# Requirements Analysis Techniques

## Basic Structured Modeling Techniques

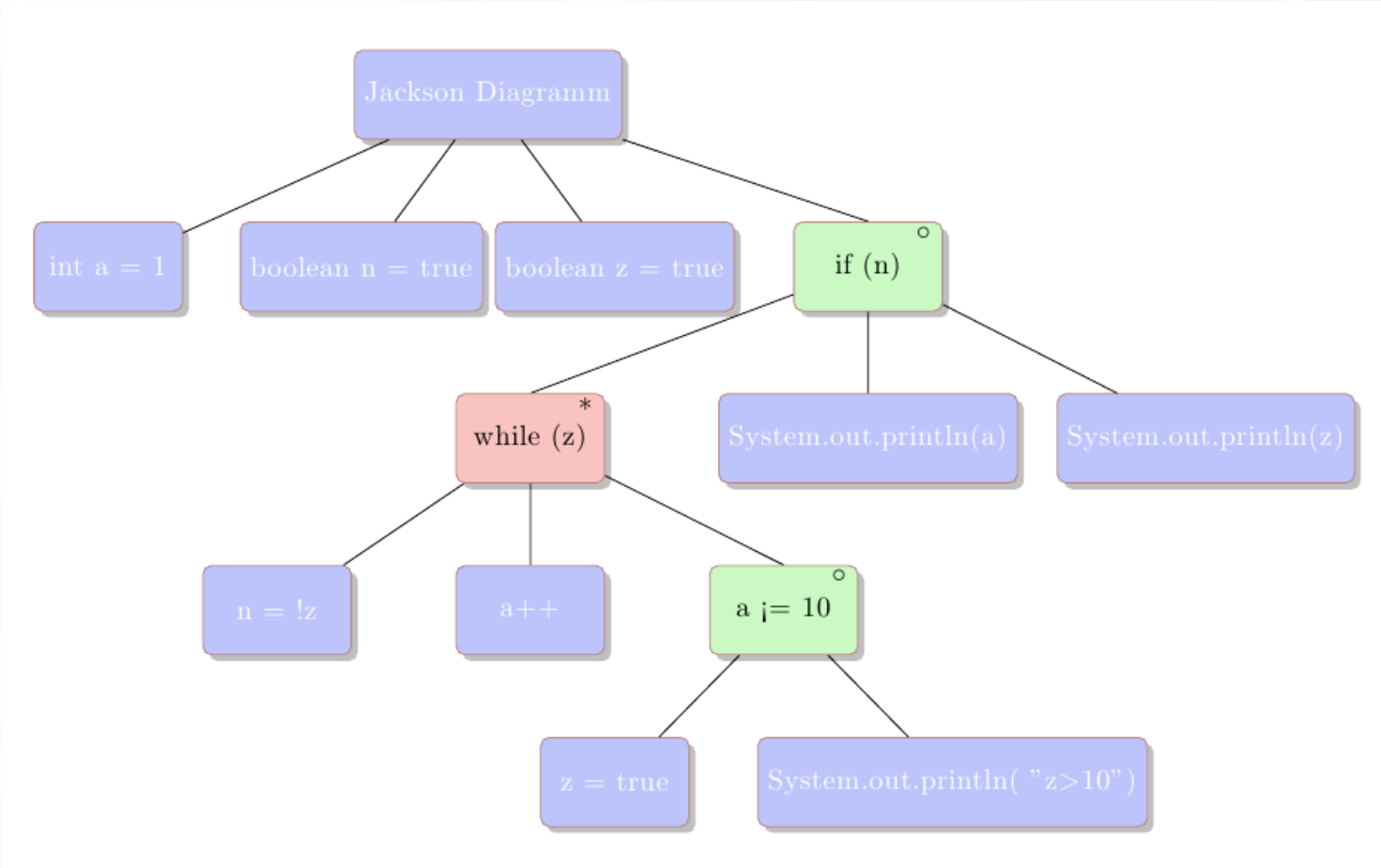
# Contents

- Introduction
- Functional Sub-model
- Data Sub-model
- Dynamic/Behavioral Sub-model
- Relationship between these (Sub)Models

