MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Code: COB 4116

Course Name: ***Software Engineering***

Lecture Name: MAGEMO A.

Phone:0754238865

E-Mail:magemohoward30@gmail.com

Contact hour: Monday through Friday

LECTURE 3

Software Requirements

- The **software requirements** are description of features and functionalities of the target system.
- Requirements convey the expectations of users from the software product.
- The requirements can be obvious or hidden, known or unknown, expected or unexpected from client’s point of view.

LECTURE 3

Requirement Engineering Process

It is a four step process, which includes –

1. Feasibility Study
2. Requirement Gathering
3. Software Requirement Specification
4. Software Requirement Validation

Feasibility study

- ✓ This **feasibility study** is focused towards goal of the organization. This study **analyzes** whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization.
- ✓ It explores technical aspects of the project and product such as usability, maintainability, productivity and integration ability.
- ✓ The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

Requirement Gathering

- If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user.
- Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

Software Requirement Specification

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- SRS defines how the intended software will interact with hardware, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

Software Requirement Specification

SRS should come up with following features:

- User Requirements are expressed in natural language.
- Technical requirements are expressed in structured language, which is used inside the organization.
- Design description should be written in Pseudo code. Format of Forms and GUI screen prints.Conditional and mathematical notations for DFDs etc.

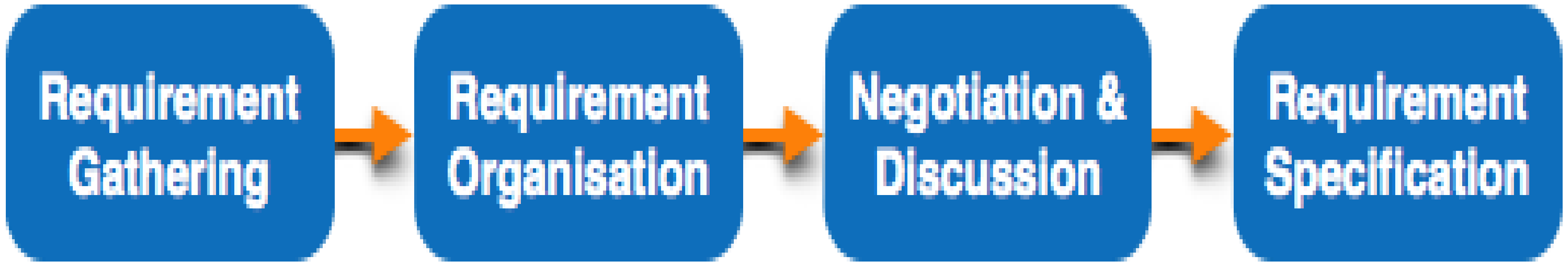
Software Requirement Validation

Requirements can be checked against following conditions –

- ✓ If they can be practically implemented
- ✓ If they are valid and as per functionality and domain of software
- ✓ If there are any ambiguities
- ✓ If they are complete
- ✓ If they can be demonstrated

Requirement Elicitation Process

- Requirement elicitation process can be depicted using the following diagram:



- **Requirements gathering** - The developers discuss with the client and end users and know their expectations from the software.
- **Organizing Requirements** - The developers prioritize and arrange the requirements in order of importance, urgency and convenience.

Requirement Elicitation Process

- **Negotiation & discussion** - If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, if they are, it is then negotiated and discussed with stakeholders.
- Requirements may then be prioritized and reasonably compromised.
- **Documentation** - All formal & informal, functional and non-functional requirements are documented and made available for next phase processing.

Requirement Elicitation Techniques

- **Requirements Elicitation** *is the process to find out the requirements for an intended software system by communicating with client*, *end users*, *system users and others who have a stake in the software system development*.
- There are various ways to discover requirements:

Interviews

- Interviews are strong medium to collect requirements. Organization may conduct several types of interviews such as:
  - **Structured (closed) interviews**, where every single information to gather is decided in advance, they follow pattern and matter of discussion firmly.
  - **Non-structured (open) interviews**, where information to gather is not decided in advance, more flexible and less biased.
  - **Oral interviews**
  - **Written interviews**
  - **One-to-one interviews** which are held between two persons across the table.
  - **Group interviews** which are held between groups of participants. They help to uncover any missing requirement as numerous people are involved.

