be implementation-independent.**
  - It should specify what the system must do and not say how to do it.
  - Specify externally visible behaviour of the system and not discuss the implementation issue.

Properties of a Good SRS Document

- **It should be modifiable.**
  - **Easy to change,**
    - i.e., it should be well-structured.
    - Well-structured document is easy to understand and modify.
- **It should be consistent.**
- **It should be complete.**

Properties of a Good SRS Document (cont...)

- **It should be traceable**
  - You should be able to trace which part of the specification corresponds to which part of the design, code, etc and vice versa.
- **It should be verifiable**
  - e.g. “system should be user friendly” is not verifiable.
  - e.g. “A requirement that is checking of availability of books in the library” is verifiable.

Attributes of a Bad SRS Document (cont...)

- **Over-specification**
  - It occurs when analyst tries to address “how to” aspects in the specification.
  - Example, library membership record need to be stored indexed on the member’s first name or library membership identification number.
- **Forward referencing**
  - One should not refer to aspects that are discussed much later in the SRS document.
  - Reduces readability.

Properties of a Bad SRS Document (cont...)

- **Wishful thinking**
  - Description of aspects which would be difficult to implement.
- **Noise**
  - e.g., register customer function , suppose analyst writes that customer registration department is manned by clerks who report for work between 8 am and 5 pm, 7 days a week.

SRS should not include...

- **Project development plans**
  - E.g. cost, staffing, schedules, methods, tools, etc
    - Lifetime of SRS is until the software is made obsolete
    - Lifetime of development plans is much shorter
- **Product assurance plans**
  - Configuration Management, Verification & Validation, test plans, Quality Assurance, etc
    - Different audiences
    - Different lifetimes

SRS should not include (cont...)

- **Designs**
  - Requirements and designs have different audiences
  - Analysis and design are different areas of expertise

Alternatives to Natural Language(NL) specification

- Structured Natural Language (modules)
  - Usage of forms or templates
- Design Description Languages with graphical notations
  - UML Use cases, message sequence charts etc.
- Mathematical specifications
  - ADTs

Structured language specifications

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

Form-based specifications

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

what input to the module and output going as a input to other module and any sort of pre conditions on the inputs need to be validated by the module. For example, having a name then the pre condition could be not having any digital characters.

Form-based specification Example

Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered
Destination	Main control loop
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side-effects	None

Tabular specification

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

Tabular specification

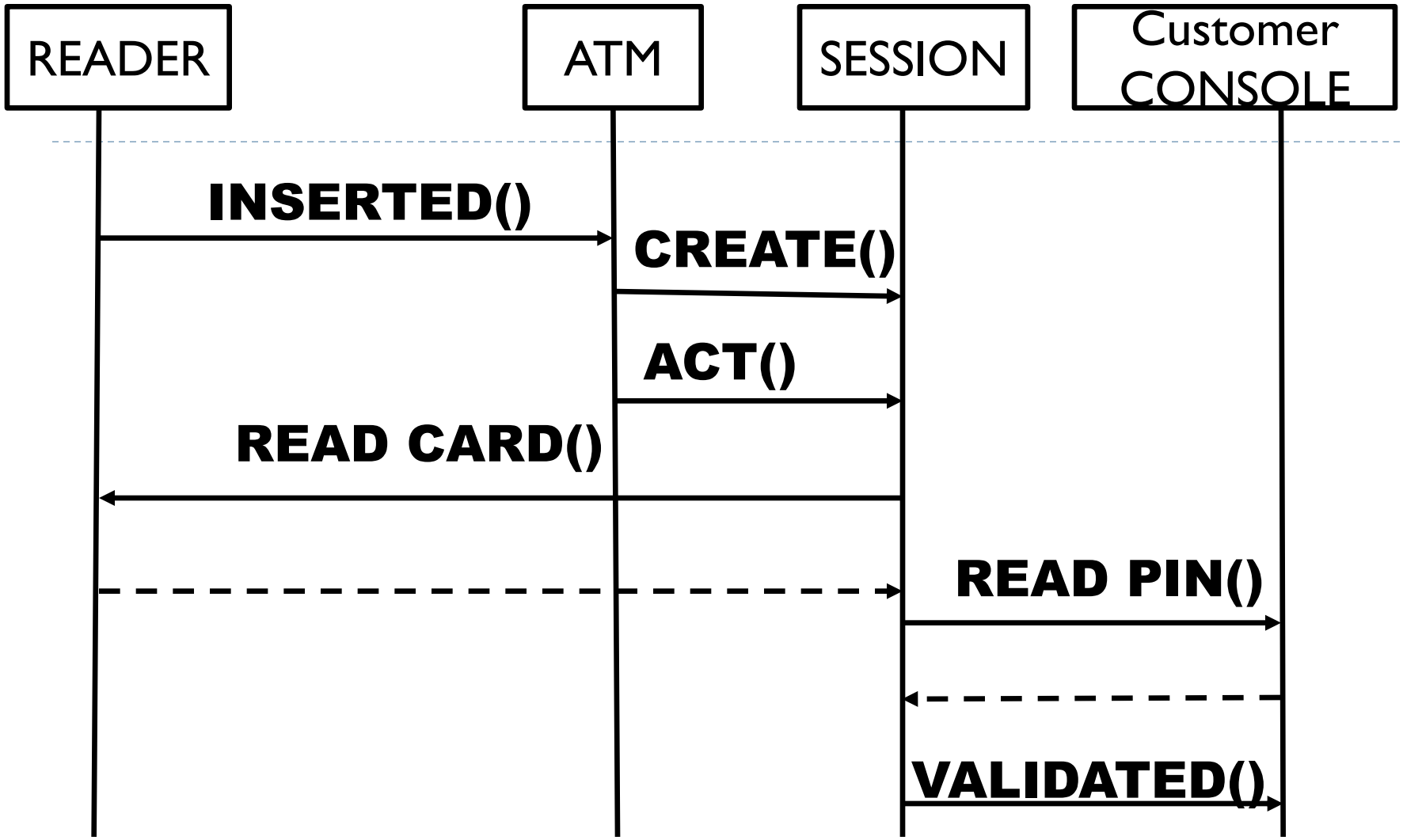
Condition	**Action**
Sugar level falling ($r_2 < r_1$)	CompDose = 0
Sugar level stable ($r_2 = r_1$)	CompDose = 0
Sugar level increasing and rate of Increase decreasing ($(r_2-r_1) < (r_1-r_0)$)	CompDose = 0
Sugar level increasing and rate of Increase stable or increasing ($(r_2-r_1) >= (r_1-r_0)$)	CompDose = round($(r_2-r_1)/4$) if rounded result = 0 then CompDose=MinimumDose

Graphical models (primarily UML derivatives)

- Graphical models are most useful when you need to show how state changes or where you need to describe a sequence of actions.

Sequence diagrams

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



- This is how it used to formally specify the user requirement

Interface specification

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.

- Three types of interface may have to be defined
  - Procedural interfaces;
  - Data structures that are exchanged;
  - Data representations.

- Formal notations are an effective technique for interface specification.

PDL interface description

```
Interface PrintServer{
        //Defines an abstract Printer Server
    void initialize(Printer p);
    void print(Printer p, Document d);
    void displayPrintQueue(Printer p);
    void cancelPrintJob(Printer p, Document d);
} //Print Server
```

The requirement document

- The requirement 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.