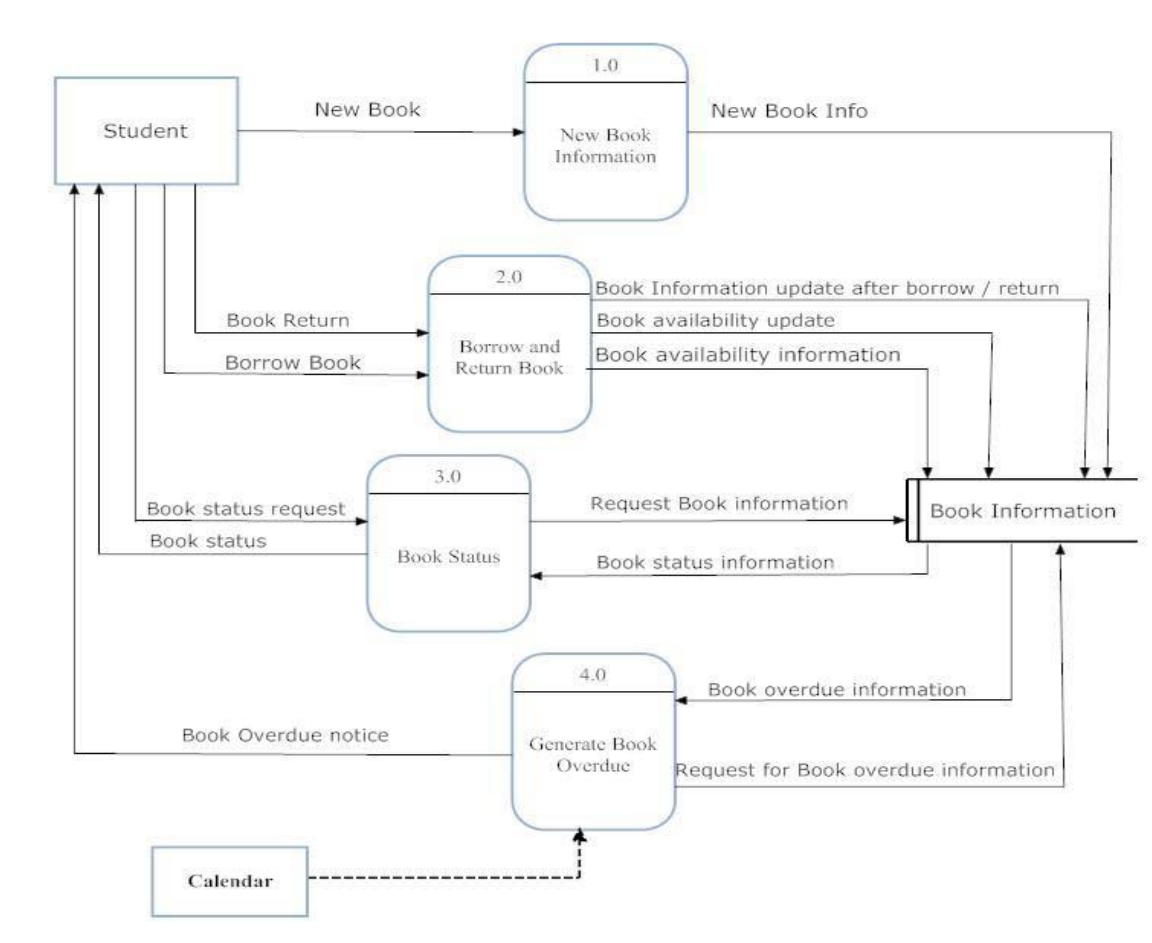
# Sub-models

- *Functional Model* describes the functionality of the systems from the user's point of view. Data Flow Diagrams (DFD), Use Case Diagrams, etc.)
- *Data Model* consists of three interrelated pieces of information: the data object, the attributes that describe the data objects, and the relationships that connect data objects to one another. Entity Relationship Diagram (ERD)
- *Dynamic/Behavioral Model* describes the components of the systems with interesting behavior. (DFD, Jackson Structure Diagrams , statechart diagrams, sequence diagrams, activity diagrams, etc.)

# Jackson Diagram



# DFD Diagram


Student
New Book
1.0
New Book Information
New Book Info
2.0
Borrow and Return Book
Book Return
Borrow Book
Book Information update after borrow / return
Book availability update
Book availability information
3.0
Book Status
Book status request
Book status
Request Book information
Book status information
Book Information
4.0
Generate Book Overdue
Book overdue information
Book Overdue notice
Request for Book overdue information
Calendar

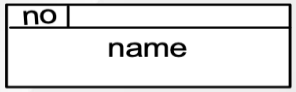
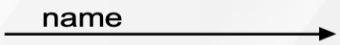
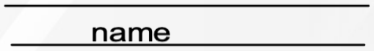
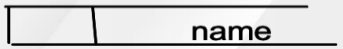
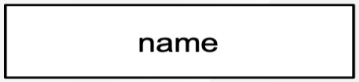
# Different Systems/Applications- Different Emphasis

- Information Systems – the emphasis on data and functional sub-models
- Process Control Systems – the emphasis on functional and dynamic sub-models

# Functional Sub-model

- Functional Model describes the functionality of the systems from the user's point of view. (DFD, use case diagram, etc.)
- The main basic modeling technique is Data Flow Modeling. It involves the development of Data Flow Diagrams (DFDs) and (in addition) Elementary Process Specifications and Data Flow and Data Store definitions (part of Data Dictionary)
- In Structured System Analysis & Design Methodology(SSADM) this is followed by Function Definition and Conceptual Process Modeling (ECDs, EAPs, UPMs, EPMs - to specify elementary processes in more detail). ECD:*Effect Correspondence Diagram,* EAP: *Enquiry Access Path,* UPM: *Unconstrained Process Model* , EPMs: *Entity Process Models*

# Data Flow Diagrams (DFDs) - Main Elements (notation)

| | Yourdon | SSADM |
|---|---|---|
| • **Process** |  |  |
| • **Data flow** |  | |
| • **Data store** |  |  |
| • **Terminator (external entity)** |  |  |

# DFDs Main Elements (meaning)

- **PROCESS -** processes show what systems do – the functional requirements. Each process has one or more data inputs and produces one or more data outputs (there are some exceptions but very rare!)
- **DATA FLOWS** model the passage of data in the system
- **DATA STORE (FILE)** is a repository of data. It contains data that is retained in the system (persistent data). Processes can enter data into a data store or retrieve data from the data store.
- **TERMINATORS / EXTERNAL ENTITIES** are outside the system, but they supply input data into the system and/or use the system output e.g other systems, other organizations, system users, departments outside the system, etc

# DFDs - Basic Rules 1

**Data flows can be drawn between:**

- two processes (only when they are 'tightly' coupled i.e. they are 'executed' in the same, relatively short time 'frame')
- a data store and a process
- a process and a terminator

**Data flows** must not be drawn between two data stores !!!

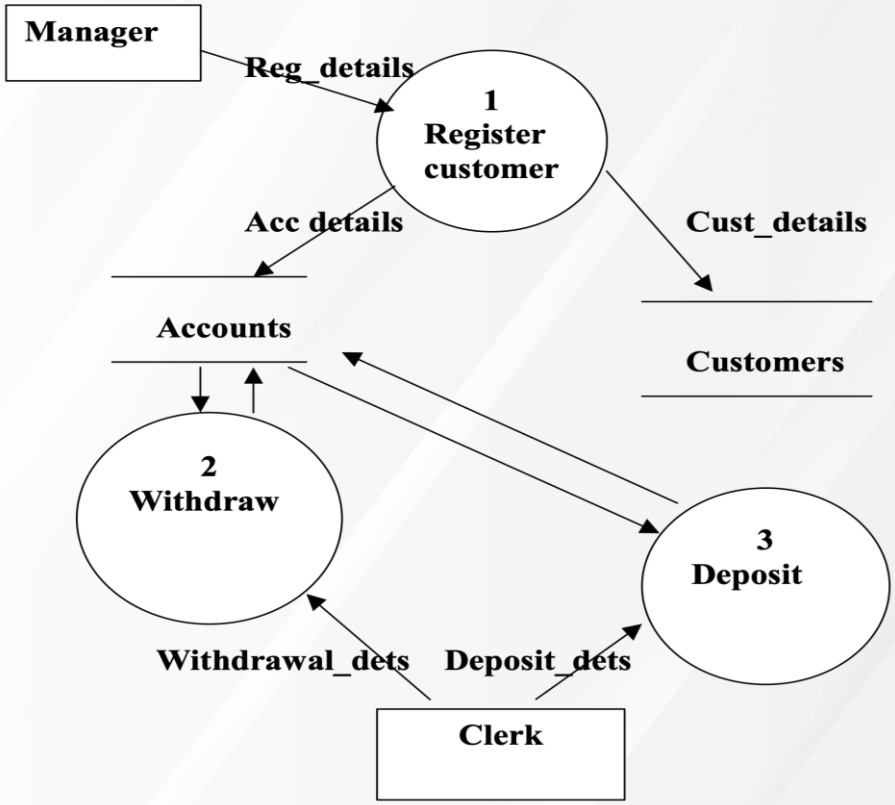
**Avoid** (if possible) flows between terminators and data stores, and between two terminators.