Surveys

- Organization may conduct surveys among various stakeholders by querying about their expectation and requirements from the upcoming system.

Questionnaires

- A document with pre-defined set of objective questions and respective options is handed over to all stakeholders to answer, which are collected and compiled.
- A *shortcoming of this technique* is, if an option for some issue is not mentioned in the questionnaire, the issue might be left unattended.

Task analysis

- Team of engineers and developers may analyze the operation for which the new system is required. If the client already has some software to perform certain operation, it is studied and requirements of proposed system are collected.

Domain Analysis

- Every software falls into some domain category. The expert people in the domain can be a great help to analyze general and specific requirements.

Brainstorming

- An informal debate is held among various stakeholders and all their inputs are recorded for further requirements analysis.

Prototyping

- Prototyping is building user interface without adding detail functionality for user to interpret the features of intended software product. It helps giving better idea of requirements.

Prototyping

- If there is no software installed at client’s end for developer’s reference and the client is not aware of its own requirements, *the developer creates a prototype based on initially mentioned requirements.* The prototype is shown to the client and the feedback is noted. The client feedback serves as an input for requirement gathering.

Observation

- Team of experts visit the client’s organization or workplace.
- They observe the actual working of the existing installed systems.
- They observe the workflow at client’s end and how execution problems are dealt.
- The team itself draws some conclusions which aid to form requirements expected from the software.

Software Requirements Characteristics

- A complete Software Requirement Specifications must be:

- Clear
- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable
- Verifiable
- Prioritized
- Unambiguous
- Traceable
- Credible source

Software Requirements

- ➢ Broadly software requirements should be categorized in two categories:

Functional Requirements

- ➢ Requirements, which are related to functional aspect of software fall into this category.
- ➢ They define functions and functionality within and from the software system.

Examples -

- ✓ Search option given to user to search from various invoices.
- ✓ User should be able to mail any report to management.

Non-Functional Requirements

- Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of.

Non-functional requirements include -

- ✓ Security
- ✓ Storage
- ✓ Configuration
- ✓ Performance

User Interface requirements

- UI is an important part of any software or hardware or hybrid system.
- User interface requirements are briefly mentioned below -
  - ✓ Content presentation
  - ✓ Easy Navigation
  - ✓ Simple interface
  - ✓ Responsive

Software System Analyst

- **System analyst in an IT organization** is a person, who analyzes the requirement of proposed system and ensures that requirements are conceived and documented properly & correctly.
- Role of an analyst starts during Software Analysis Phase of SDLC. It is the responsibility of analyst to make sure that the developed software meets the requirements of the client.

System Analysts have the following responsibilities:

- ✓ Analyzing and understanding requirements of intended software
- ✓ Understanding how the project will contribute in the organization objectives
- ✓ Identify sources of requirement
- ✓ Validation of requirement
- ✓ Develop and implement requirement management plan
- ✓ Documentation of business, technical, process and product requirements
- ✓ Coordination with clients to prioritize requirements and remove and ambiguity
- ✓ Finalizing acceptance criteria with client and other stakeholders

Software Design Basics

- Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

Software Design Levels

- ➢ Software design yields three levels of results:
- ✓ **Architectural Design -** The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.

Software Design Levels

- ✓ **High-level Design-** The high-level design breaks the ‘single entity-multiple component’ concept of architectural design into less-abstracted view of sub-systems and modules and depicts their interaction with each other.
- ➢ High-level design focuses on how the system along with all of its components can be implemented in forms of modules.
- ➢ It recognizes modular structure of each sub-system and their relation and interaction among each other.

Software Design Levels

- ✓ **Detailed Design-** Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules.

