Topics

- ✓ Introduction
- ✓ Software Engineering Models
- ✓ BPM, BRMS, SOA
- ✓ Software Requirements Engineering
- ➢ Software Analysis Models

6. Software Project Management
7. Software Design Concepts, Principles and Models
8. Software Coding Practices
9. Software Testing Techniques
10. Software Quality Assurance
11. Emerging Trends in Software Engineering

Why Software Analysis?

- Software Analysis enables one to eliminate or reduce ambiguous, incomplete, inconsistent statements from Software Requirements Specifications

- It does the above by providing two-dimensional diagrams, that can be discussed with ease with the business users

- It provides the necessary and convenient conduit between Software RS and Software Design

Why Software Analysis?

Have you got a complete picture of the following specifications?

- The 5-BHK bungalow should have a door to the north-east corner of the ground-floor drawing room, which opens inside along the north-wall and just coincides with edge of the bay-window of trapezoidal shape with the smaller parallel side protruding to the north of the north wall by 2.5 feet; there should be another bay-window in the drawing room with a gap of 5 feet with the above mentioned bay-window and whose west edge coincides with the north-west door that opens inside along the north wall; . . . (specifications continue)

An analysis of the above specifications of a customer, using the 'plan and elevation' ***diagrams*** will offer much better clarity to both the customer and the builder!

Software Analysis Models

- Dataflow Diagrams
- Use-Case Analysis
- Use-Case Activity Diagrams
- State Diagrams

What do Data Flow Diagrams Offer?

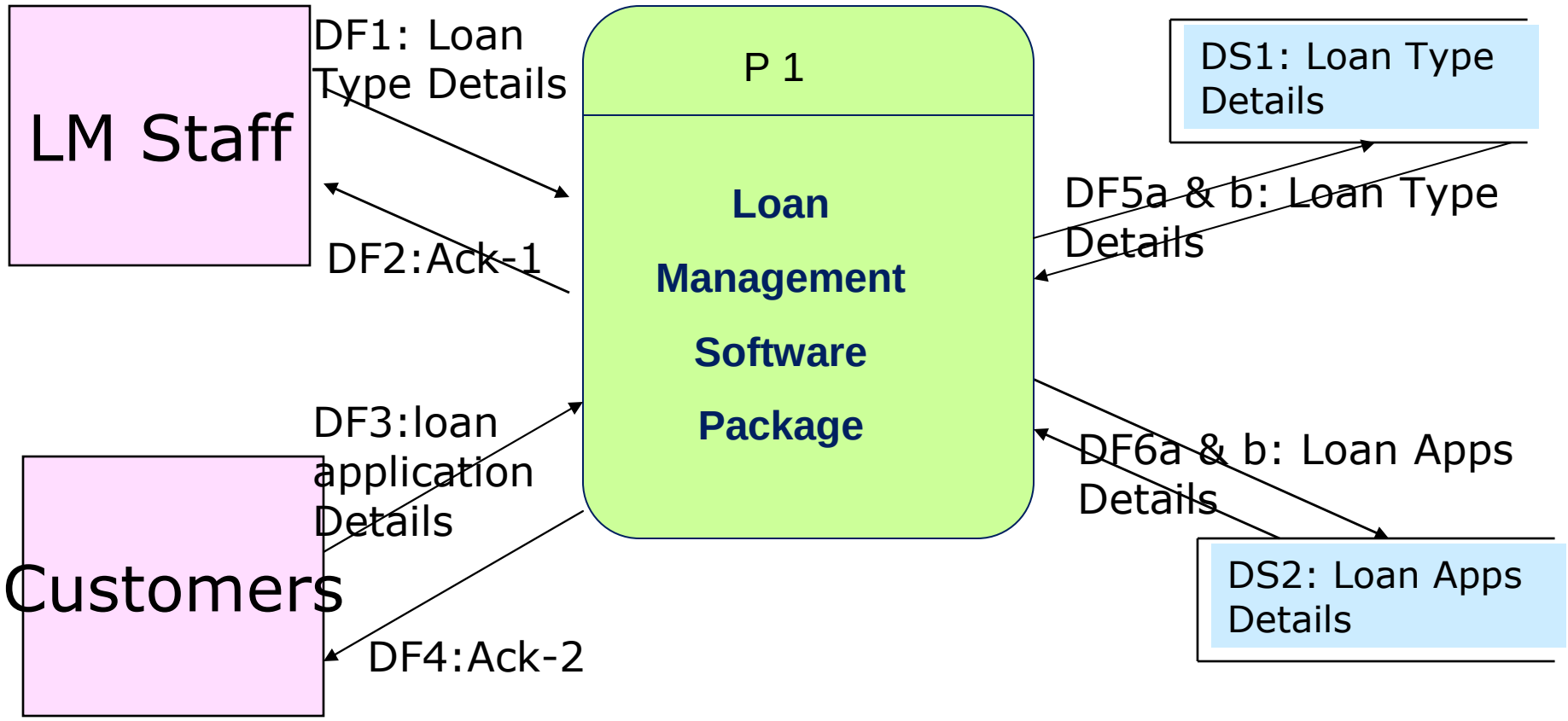
- A diagrammatic view of the flow of data through the business operations (which are being automated)

