2018-09-09

# Lesson 3: Methodology I: Analysis

## Database Application Development Methodology:

→ Will not work for software projects; this is for databases only

→ Certain assumptions are made: (1) business processes are well designed, (2) Documents are known, (3) Tasks are known, (4) System boundary is known, and (5) one DB schema unifying all views can be designed.

→ Designing databases to accomplish the aforementioned assumptions are very difficult

↳ Difficult: interests, goals, power, politics

↳ Problems with the methodology? No!

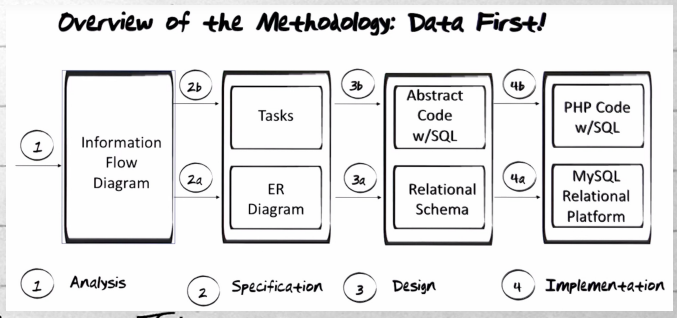
↳ Problem with the organization? Yes!

## The Software Process

→ Waterfall (Ugh!): In this course, we'll focus on (1) Analysis, (2) Specification, (3) Design, and (4) Implementation

## Overview of Methodology: Data First!

→ In DB Design, the data always comes first!




**Phase I = Analysis**: The customer will give requirements, and that will lead to an **information Flow diagram**

**Phase II: Specification**: The first diagram is the ER diagram, which represents the data modeling

**Phase III: Design**: Relational Schema with Abstract code in SQL

**Example Project Description**: All documents must be known in advance.

---

Summary: In order for the Database Application Design Methodology to work, business / user requirements must be known in advance

# Lesson 3 : Methodology I : Analysis

2018-09-10

Analysis: From the customer requirements, we create an Information Flow Diagram (IFD)

Analysis (IFD): An IFD has certain symbols and terminology

→ Ellipsis / Oval represents a Task Name, and a rectangle represents a Document Name, or screens with input / output

→ Arrows represent information flow in and out of objects

→ Dashed / broken line represents the system boundary.

→ Some documents / tasks are input, output, or both

* Rules: Never connect two documents, never connect two tasks

Input document: Something that writes to the database

Output document: Something that reads from the database

Two Fundamental Questions

=> What goes into the database?

=> What comes out of the database?

These lead to the following observations:

- Everything in DB must come from somewhere
- Everything in input documents must go somewhere
- Everything in DB must be used for something
- Everything on output document must come from somewhere