CAPTURING REQUIREMENTS

Mehwish Mumtaz

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

Fred Brooks

Requirements

- A ***requirement*** is an expression of desired behaviour
- Requirements focus on the customer needs, not on the solution or implementation

A Systems Approach (Contd.)

- Objects
  - i.e. things
- Activities
  - Actions taken
  - Input / Outputs
- Relationships for example:
  - Which object performs what activities
  - Which objects are associated with other objects
- System Boundary
  - Who generates input and who receives output
  - Which objects/activities are part of the system and which are not
  - Nested systems, related systems, interrelated systems

- We need a time series model for the purpose of forecasting demand
- Why projects fail
  - One reason incomplete requirements
  - Requirement gathering tasks- Requirement Analyst/System Analyst
  - **Important Note:**

Note that none of these requirements specify how the system is to be implemented. There is no mention of what database-management system to use, whether a client-server architecture will be employed, how much memory the computer is to have, or what programming language must be used to develop the system. These implementation-specific descriptions are not considered to be requirements (unless they are mandated by the customer). The goal of the requirements phase is to understand the customer's problems and needs. Thus, requirements focus on the customer and the problem, not on the solution or the implementation. We often say that requirements designate *what* behavior the customer wants, without saying *how* that behavior will be realized. Any discussion of a solution is premature until the problem is clearly defined.

Inception

- Identify: all stakeholders, measurable benefits of successful implementation, possible alternatives
- Ask questions stepwise, as early as possible, first meeting/encounter
- Possible questions at 1st step (stakeholders, overall goals and benefits):
  - Who is behind the request for this work?
  - Who will use this solution?
  - What will be the economic benefit of a successful solution?

•Goods Retailer with physical shops;
•Food retailer
•E-commerce retailer
•Wholesaler (any)

Inception

- Possible questions at 2nd step (detailed understanding and customer perception about the solution):
  - What problems(s) will this solution address?
  - Can you show me (or describe) the business environment in which the solution will be used?
  - How do you characterize the 'good' output?
- Possible questions at 3rd step (effectiveness of communication):
  - Are you the right person to answer these questions?
  - Are my questions relevant to the problem that you have?
  - Can anyone else provide additional information?

Requirements Elicitation

- Get more detailed requirements
  - Problem **Customer** knows their business but do not good at describing the business
  - Developer problem ?jargons, different words have different meaning

How Developers See Users	How Users See Developers
Users don't know what they want.	Developers don't understand operational needs.
Users can't articulate what they want.	Developers can't translate clearly stated needs into a successful system.
Users are unable to provide a usable statement of needs.	Developers set unrealistic standards for requirements definition.
Users have too many needs that are politically motivated.	Developers place too much emphasis on technicalities.
Users want everything right now.	Developers are always late.
Users can't remain on schedule.	Developers can't respond quickly to legitimately changing needs.
Users can't prioritize needs.	Developers are always over budget.
Users are unwilling to compromise.	Developers say "no" all the time.
Users refuse to take responsibility for the system.	Developers try to tell us how to do our jobs.
Users are not committed to development projects.	Developers ask users for time and effort, even to the detriment of the users' important primary duties.