# DFDs - Basic Rules 2

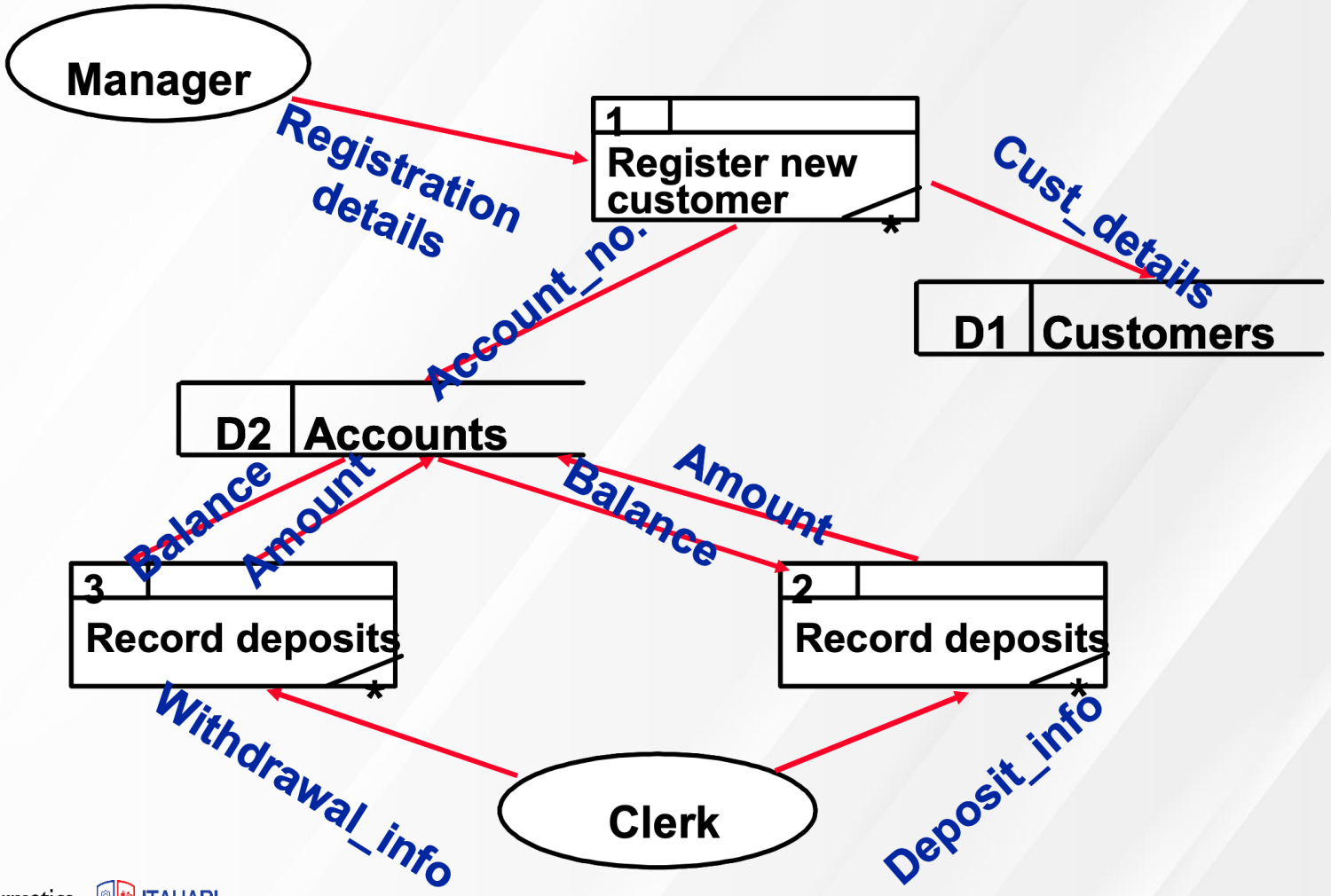
**'Dodgy' Situations (to be double checked) are:**

- processes with inputs only
- processes with outputs only
- data stores with inputs only
- data stores with outputs only
- does each process receive all the data it requires?
- are all main data flows labelled?

# Simple DFD - Example (Yourdon's Notation)



# DFD - SSADM Notation



# DFDs - 'Leveling'

- The idea of 'leveling' is  to present the system at different levels of detail.
- The top-level diagram (level 0) is known as the context diagram (it models the whole system as a single process).
- The level 1 DFD gives an overview of the whole system, identifying the major processes, data flows and data stores.
- Processes from level 1 can be expanded to show more detail (if necessary) at level 2. Processes from level 2 can be further expanded, etc.

