Diploma in Computer Engineering

Software Engineering

Dr. Vicky Kumar
Guru Nanak Dev DSEU Rohini Campus

Software Engineering

- **Software**
  - Application Software
  - System Software
  - Firmware
- **Engineering**
  - Application of science and math to solve problems
  - Use of best practices to design, evaluate, develop, test, modify, install, inspect and maintain a wide variety of products and systems

Definition

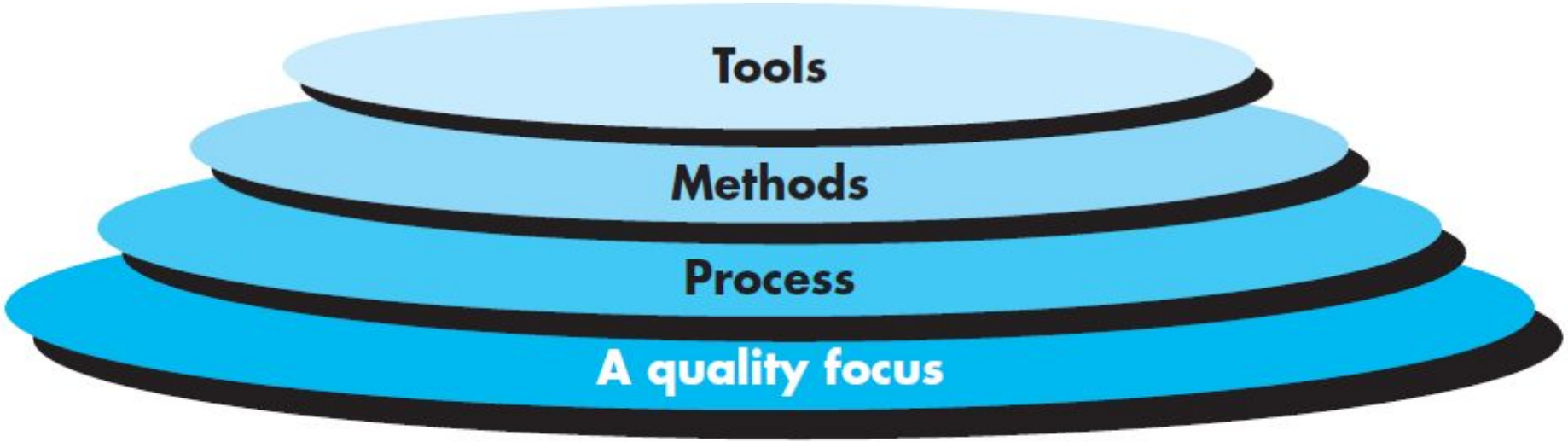
- The IEEE fully defines software engineering as:

  The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

S/W Engg. is about

Source: https://static.javatpoint.com

Software Engineering: A layered technology

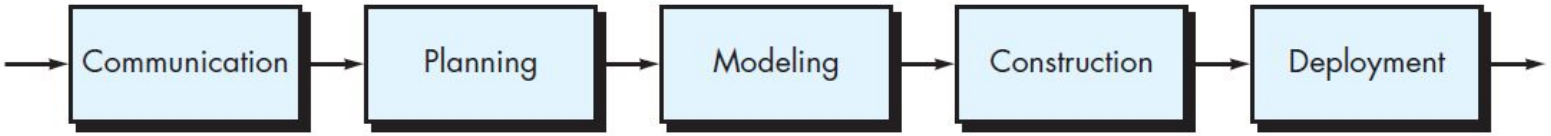


Contd.