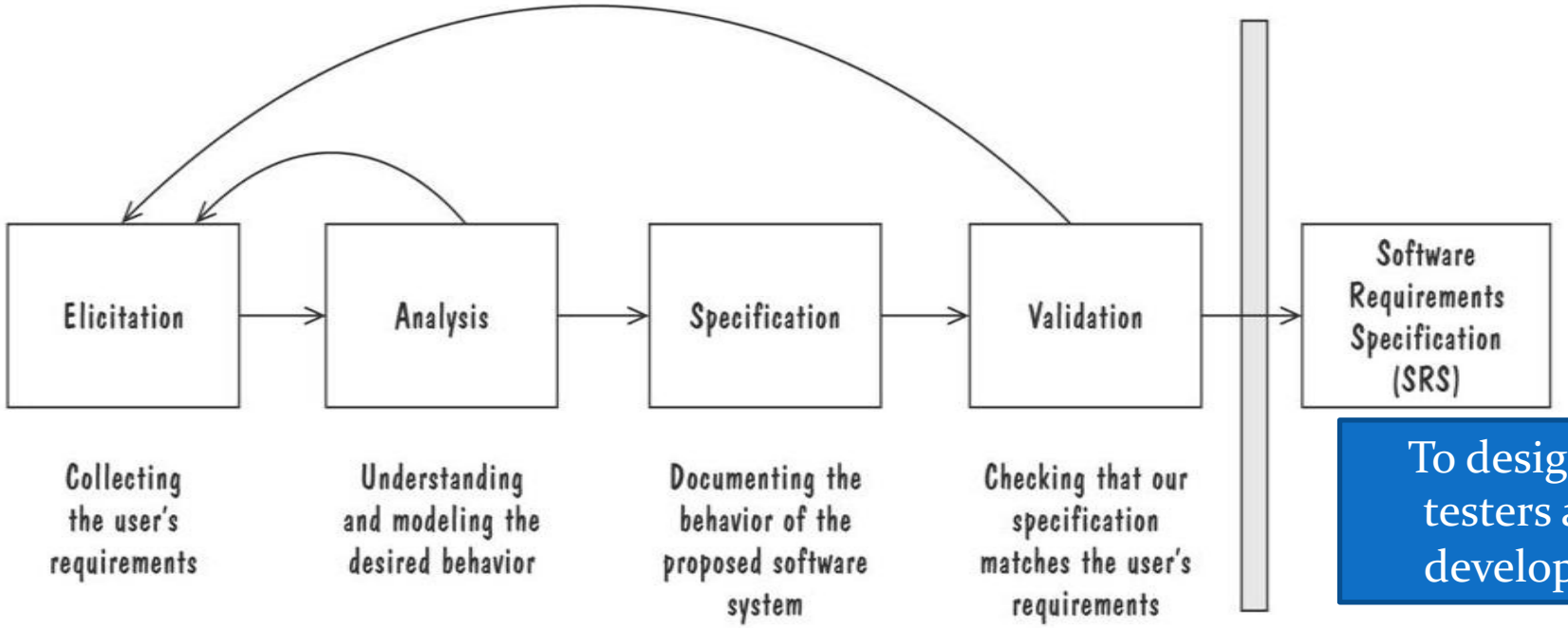
Requirements Elicitation

- Customers do not always understand what their needs and problems are
- It is important to discuss the requirements with everyone who has a stake in the system
- Come up with agreement on what the requirements are
  - If we cannot agree on what the requirements are, then the project is doomed to fail

Requirements Elicitation

- Different Stakeholders are:
  - **Clients**: pay for the software to be developed
  - **Customers=User**: buy the software after it is developed
  - **Users**: use the system
  - **Domain experts**: familiar with the problem that the software must automate example Consulting Banker for bank system
  - **Market Researchers**: conduct surveys to determine future trends and potential customers
  - **Lawyers or auditors**: familiar with government, safety, or legal requirements
  - **Software engineers** or other technology experts

Requirements Elicitation



Process for Capturing Requirements

Asking Questions/Study Similar Systems	Prototype	How to implement the solution	Matches customer Expectations

Requirements Elicitation

Conflict of Requirements

Example: Three modes of a software

Requirement Analyst?
Analysis Skills

Requirements Elicitation

In addition to interviewing stakeholders, other means of eliciting requirements include

- Reviewing available documentation, such as documented procedures of manual tasks, and specifications or user manuals of automated systems
- Observing the current system (if one exists), to gather objective information about how the users perform their tasks, and to better understand the system we are

the old system continues to be used because it provides some critical function that the designers of the new system overlooked

- Apprenticing with users (Beyer and Holtzblatt 1995), to learn about users' tasks in more detail, as the user performs them
- Interviewing users or stakeholders in groups, so that they will be inspired by one another's ideas
- Using domain-specific strategies, such as Joint Application Design (Wood and Silver 1995) or PIECES (Wetherbe 1984) for information systems, to ensure that stakeholders consider specific types of requirements that are relevant to their particular situations
- Brainstorming with current and potential users about how to improve the proposed product

Elaboration

- Analyze, model, specify…
- Some Analysis Techniques
  - Data Flow Diagrams (DFD)
  - Use case Diagram
  - Object Models (ER Diagrams, Abstract class diagrams)
  - Decision Tables
  - State Diagrams (State charts)
  - Fence Diagrams
  - Event Tables
  - Petri Nets
  - Event Traces (Message Sequence Charts)

Flow-oriented, Scenario-based, Class-based, Behavioral

Data Flow Diagrams (DFDs)

- A data-flow diagram (DFD) models functionality and the flow of data from one function to another
  - A bubble represents a *process (or data transform)*
  - A labelled arrow represents *data flow*, the label describes the data being transferred
  - A *data store*: a formal repository or database of information
  - Rectangles represent *Information source/sink*: entities that provide input data or receive the output result
  - Heritage identifier: to document stepwise refinement

Creating Data Flow Diagrams

Steps:

1. Create a list of activities
2. Construct Context Level DFD
   (identifies external entities and processes)
3. Construct Level 0 DFD
   (identifies manageable sub process)
4. Construct Level 1- n DFD
   (identifies actual data flows and data stores)
5. Check against rules of DFD