- Why the flow of data is important for building a software package?

Systems Analysis Using DFDs

- A software package,
  - **Receives** information / data from Entities of the real-world (e.g. it receives loan-type-details from Loan-Management-Staff, loan-application-details from Bank Customers, etc.)
  - **Receives** information / data from Data-Stores (e.g. get loan-type-details from the Data-Store before they are modified, etc.)
  - **Processes** that information / data as per the steps of the business (e.g. validate a new loan-type-details, under-write an existing loan-application-details etc.)
  - **Outputs** information / data to the real-world entities (e.g. informs Loan-Management-Head of the pending approvals, Bank Customers about the status of their loan-application, etc.)
  - **Writes** information / data onto Data-Stores (e.g. once a loan-application is submitted, it writes the details onto its Data-Store for future processing)

Example of a Data Flow Diagram



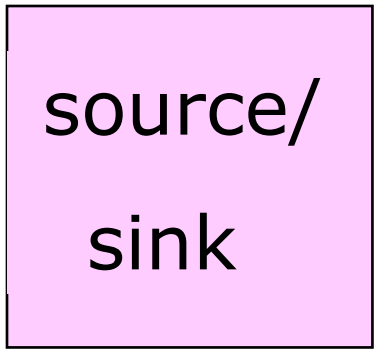
Systems Analysis Using DFDs

- Focus is the *logical* view of the system, not the physical
- "What" the system is to accomplish, not how

- Tools:
  - data flow diagrams
  - data dictionary
  - process specification
  - entity-relationship diagrams

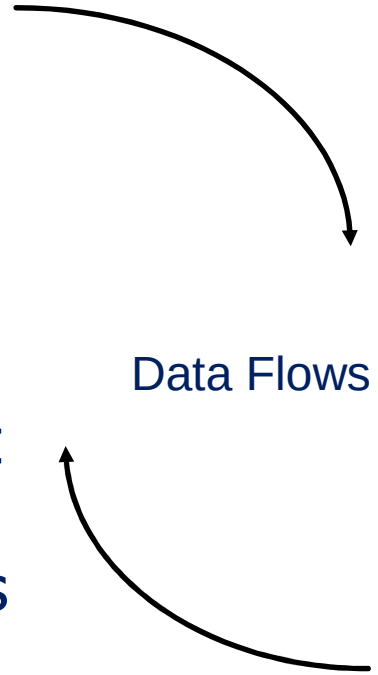
Sources/Sinks (external entities)

- Any class of people, an organization, or another system which exists outside the system you are studying.
- Form the boundaries of the system.
- The system and external entities exchange data in the form of data flows.
- Must be named, titles preferred to names of individuals - use a noun



Data Flows

- data in motion
- marks movement of data through the system - a pipeline to carry data
- connects the processes, external entities and data stores
- Unidirectional
- originate OR end at a process (or both)
- name as specifically as possible - reflect the composition of the data - a noun
- do not show control flow! Control flow is easy to identify- a signal with only one byte - (on/off).
- HINT: if you can't name it: either it's control flow, doesn't exist or you need to get more information!