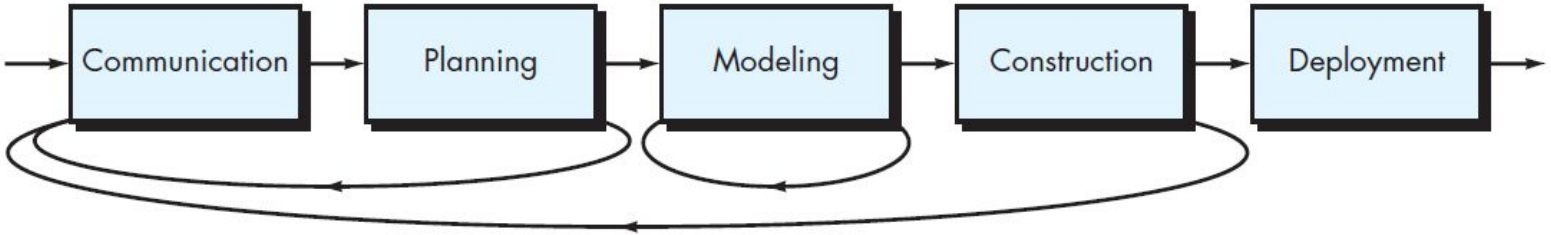
- **Quality focus:** Utmost attention to quality
  - CMM, Six sigma, ISO
- **Process:** Framework for timely delivery of software technology
  - Defines who is doing what, when and how to reach a certain goal
  - Management control of the software products
  - E.g. Models, Documents, Data, Reports, Forms, etc.
- **Methods:** Technical how-to’s for building software
  - Requirement analysis, design, development, testing, deployment, maintenance
- **Tools:** Automated or semi-automated support for process and methods
  - Flowchart makers, report/document generator, code generator, code analyser

Software Process

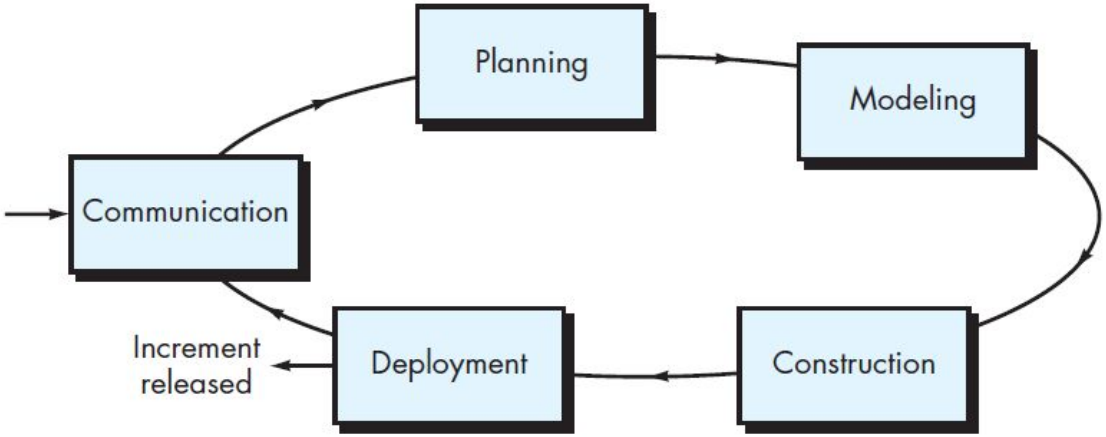
- Communication
- Planning
- Modeling
- Construction
- Deployment



(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow

Software Development Life Cycle (SDLC)

- Feasibility study
- Requirement analysis and specification
  - Requirement Gathering and Analysis
  - Requirement Specification
- Designing
- Coding and Unit Testing
- Integration and system testing
- Deployment
- Maintenance