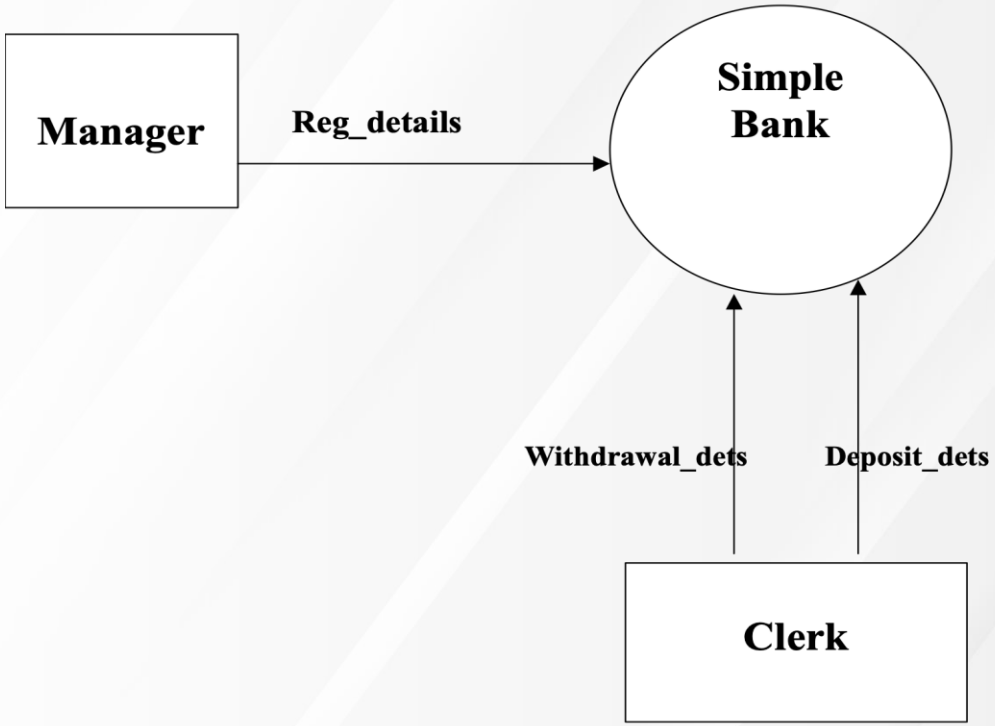
# ‘Levelling’ (cont.)

- In the classical approach DFDs are built in a top-down fashion. First of all the context diagram is produced, next it is refined/partitioned into lower-level DFDs.
- In a lower-level DFD, a process (from a higher level) is expanded to reveal more details. The numbering convention for ‘parent’ and ‘child’ processes is as follows: if the parent’s no is e.g. 5 then its ‘children have numbers 5.1, 5.2, 5.3, etc.
- Typically when we get to the point at which the operation of a process can be described in about half page using a pseudo code/structured English, We can stop partitioning.
- Processes which have not been partitioned (decomposed) are called ‘atomic’ or elementary processes. They belong to a ‘bottom’ level.

# ‘Levelling’ - Balancing Rules

- Balancing rules require that data flows entering or leaving a parent diagram (i.e. a process at a higher level) must be equivalent to those on the ‘child’ diagram. It means that new flows may not be arbitrarily introduced at lower, more detailed levels !

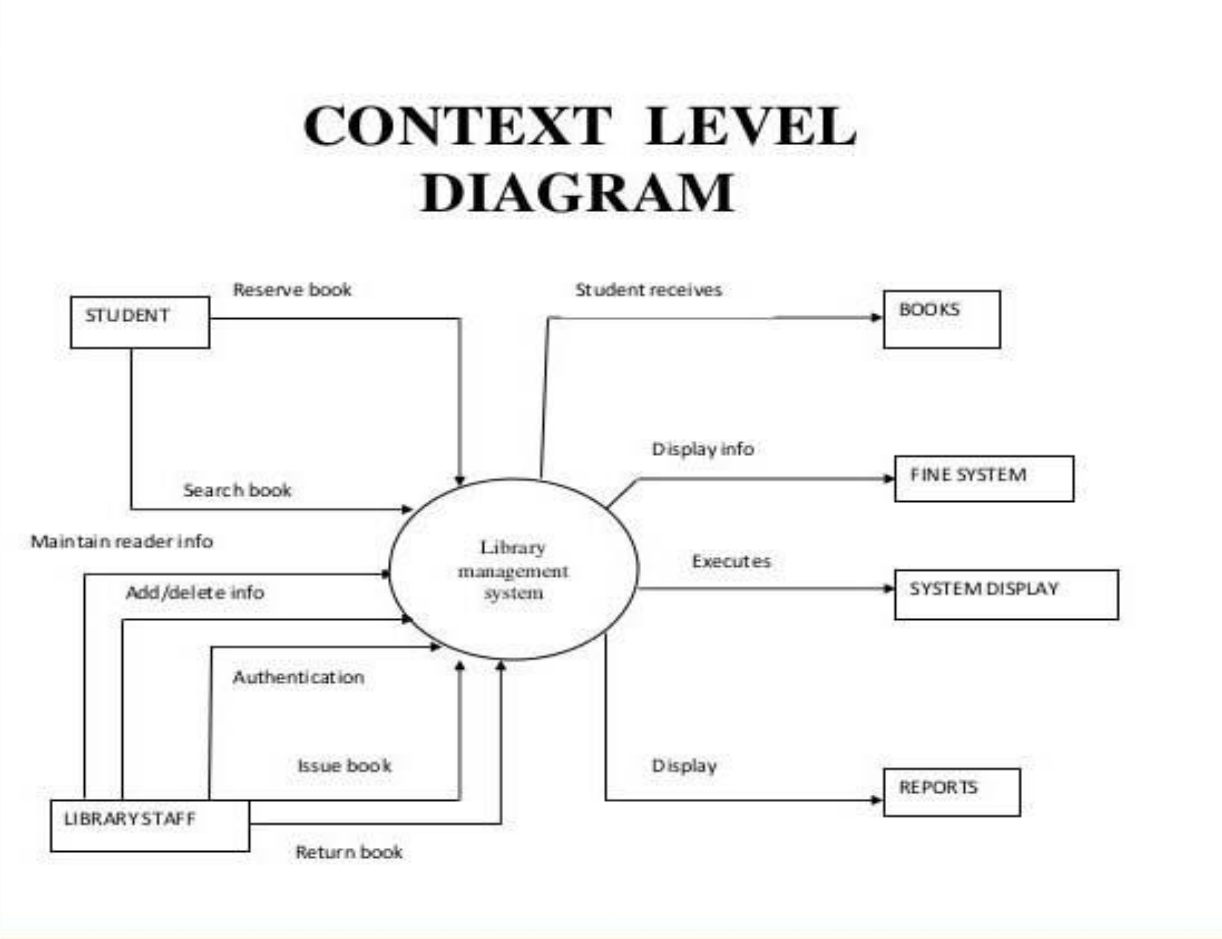
# Simple Bank - Context DFD i.e. Level 0