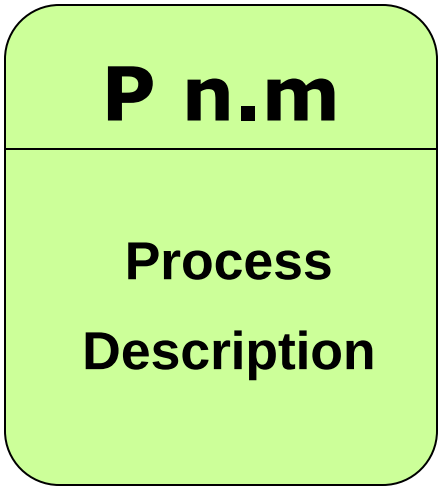
Processes

- transform incoming data flows into outgoing data flows
- represent with a bubble or rounded square
- name with a strong VERB/OBJECT combination; examples:

  create_exception_report

  validate_input_characters

  calculate_discount



Data Stores

- data at rest
- represents holding areas for collection of data, processes add or retrieve data from these stores
- name using a noun (do not use 'file')
- only processes are connected to data stores
- show net flow of data between data store and process. For instance, when access a DBMS, show only the result flow, not the request

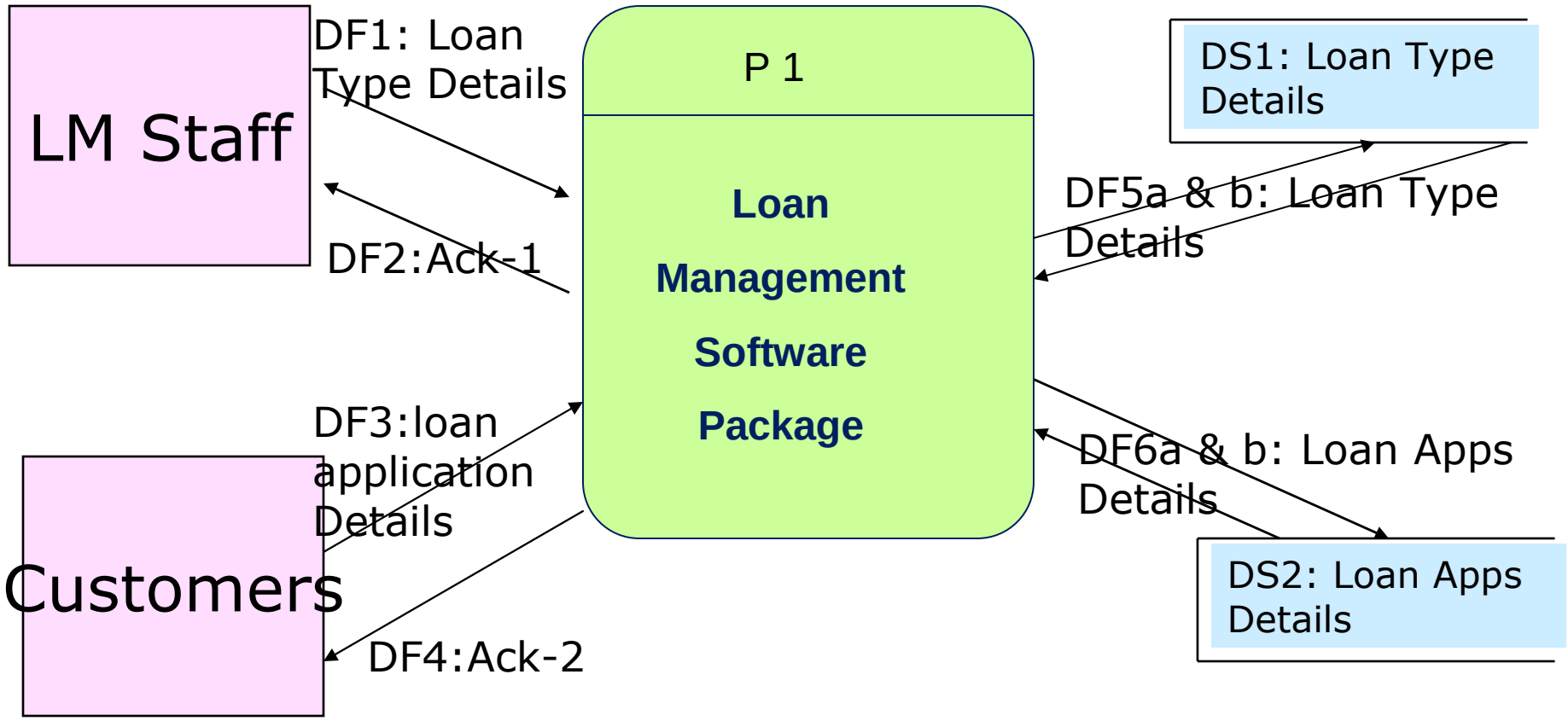
Different Types of DFDs

- Level-0 diagram (context diagram)
- Level-*n* diagram

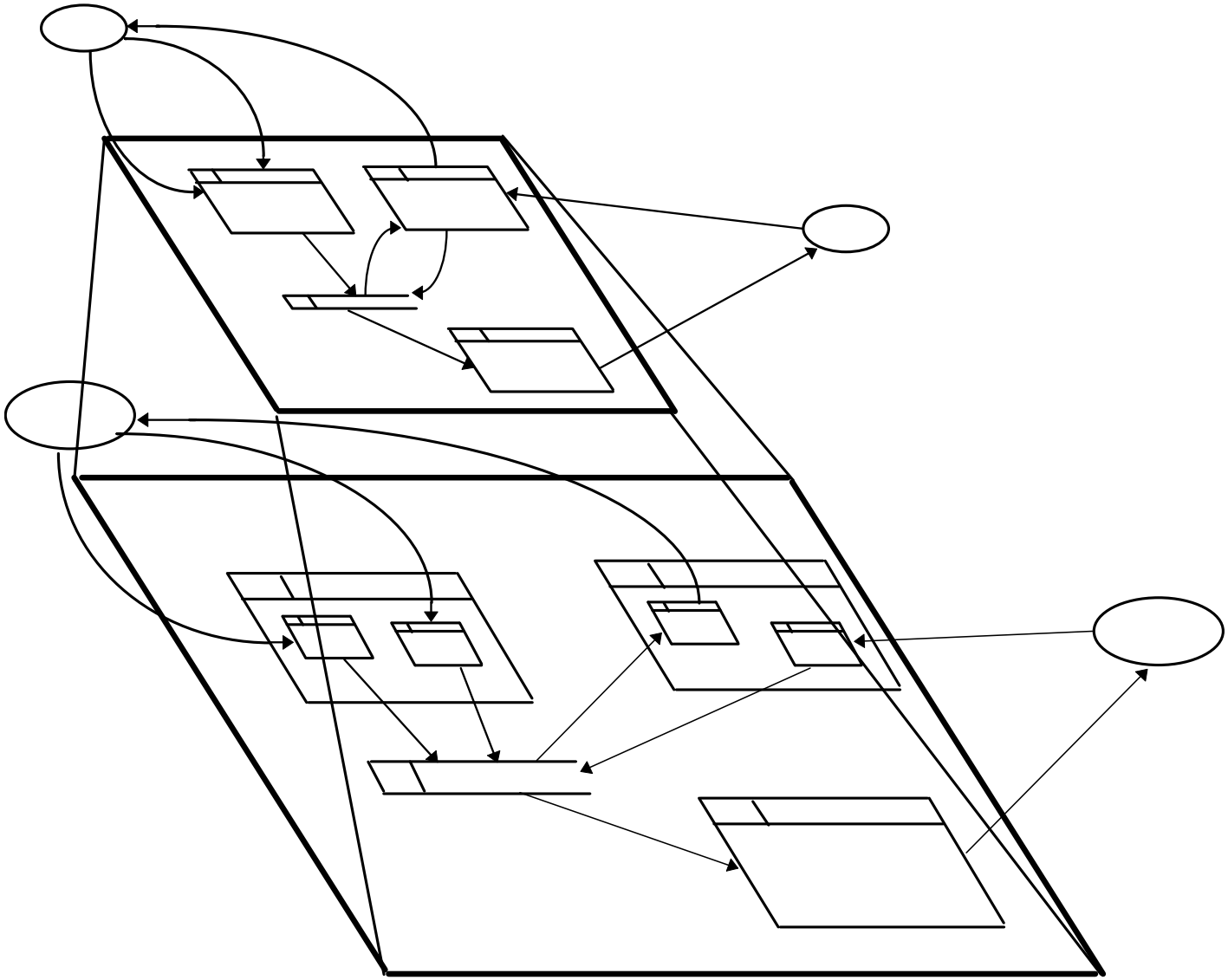
Drawing a Level-0 Diagram

- List the Categories of Business Users who interact with the Software Package
- List the major data stores, whose data is required to carry on the automated operations
- Show the entire Software Package as a single Process Box with number as P 1