Software Process Models

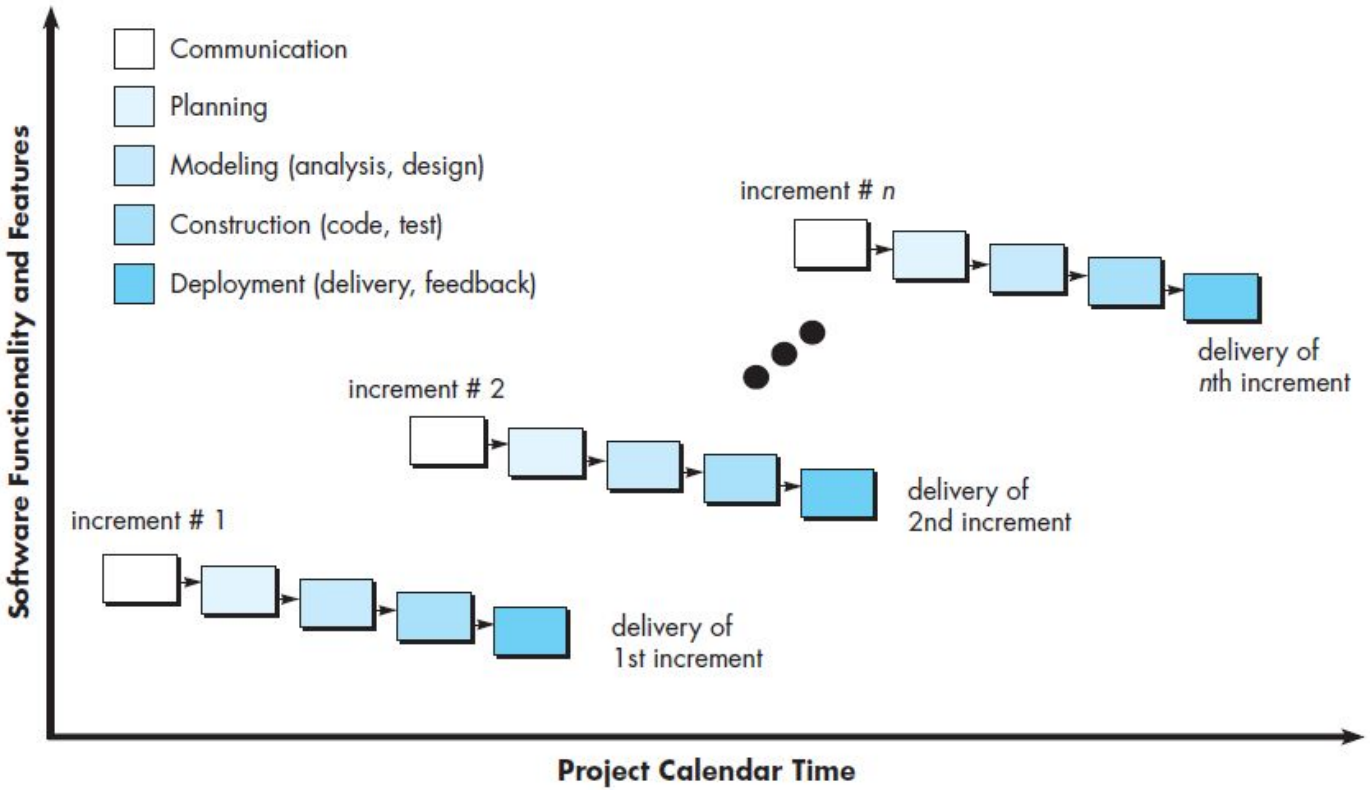
- **Classical Waterfall Model**

  (https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/?ref=lbp)
- **Iterative Waterfall Model**

  (https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/)
- **Rapid Application Development Model**

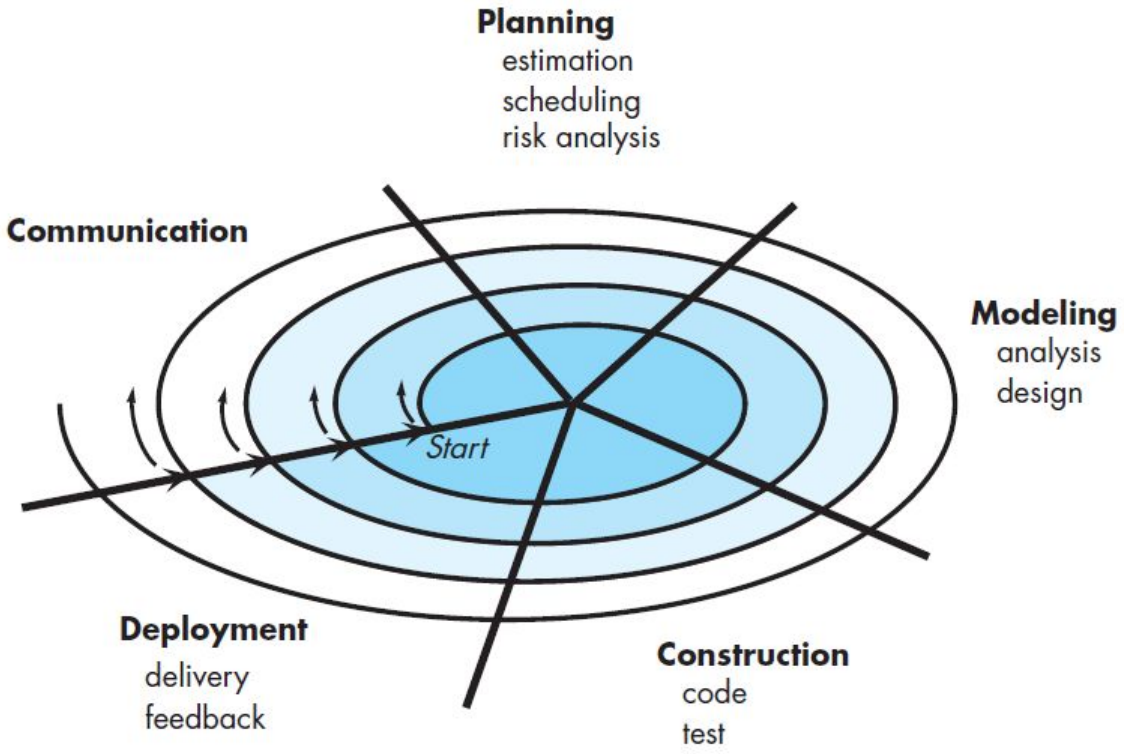
  (https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/)