Modularization

- **Modularization** is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently.
- These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.

Modularization

Advantage of modularization:

- ✓ Smaller components are easier to maintain
- ✓ Program can be divided based on functional aspects
- ✓ Desired level of abstraction can be brought in the program
- ✓ Components with high cohesion can be re-used again
- ✓ Concurrent execution can be made possible
- ✓ Desired from security aspect

Concurrency

- In software design, concurrency is implemented by splitting the software into multiple independent units of execution, like modules and executing them in parallel. In other words, concurrency provides capability to the software to execute more than one part of code in parallel to each other.
- It is necessary for the programmers and designers to recognize those modules, which can be made parallel execution.

Example

- The spell check feature in word processor is a module of software, which runs along side the word processor itself.

Design Verification

- The output of software design process is design documentation, pseudo codes, detailed logic diagrams, process diagrams, and detailed description of all functional or non-functional requirements.
- The next phase, which is the implementation of software, depends on all outputs mentioned above

Software Analysis & Design Tools

- Software analysis and design includes all *activities, which help the transformation of requirement specification into implementation*
- It helps human-readable requirements to be transformed into actual code.
- Let us see few analysis and design tools used by software designers:

Software Analysis & Design Tools

1. Data flow diagram(DFD)
2. Structure Charts
3. Hipo Diagram
4. Structured English
5. Pseudo-code
6. Decision Tables
7. Entity-Relationship Model
8. Data Dictionary

Software Analysis & Design Tools

Data Flow Diagram

- Data flow diagram is graphical representation of flow of data in an information system.
- It is capable of depicting incoming data flow, outgoing data flow and stored data.
- The DFD does not mention anything about how data flows through the system.

Software Analysis & Design Tools

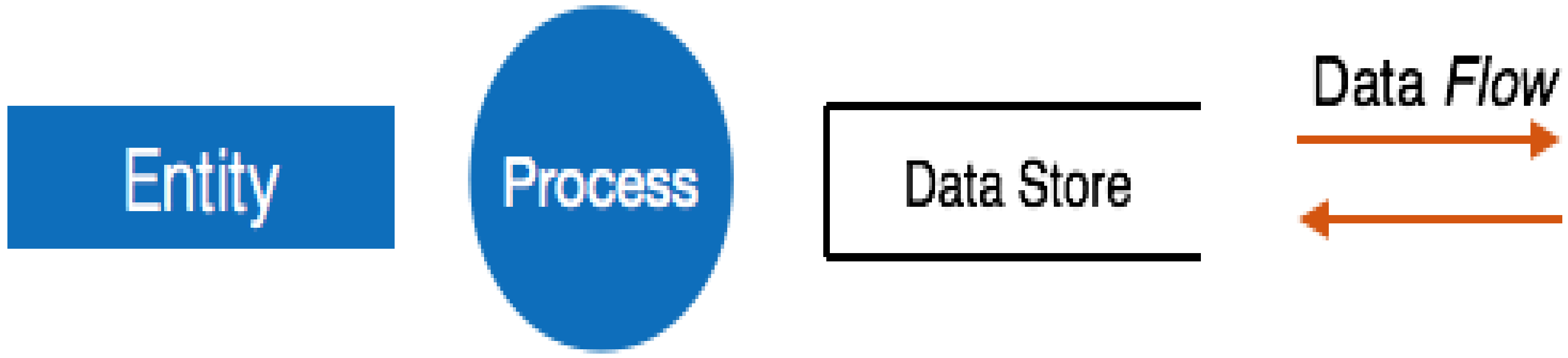
Types of DFD

- Data Flow Diagrams are either Logical or Physical.
- **Logical DFD** - This type of DFD concentrates on the system process, and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

Software Analysis & Design Tools

DFD Components

- DFD can represent Source, destination, storage and flow of data using the following set of components