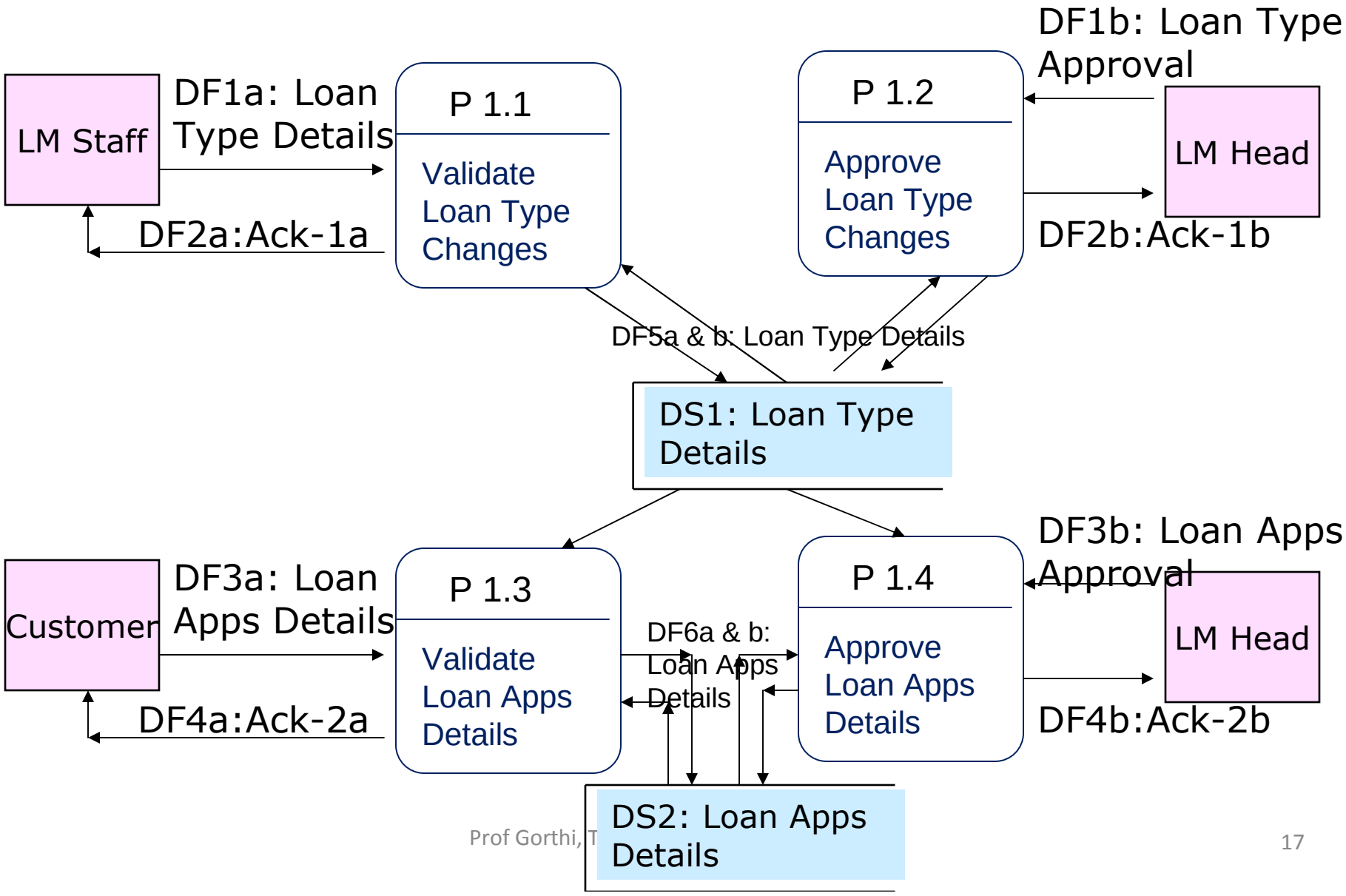
Level – 0 DFD


LM Staff
DF1: Loan Type Details
DF2:Ack-1
P 1
Loan Management Software Package
DS1: Loan Type Details
DF5a & b: Loan Type Details
DF3:loan application Details
DF4:Ack-2
Customers
DF6a & b: Loan Apps Details
DS2: Loan Apps Details