Incremental Model



Spiral Model

(https://www.geeksforgeeks.org/software-engineering-spiral-model/?ref=lbp)



Software Requirement Specifications (SRS)

- **Requirement:**

  (1) A condition of capability needed by a user to solve a problem or achieve an objective;

  (2) A condition or a capability that must be met or possessed by a system to satisfy a contract, standard, specification, or other formally imposed document”

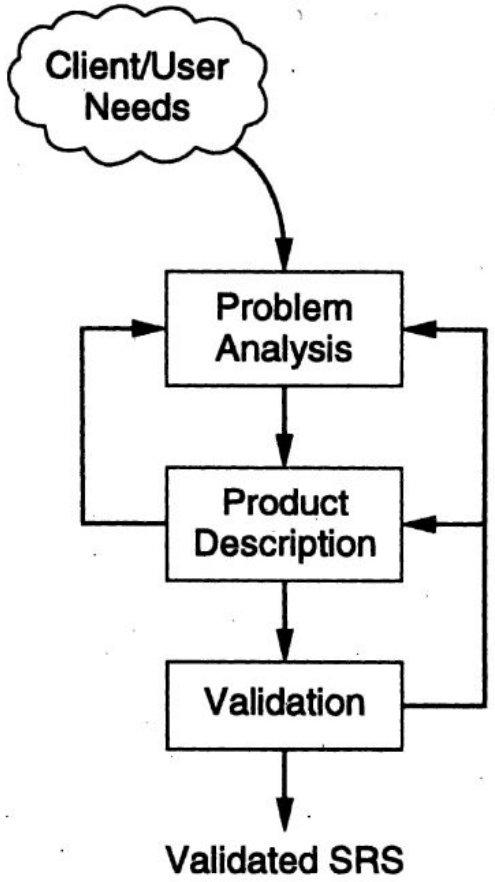
- **SRS:** describes what the proposed software should do without describing how the software will do it.

Value of a Good SRS

- Bridges the communication gap between client and developer
- Through SRS, the client clearly describes what it expects from the supplier, and the developer clearly understands what capabilities to build in the software
- Without a proper SRS, there is no way a client can determine if the software being delivered is what was ordered, and there is no way the developer can convince the client that all the requirements have been fulfilled
- An SRS provides a reference for validation of the final product

Requirement Process

- **Requirement Analysis**
  - Thorough study of problem statement
  - Meeting with clients and even with end-users
- **Requirement Specification**
- **Requirement Validation**



Requirement Specification

- **Characteristics**
  - **Correct**
  - **Complete**
  - **Unambiguous**
  - **Verifiable**
  - **Consistent**
  - **Ranked for importance or stability**

Component of an SRS

Requirements pertaining to the

- **Functionality**
  - Describe the relationship between the input and output of the system
- **Performance**
  - **Static:** No. of users or terminals to be supported
  - **Dynamic:** Response time or throughput
- **Design constraints imposed on an implementation**
  - Hardware and Software resources constraints, standards compliance, reliability, fault-tolerance, security
- **External interfaces**
  - Interfaces with hardware, software, and end-users