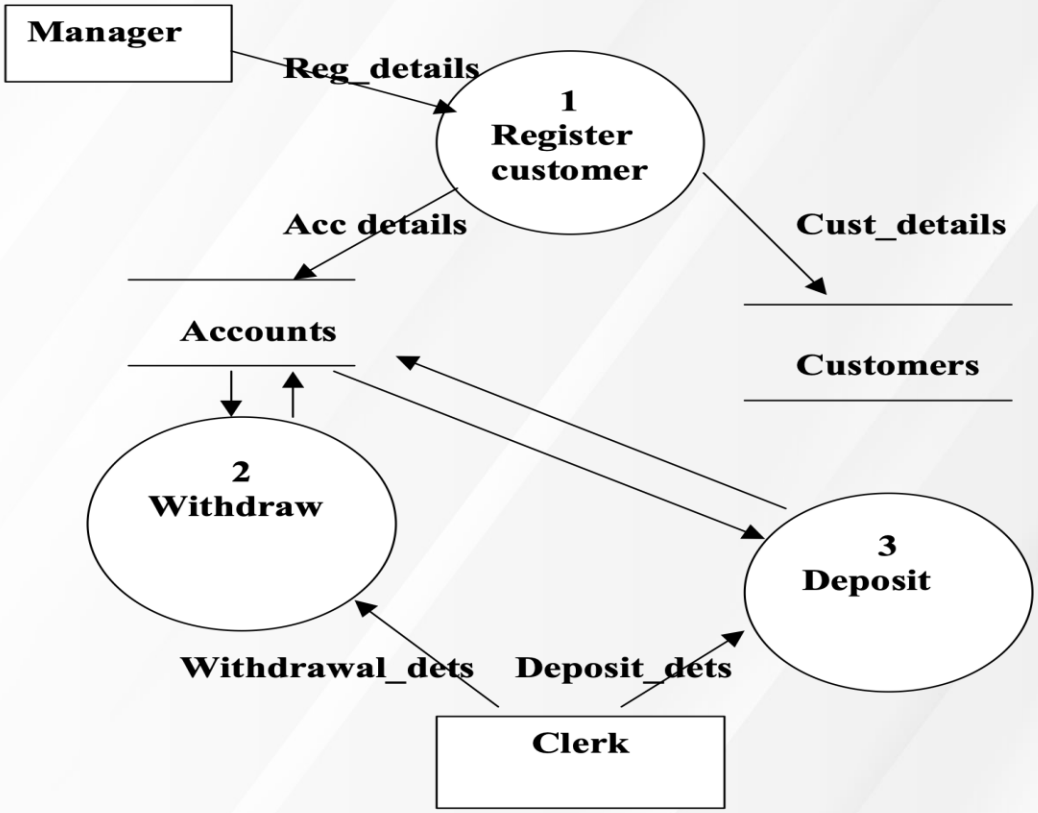
# Context Level Diagram

- It is the Highest Level of DFD
- It contain only One Process
- Process must name with System Name
- All External Entities will be shown at Context Level
- Only Major Data Flows will be shown at Context Level
- Process much be numbered with 0 (Zero)

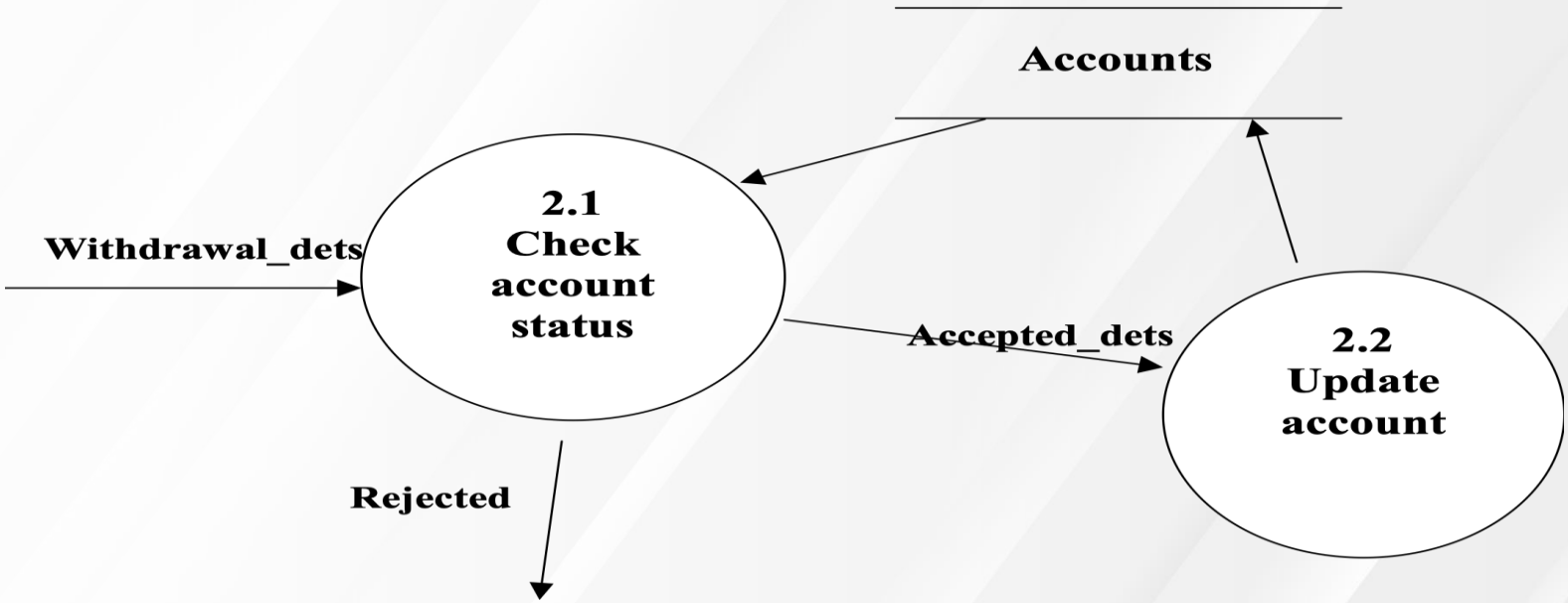
# 0 Level DFD Diagram



# Simple Bank - Level 1 DFD



# Simple Bank - Level 2 DFD



# Elementary Process

Specification/Definition

Notations

- Structured English/pseudocode (the most popular)
- Decision tables
- PRE/POST conditions
- Diagrams e.g. Jackson-like diagrams in SSADM