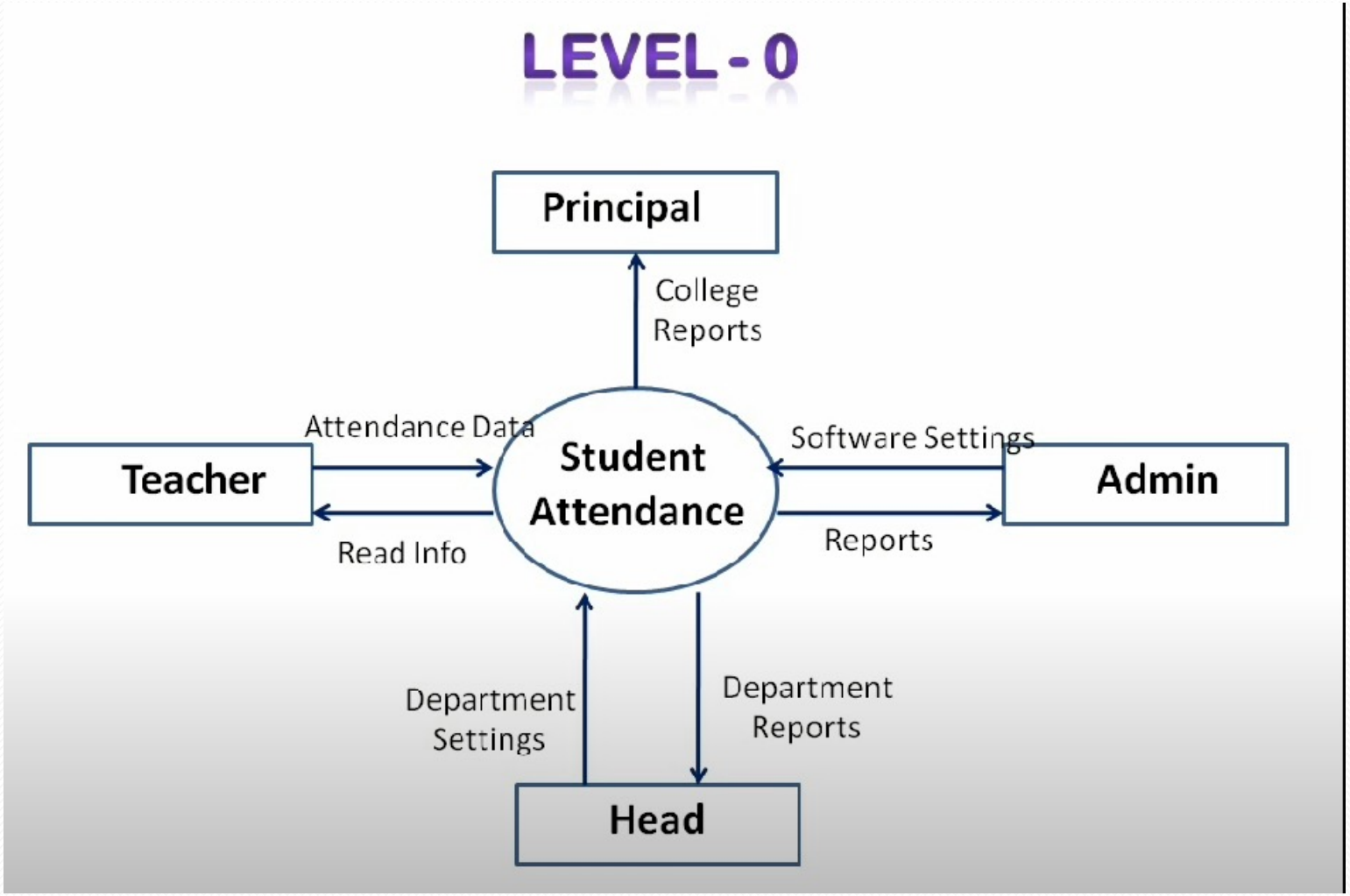
DFD Naming Guidelines

- External Entity → Noun
- Data Flow → Names of data
- Process → verb phrase
  - a system name
  - a subsystem name
- Data Store → Noun