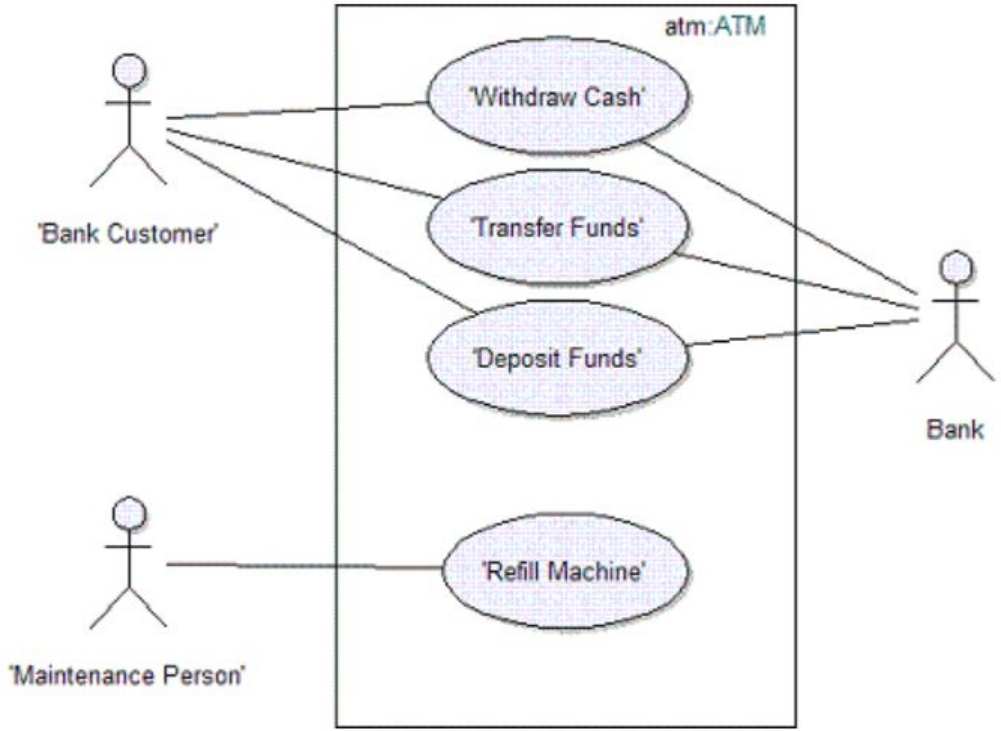
Structure of an SRS

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, Acronyms, and Abbreviations
    - 1.4 References
    - 1.5 Overview
2. Overall Description
    - 2.1 Product Perspective
    - 2.2 Product Functions
    - 2.3 User Characteristics
    - 2.4 General Constraints
    - 2.5 Assumptions and Dependencies
3. Specific Requirements

Contd.

3. Detailed Requirements
    - 3.1 External Interface Requirements
        - 3.1.1 User Interfaces
        - 3.1.2 Hardware Interfaces
        - 3.1.3 Software Interfaces
        - 3.1.4 Communication Interfaces
    - 3.2. Functional Requirements
        - 3.2.1 Mode 1
            - 3.2.1.1 Functional Requirement 1.1
            - :
            - 3.2.1.n Functional Requirement 1.n
        - :
        - 3.2.m Mode m
            - 3.2.m.1 Functional Requirement m.1
            - :
            - 3.2.m.n Functional Requirement m.n
    - 3.3 Performance Requirements
    - 3.4 Design Constraints
    - 3.5 Attributes
    - 3.6 Other Requirements