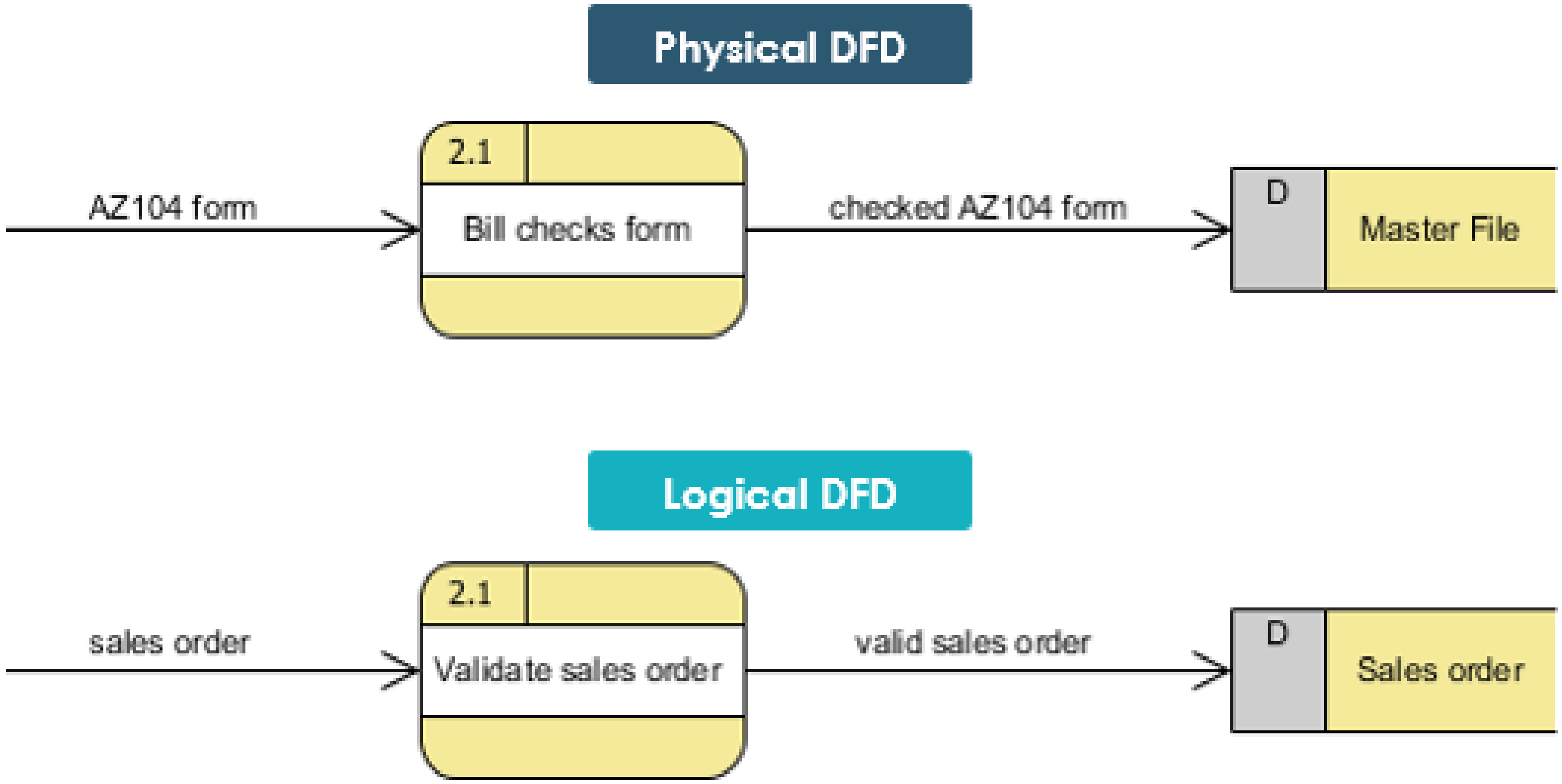


Software Analysis & Design Tools

- ❖ **Entities** - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.
- ❖ **Process** - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- ❖ **Data Storage** - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- ❖ **Data Flow** - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Software Analysis & Design Tools