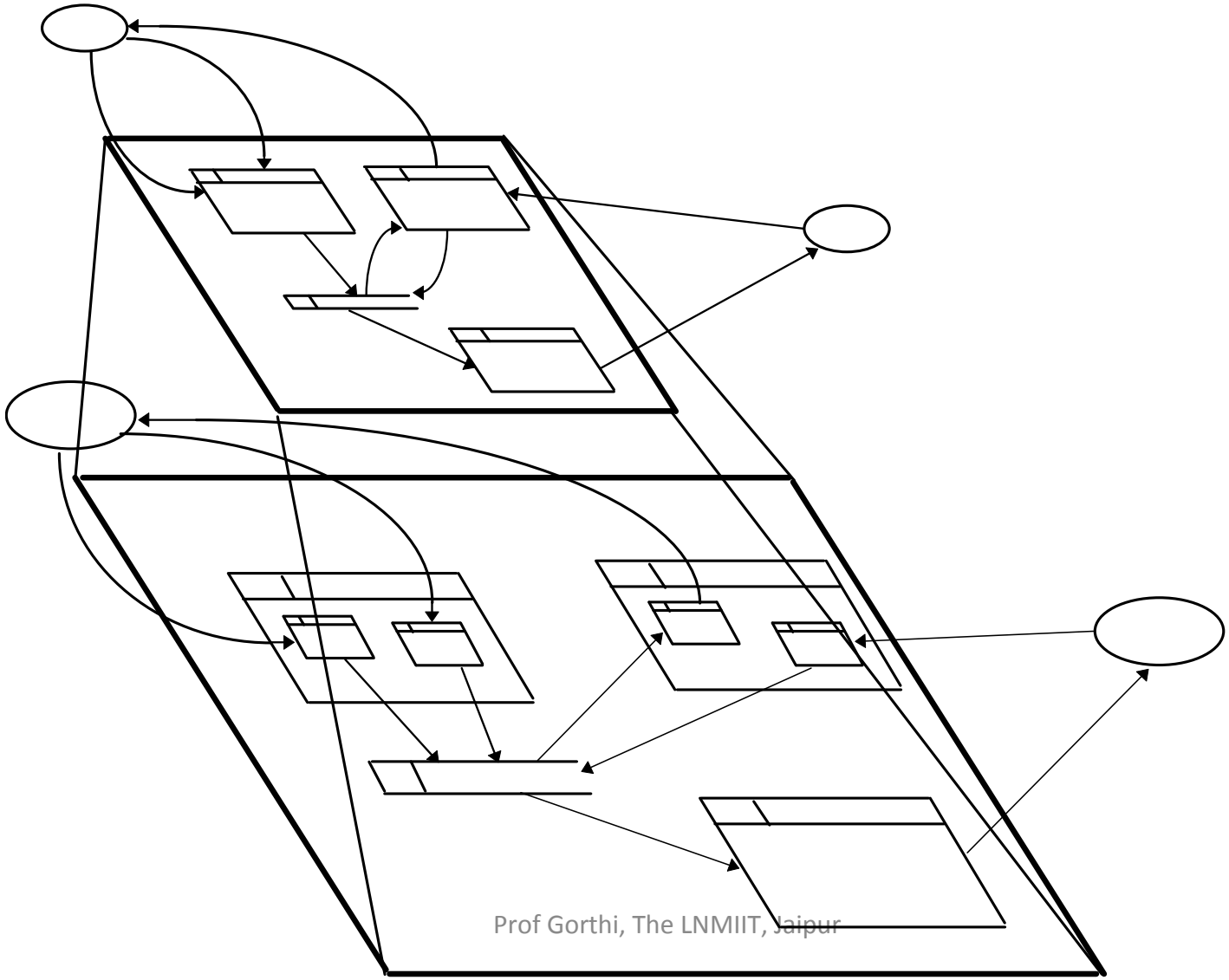
Decomposing the Level-0 DFD



Level – 1 DFD


LM Staff
DF1a: Loan Type Details
DF2a:Ack-1a
P 1.1
Validate Loan Type Changes
P 1.2
Approve Loan Type Changes
DF1b: Loan Type Approval
LM Head
DF2b:Ack-1b
DF5a & b: Loan Type Details
DS1: Loan Type Details
Customer
DF3a: Loan Apps Details
DF4a:Ack-2a
P 1.3
Validate Loan Apps Details
DF6a & b: Loan Apps Details
P 1.4
Approve Loan Apps Details
DF3b: Loan Apps Approval
LM Head
DF4b:Ack-2b
DS2: Loan Apps Details

Continue to Decompose Data Flow Diagrams till you reach unit level processes . . .