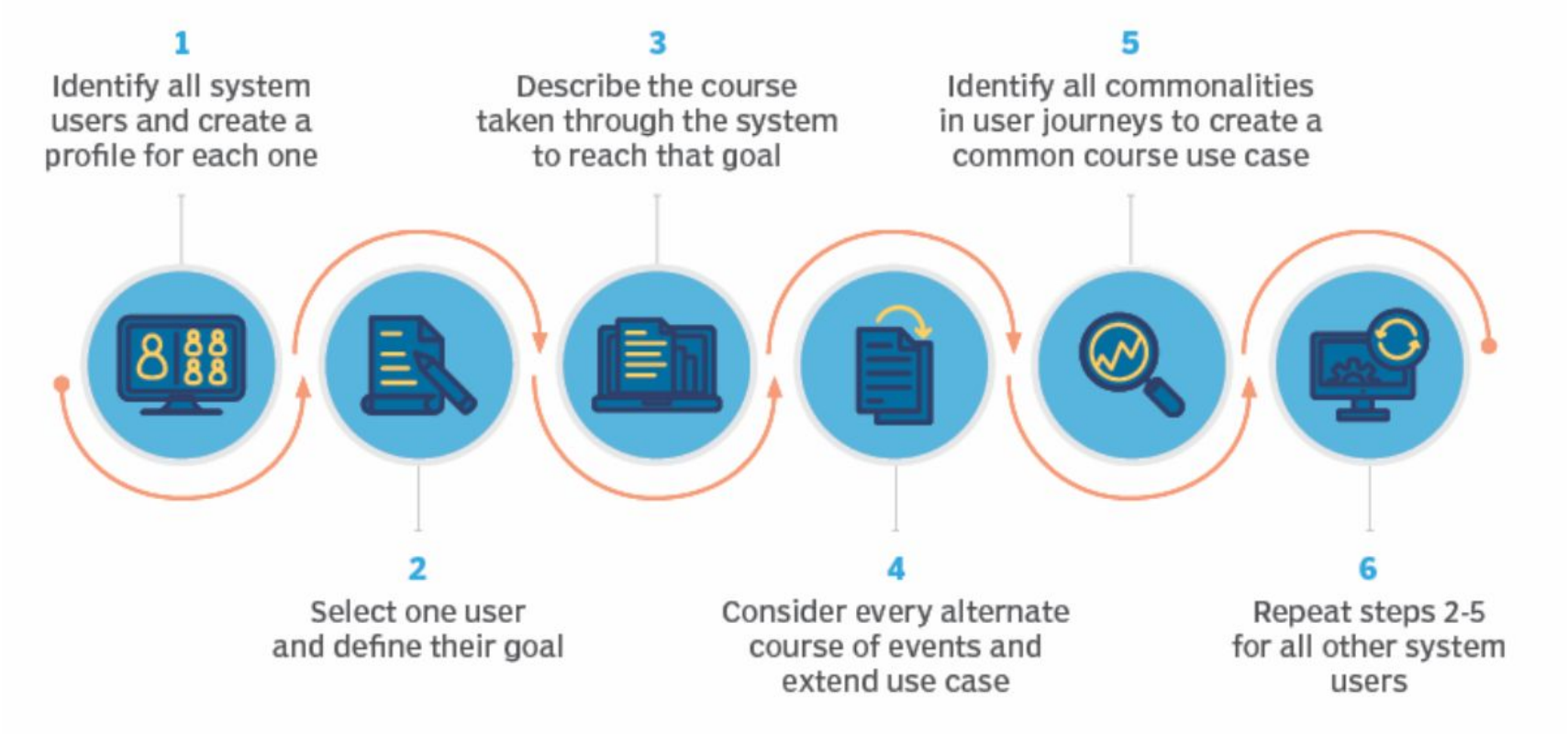
Concepts of Use Cases for Functional Specification



Components of a Use Case

- **System**
- **Goal**
- **Actor:** Person or a system which uses the system for achieving some goal
  - **Primary actor**
- **Precondition**
- **Scenario:** Sequence of activities to be performed to achieve the goal
  - **Main success scenario**
  - **Extension scenario**

Use Case Methodology



Example of a Use case

- ***UC1*: Put an item for auction**
  Primary Actor: Seller
  Precondition: Seller has logged in
  Main Success Scenario:

  1. Seller posts an item (its category, description, picture, etc.) for auction
  2. System shows past prices of similar items to seller
  3. Seller specifies the starting bid price and a date when auction will close
  4. System accepts the item and posts it