❖ Example of DFD



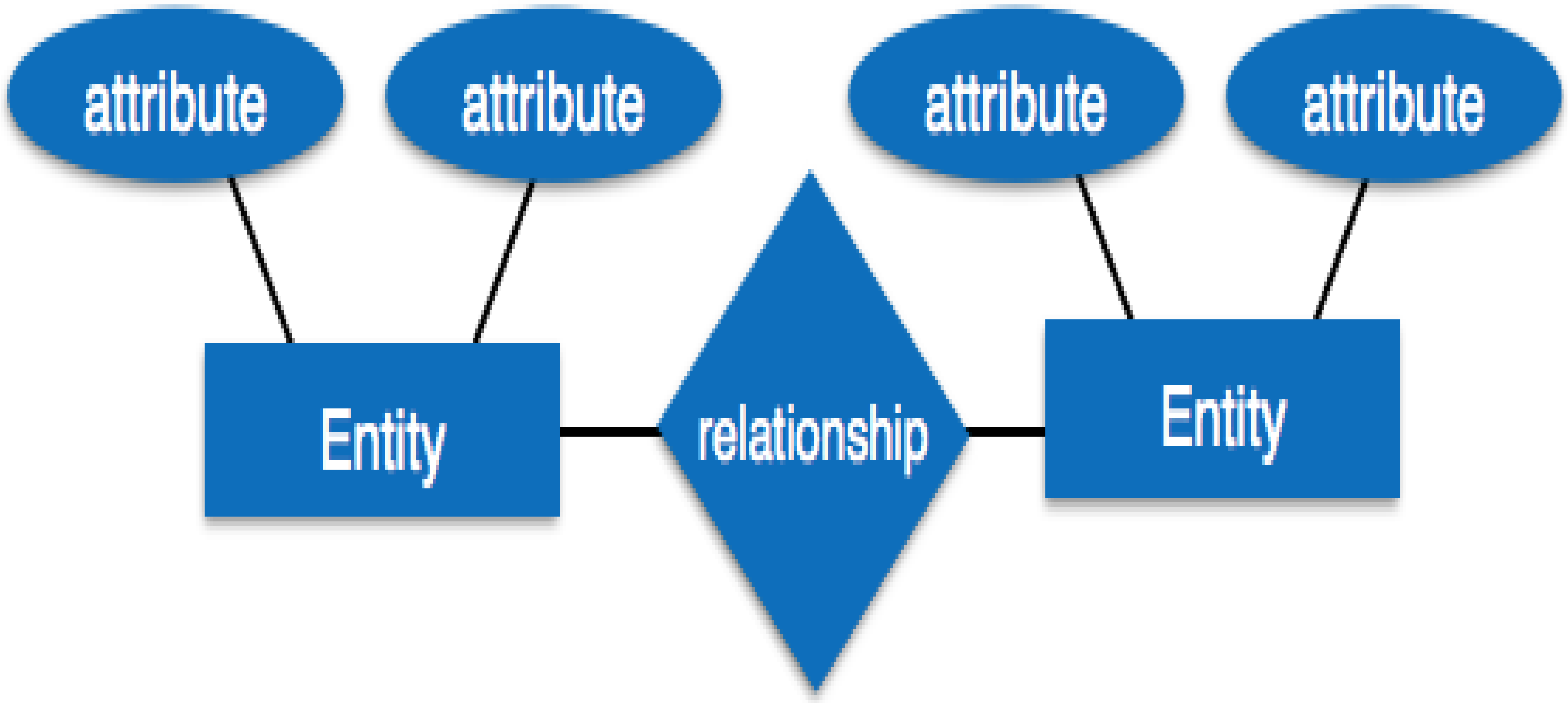
Software Analysis & Design Tools

Entity-Relationship Model

- Entity-Relationship model is a type of database model based on the notion of real world entities and relationship among them.
- We can map real world scenario onto ER database model.
- ER Model creates a set of entities with their attributes, a set of constraints and relation among them.