# Example of Process Specification

using Structured English/pseudocode for Register customer i.e. Process 1

Get Reg_details
Generate Account No
Customer Record = Cust Name, Cust Addr, Cust Tel No.
Write Customer Record to Customers
Balance = 0
Account Record = Account No., Balance, Curr Date, type
Write Account Record to Accounts

# Structured English/pseudocode Notation

- It combines the precision of a formal programming language (e.g. C, Pascal, Java) with the casual informality and readability of the English language (or any natural language)
- It means that it provides the keywords to structure process specification logic (e.g. IF, ELSE, WHILE, etc) while leaving some freedom when describing the activities and data used in the process

# Structured English/pseudocode Notation (cont.)

Possible notation (based on Java);

- Boolean and arithmetic operators
- = assignment operator
- Control structures such as: WHILE, IF ELSE, DO WHILE, etc
- Indentation is used to show groups/blocks of statements/ activities e.g. within WHILE structure
- Get, Display (or Input,Output) to input/output data
- Read,Write to read and write records of data (from and to data stores)

# The Data Dictionary

- The data dictionary is an organized listing of all the data that are pertinent to the system with precise, rigorous definitions.
- A typical data dictionary includes definitions/ descriptions of: data flows, data stores and entities (from the data sub-model)
- As data flows and data stores are ‘made up’ of data structures and data elements, these data structures and elements also should be described/defined!

# **Data Dictionary** - notations

- There are many different ‘data dictionary notations’.
- All are ‘structured’ notations i.e. they allow to specify sequences, iterations and ‘selections’ of data elements and data structures
- Examples of some notations: Jackson diagrams, BNF (Backus Naur Form ) notation, etc

# Data Dictionary – BNF Notation

The notation is summarized below:

| | |
|---|---|
| = | is composed of (definition) |
| + | and (sequence of data) |
| { } | iteration/repetition of data |
| [ \| ] | selection (either or) of data |
| * * | comment |
| () | optional data |

# Data Dictionary

Examples of BNF definitions:

Reg_Details = Cust Name + Cust Addr + Cust Tel No. + Account No. * data flow*

Cust_Details = Cust Name + Cust Addr + Cust Tel No.

Accounts = {Account} *data store*

Account = <u>Account No.</u> + Balance + Date Opened *entity*

# Data Sub-model

- Data Model consists of three interrelated pieces of information:  the data object, the attributes that describe the data objects, and the relationships that connect data objects to one another. (ERD)
- The main modeling technique is Entity Relationship modeling (Logical Data Modeling - LDM in SSADM). It involves the development of Entity Relationship Diagrams (E-R diagrams, ERDs, LDS in SSADM) and Entity Definitions/Descriptions (part of Data Dictionary)

# Entity Relationship Modeling- Entity

- Entity (or an Entity type) – any object or concept about which a system needs to hold information.
- Examples of entities;

  Library system: Book, Reader, Loan, Reservation, etc

  Doctor's surgery system: Doctor, Nurse, Patient, Appointment, etc
- Each entity (Entity type) must have a number of real-world occurrences (instances). For example in a typical Library system (see above) there are many instances of Book, Reader, Loan, etc.

# Entity Relationship Modeling- Entity (Cont.)

An entity must have a number of properties to qualify as such:

1. There must be more than one occurrence/instance of the entity.
2. Each occurrence/instance should be uniquely identifiable.
3. There must be data that we want to hold about the entity.
4. It should be of direct interest to the system.

# Entity Relationship Modelling- Entity (Cont.)

- Each data item that we hold about an entity is known as an ATTRIBUTE. For example some of the attributes of Book might be: Book Number, Title, Year, etc
- The symbol for an entity in an ERD is a round-cornered rectangle with the entity's name

# Entity Relationship Modeling- Relationships

- Relationship – a logical association between two entities (or between an entity and itself)
- In physical data structures these relationships are represented by physical links such as pointers, etc
- In logical models relationships represent business associations or rules and NOT physical links!

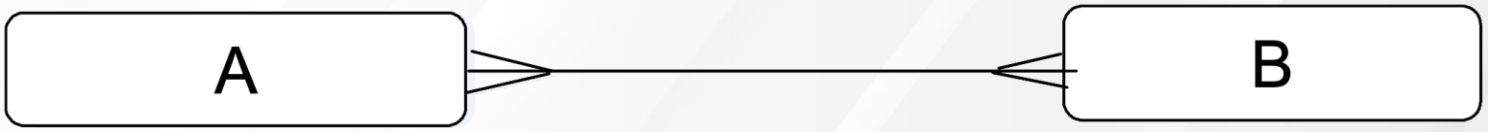
## Entity Relationship Modeling – Relationships (Cont.)

- Relationships have two important properties:
  - *Degree*
  - *Optionality*
- Degree – the number of occurrences of each entity type participating in a given relationship
- Optionality – if there can be occurrences of one entity that are not related to at least one occurrence of the other, then the relationship is said to be *optional* for that entity.