  Exception Scenarios:
  - 2 a) There are no past items of this category

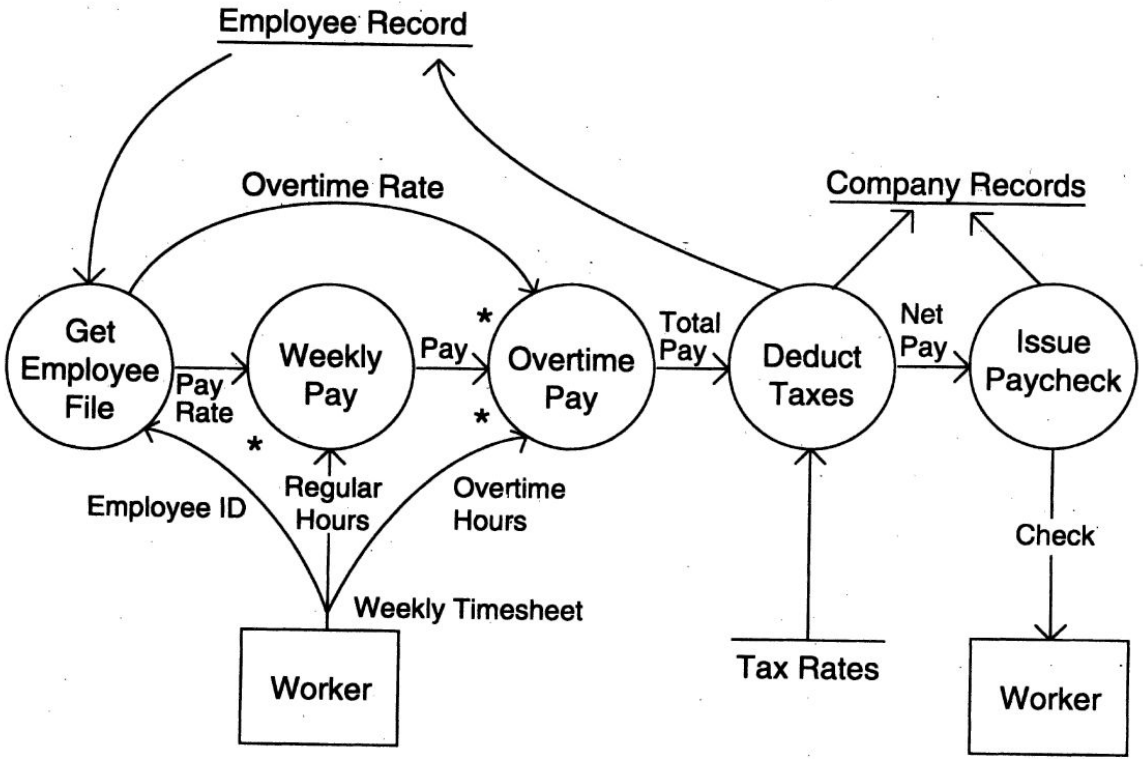
    - System tells the seller this situation

Contd.

- ***UC2*: Make a bid**
  Primary Actor: Buyer
  Precondition: The buyer has logged in
  Main Success Scenario:
  1. Buyer searches or browses and selects some item
  2. System shows the rating of the seller, the starting bid, the current bids, and the highest bid; asks buyer to make a bid
  3. Buyer specifies a bid price
  4. System accepts the bid; Blocks funds in bidders account
  5. System updates the max bid price, informs other users, and updates the records for the item

  Exception Scenarios:
  - 3 a) The bid price is lower than the current highest
    - System informs the bidder and asks to rebid
  - 4 a) The bidder does not have enough funds in his account
    - System cancels the bid, asks the user to get more funds

Data Flow Diagram for process analysis



Software metrics

- **Metric:** Measurement of software characteristics (size, cost, reliability, etc.)

  A quantitative measure of the degree to which a system, component, or process possesses a given attribute

- **Categories**
  - **Product metrics (e.g. size, complexity, reliability)**
  - **Process metrics (e.g. design, tool, technique)**
  - **Project metrics (e.g. cost, man power, efforts, time)**

Contd.

- **Type of metrics**
  - **Internal** (Lines of Code (LOC) measure)
  - **External** (Portability, reliability, functionality, usability, etc)
  - **Hybrid** (combines product, process, and resource metrics)
- **Process**
  - Identification of appropriate metrics
  - Data for Formulation of metrics
  - Analysis of results obtained based on past data
  - Interpretation of analysed results
  - Modification in the requirement, design model, coding, testing, etc.

Size-oriented metrics

- **Lines of Code (LOC) or Thousand lines of code (KLOC) forms the basis for metrics**
- **Based on LOC or KLOC following metrics are derived**
  - **Effort (man power/month)**
  - **Cost**
  - **Documentation Pages**
  - **Errors**
  - **Defects**
  - **People**

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
•	•	•	•	•	•		
•	•	•	•	•	•		
•	•	•	•	•	•		

Function-oriented metrics

- Functional-point metrics measure the functionality delivered by a system
- Basis for FP is the requirements of system
- The FP metrics can be used to

  (1) estimate the cost or effort required to design, code, and test the software;

  (2) predict the number of errors that will be encountered during testing; and

  (3) forecast the number of components and/or the number of projected source lines in the implemented system.

Contd.

- FPs are derived considering
  - Number of external inputs
  - Number of external outputs
  - Number of external inquiries
  - Number of internal logical files
  - Number of external interface files
- Weighing factor (three classes)
- Unadjusted FP count
- Value adjustment factor
- Adjusted FP count

Halstead Metrics

- Used to determine the complexity of the system
- Unique and total occurence of
  - Operators
  - Operands
- https://www.ibm.com/docs/en/rtr/8.0.0?topic=SSSHUF_8.0.0/com.ibm.rational.testrt.studio.doc/topics/csmhalstead.htm