Software Analysis & Design Tools

- ER Model can be represented as follows :



Software Analysis & Design Tools

- **Entity -** An entity in ER Model is a real world being, which has some properties called *attributes*. **For example**, Consider a school database. Here, a student is an entity. Student has various attributes like name, id, age and class etc.

Software Analysis & Design Tools

- **Relationship -** The logical association among entities is called relationship. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of associations between two entities.

Mapping cardinalities:

- one to one
- one to many
- many to one
- many to many

Software Analysis & Design Tools

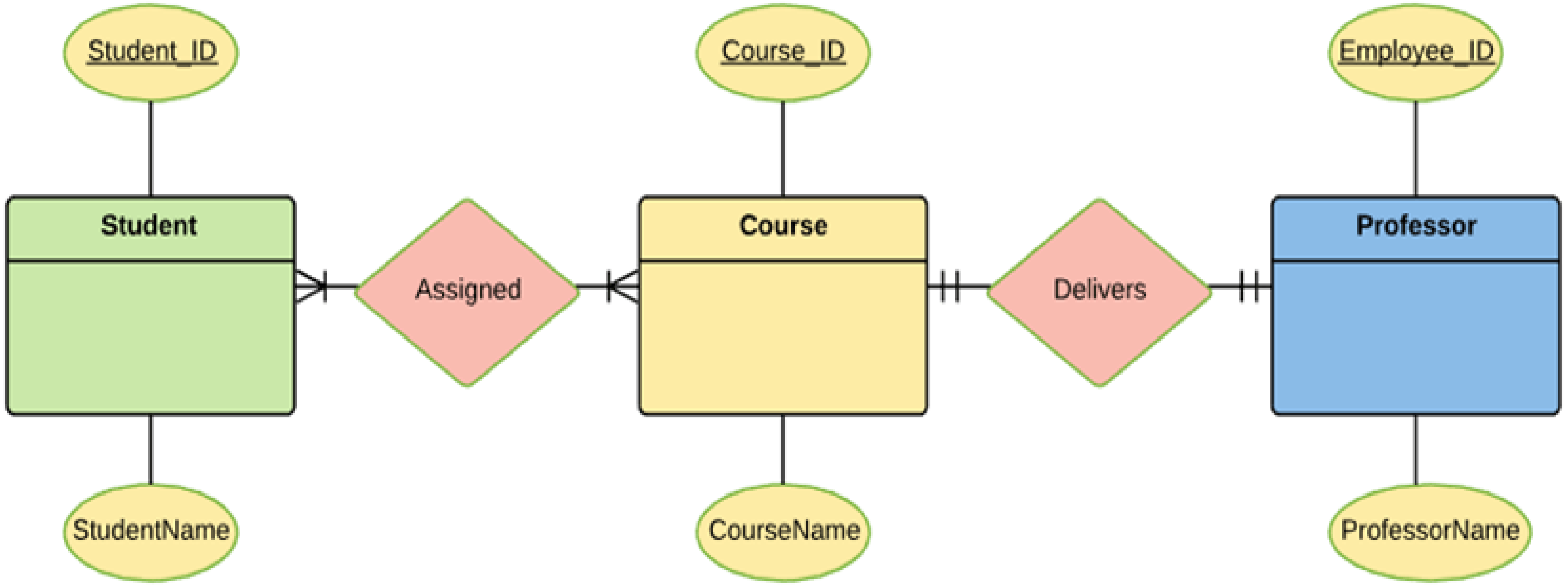
Example:

From the following two relationships

The student is **assigned** a course

Professor **delivers** a course

ER Diagram



Software Analysis & Design Tools

Self-study on ERD:

1. Learn how to identify
   i. Entity and Attribute of each entity.
   ii. Multiplicity
   iii. Relation type between entities
2. Learn how to draw an entity relation diagram from a given scenario.
3. Pseudo code

End of Lecture 3

Thanks for Listening!!!