At end, Describe each Process, DF & DS

Process Description

Unit Levl Process Description
System: *Loan Management System ;* **DF-In:** *DF-1a* **DF-Out:** *DF-2a*
Process Name: *Validate Loan Type Changes* ***Process Id:*** *P 1.1*
(1) Loan Management Staff can (a) create a new loan type or (b) modify /or (c) de-activate an existing loan type; *(2) They do so by accessing this software, choosing the options (a) or (b) or (c) and filling-in the Loan Type Details;* *(3) The process P 1.1 will validate each field of the Loan Type Details (creation or changes) and if there are any errors, it will display the same to the LM Staff;* *(4) The LM Staff will then correct the errors and re-submits the details;* *(5) If there are no errors, the process P1.1 will write the details onto the Data Store, DS1.*

At end, Describe each Process, DF & DS

Data Flow Description

Data Flow	Data Item	Remarks
DF1a: Loan Type Details	1. Loan Type Name 2. Loan Type Description 3. Eligibility Rules 4. Activation Date	
DF2a: Ack – 1a	1. Error Message OR 2. New Loan Type ID as a Creation / Changes Acknowledgement	
DF1b: Loan Type Approval	1. Loan Type ID 2. Name of the Approving Authority 3. Date and Time of Approval	
DF2b: Ack- 1b	1. Acknowledgement Message that 'Loan Type ID' will be Activated with effect from 'Activation Date'	