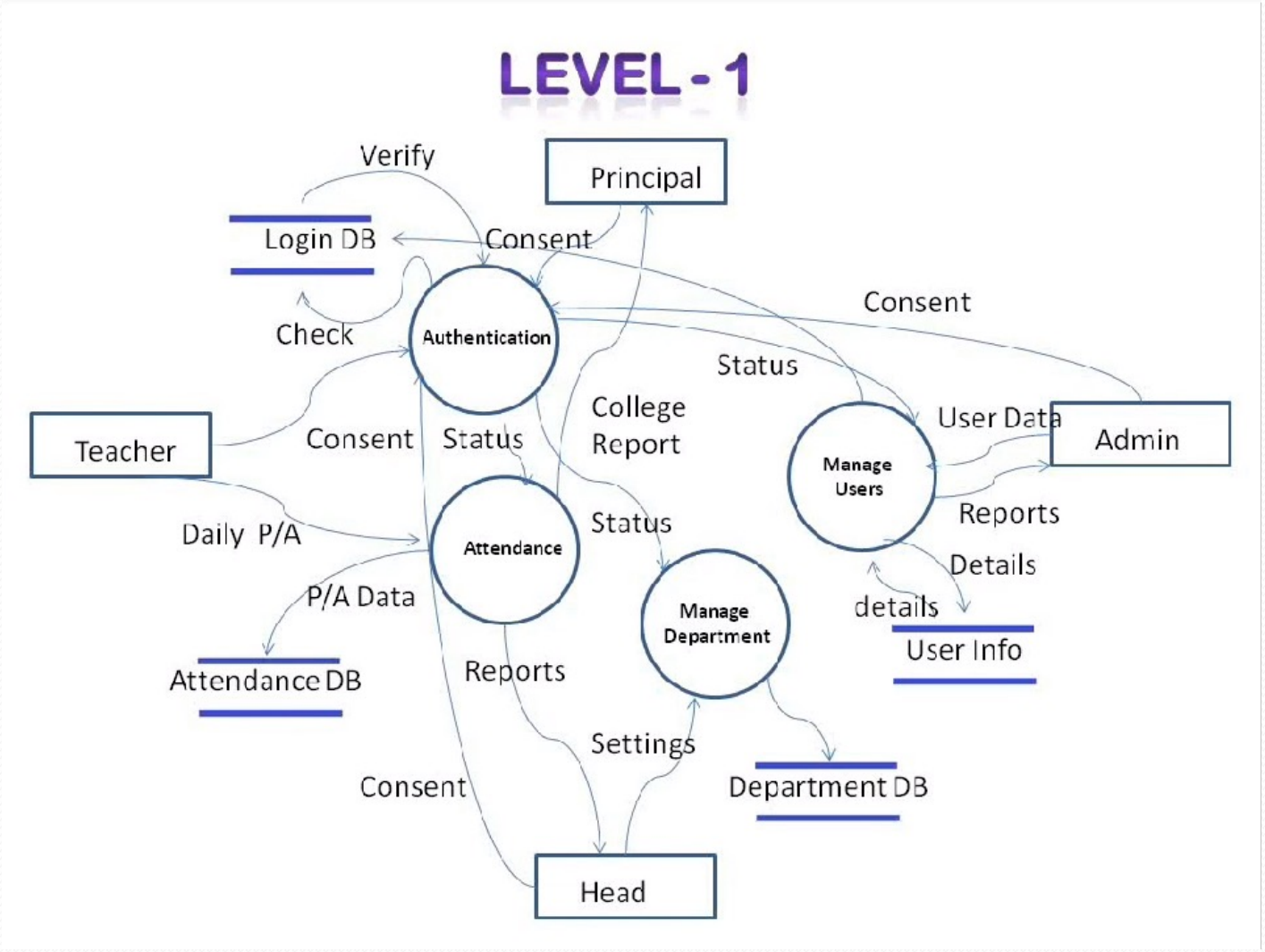
Requirements of attendance Management System

- Admin can manager users and view reports
- Principal can view college reports
- Teacher mark attendance and view attendance
- Head allocate departments and view department report

Level 0 DFD-Contextual Diagram



Level 0 DFD-Contextual Diagram



Data Flow Diagrams (Contd.)

- Advantage:
  - Provides an intuitive model of a proposed system's high-level functionality and of the data dependencies among various processes
- Disadvantage:
  - Can be aggravatingly ambiguous to a software developer who is less familiar with the problem being modelled

Checking Consistency of DFDs

- The following checklist can be used to verify the consistency of a collection of DFDs:
  1. Is each requirements function represented by a data transform somewhere in the DFDs?
  2. Is each system input and output represented in the DFDs?
  3. Is each input and output from higher-level DFDs reproduced correctly in any lower-level refinement DFDs?
  4. Is each transform in the lowest-level DFDs is a primitive one?

Checking Consistency of DFDs (Contd.)

- The following checklist can be used to verify the consistency of a collection of DFDs (contd.):
  5. Do all information flows have labels that correspond to an entry in the data dictionary?
  6. Do all data dictionary entries appear in some DFD?