# Entity Relationship Modeling- Relationships (Cont.)

There are three types of degree:

1.Many-to-many (m:n)




Each occurrence of A is related to *one* or *more* occurrences of B, and each occurrence of B is related to *one* or *more* occurrences of A

**For ex: many students can join many subjects**

# Entity Relationship Modeling- Relationships (Cont.)

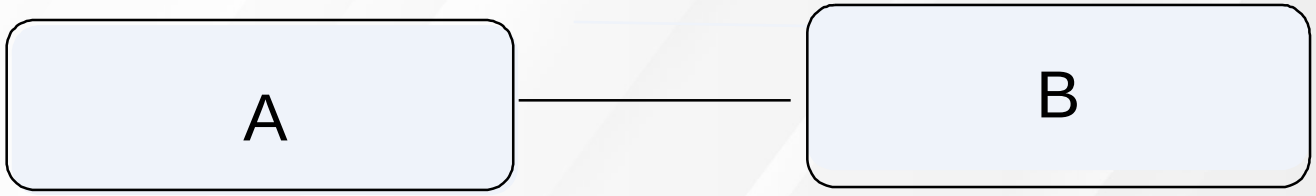
2. One - to - many (1:m)


Each occurrence of A is related to *one or more* occurrences of B, but each occurrence of B is related to *only one* occurrence of A

**For ex: each customer can have many sales orders**

# Entity Relationship Modeling – Relationships (Cont.)

3. One-to-one (1:1)




Each occurrence of A is related to *only one* occurrence of B, and each occurrence of B is related to *only one* occurrence of A

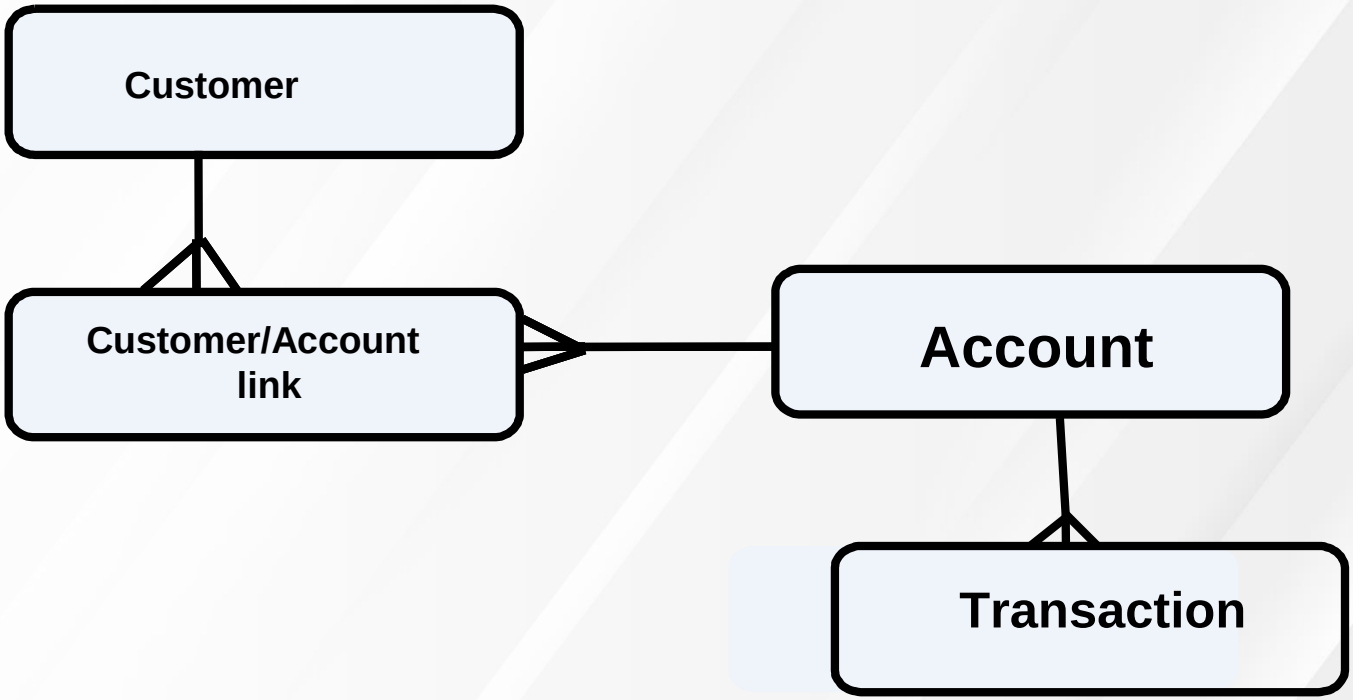
**For ex: One passport number by one citizen**

# Entity Relationship Modeling –Relationships (Cont.)

- Optionality – if there can be occurrences of one entity that are not related to at least one occurrence of the other, then the relationship is said to be *optional* for that entity.


- Each A *may* be related to one B; Each B *must* be related to one A
- Dashed line denotes optional relationship

# E-R Diagrams (Using SSADM Notation)



# Entity Definitions/Descriptions

- In some organizations only brief descriptions/definitions will be required, while in others they may consist of several sections including details of attributes, relationships, volumes of data, user access, etc
- For CS5002 brief descriptions/definitions will be used which list attributes and show primary keys. These descriptions can be included in Data Dictionary (together with definitions of data flows and data stores).

# Examples of Simple Entity Definitions

Customer = Cust No + Cust Name + Cust Addr +Cust Tel No….

Account = Account No. + Balance + Date Opened….

alternatively they can be specified as follows e.g:

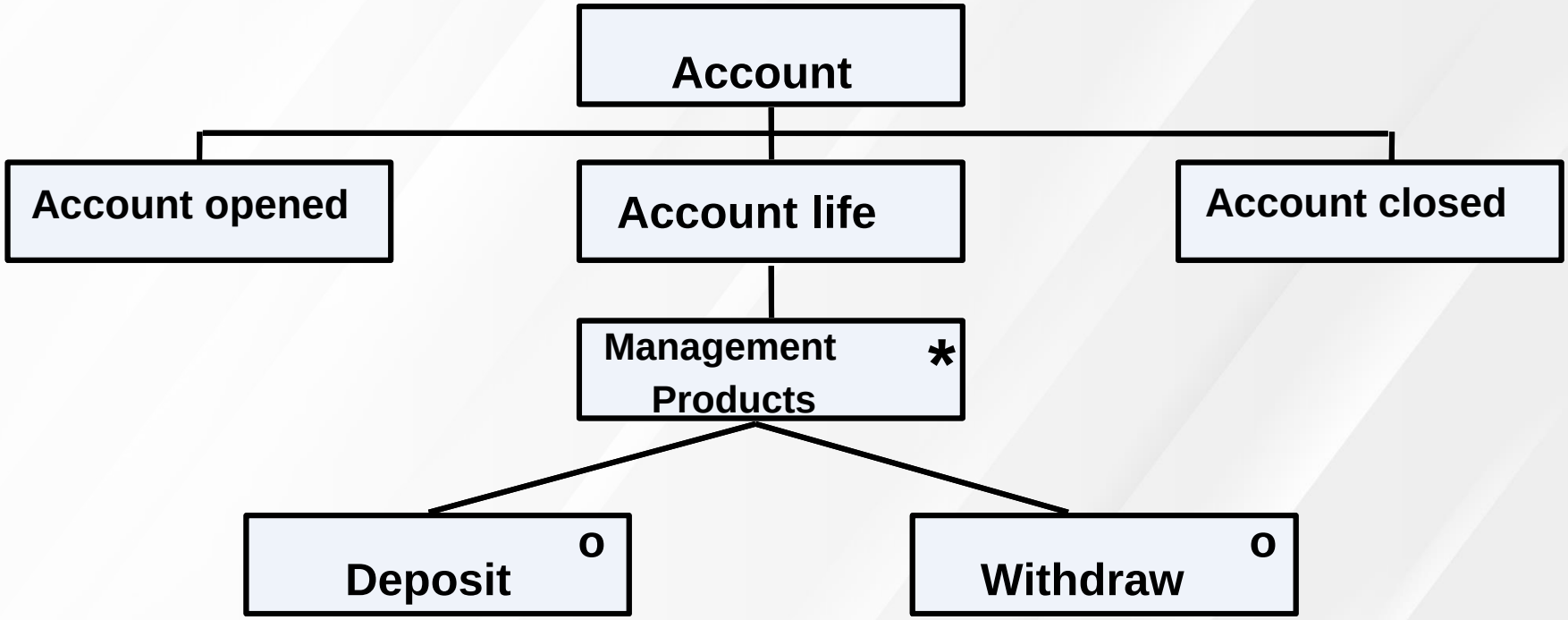
CUSTOMER

| | |
|---|---|
| Cust No | Cust name |
| Cust Addr | Cust Tel No |

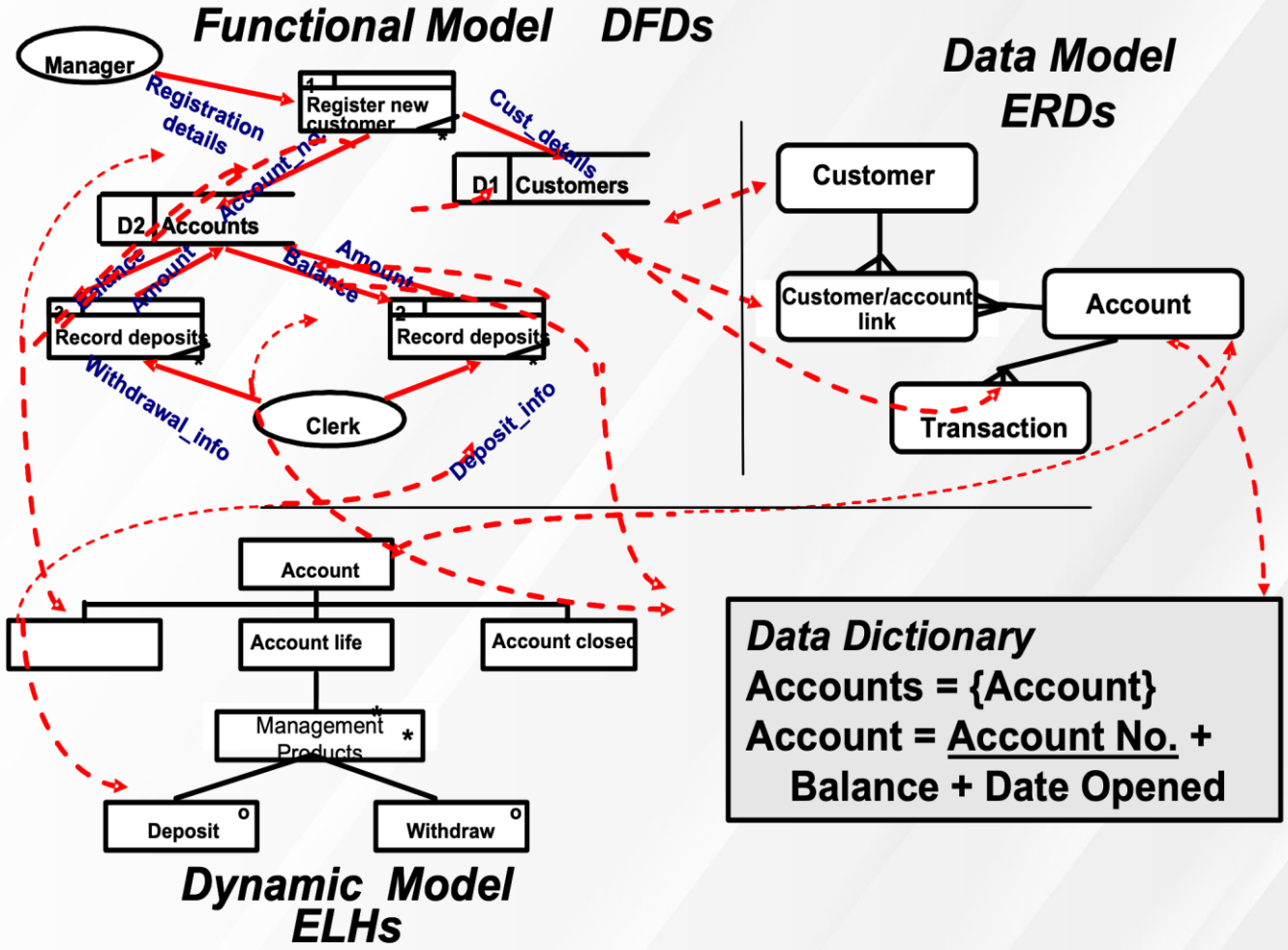
# Dynamic Sub-model

- Dynamic Model describes the components of the systems that have interesting behavior.
- Modeling techniques for this sub-model include Entity Life Histories (ELHs), State Transition Diagrams (STDs) etc. in O.O. statechart diagrams, sequence diagrams, activity diagrams, etc.

# Entity Life Histories (Using Jackson Notation Adopted in SSADM)



# Relationships between Sub-models



# Summary

- By looking at the system from three different perspectives we learn more about the system itself.
- Each sub-model represents a particular aspect of the system.
- By checking one sub-model against another, serious errors can be detected.