At end, Describe each Process, DF & DS

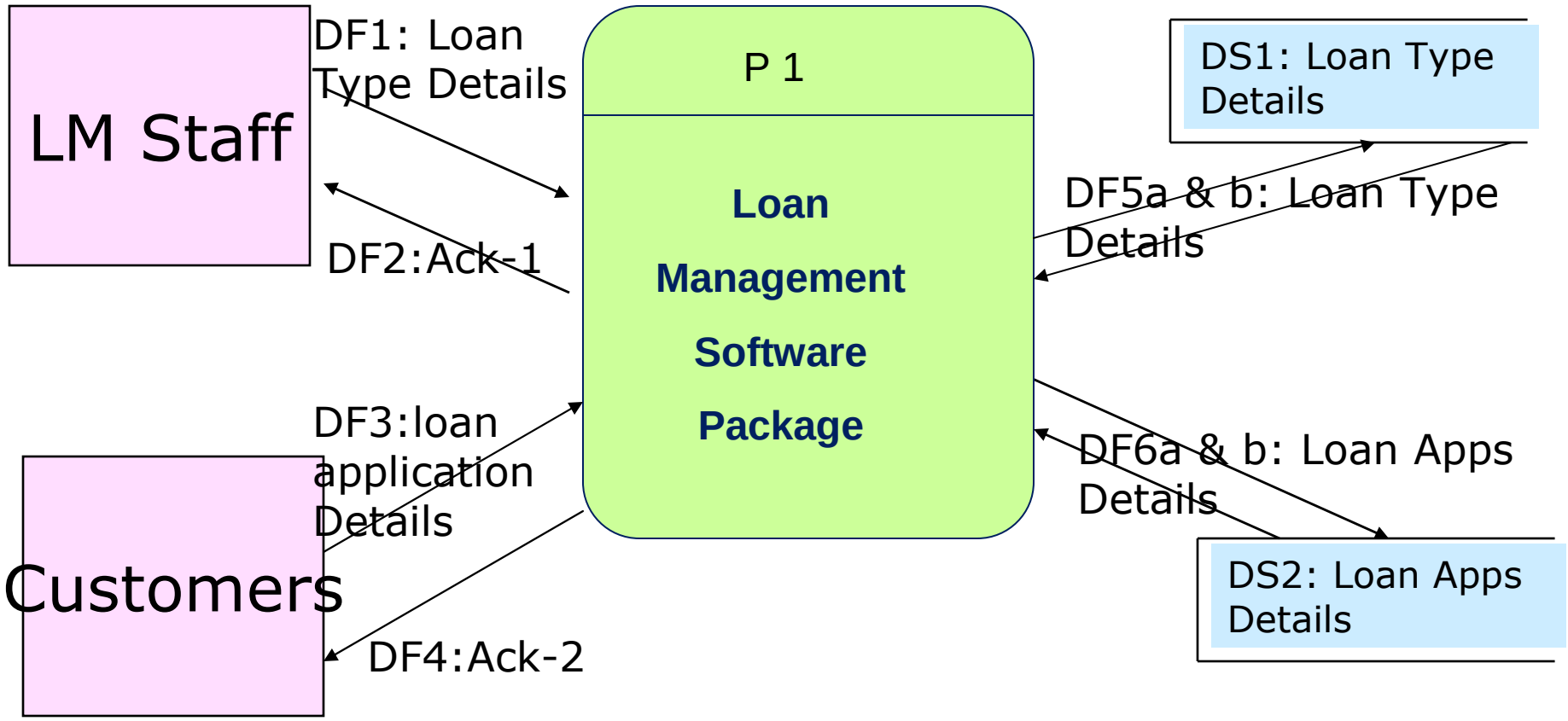
Data Store Description

Data Store	Data Item	Remarks
DS1: Loan Type Details	1. Loan Type Name 2. Loan Type Description 3. Eligibility Rules 4. Activation Date	
DS2: Loan Application Details	1. User-Name 2. User DoB 3. User Address / email-id / tel no 4. Loan Type ID 5. Amount 6. Number of years of repayment 7. Date of Loan Application 8. Date of Loan Approval 9. Date of Loan Commencement 10. Monthly EMI	

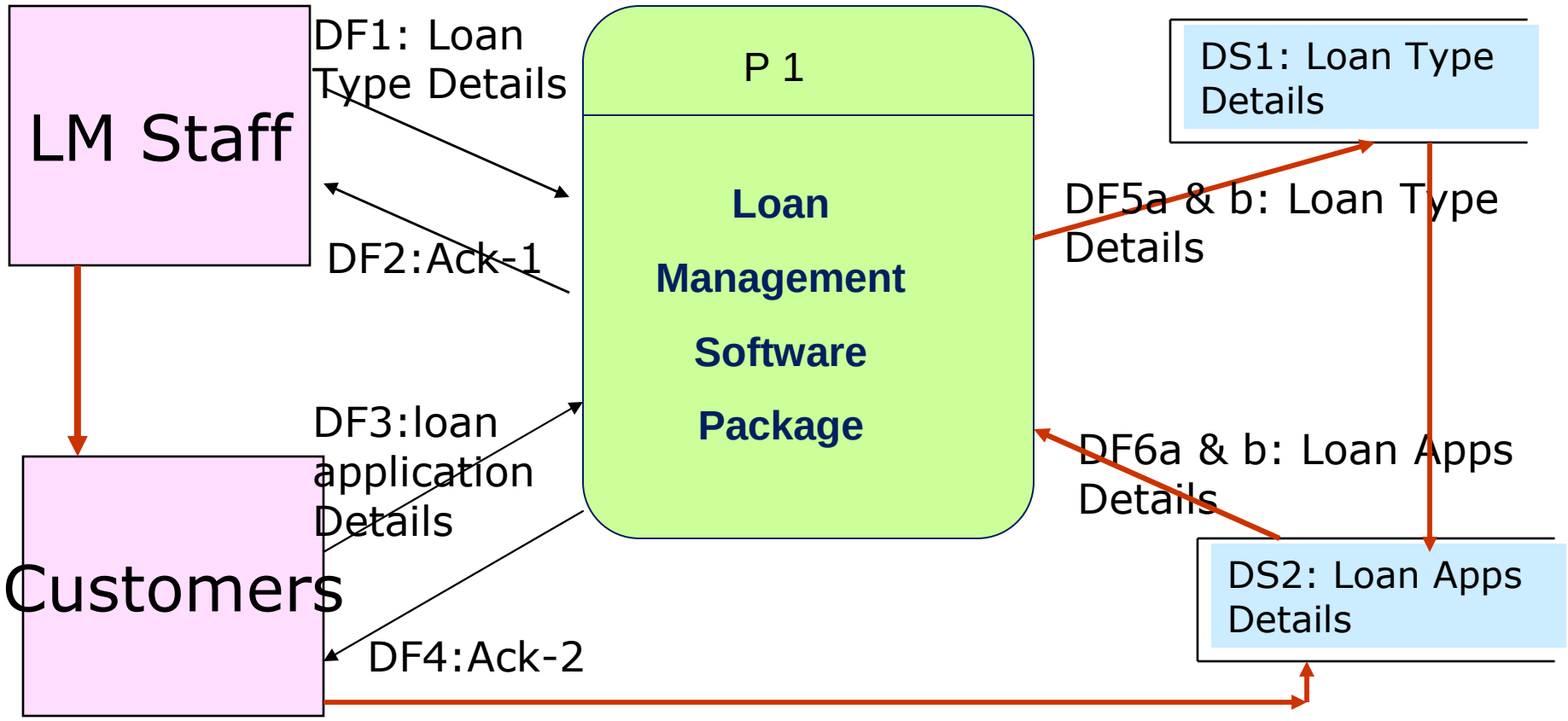
Quality Guidelines

- Completeness
  - all components included & in project dictionary
- Consistency
  - between levels: balancing, leveling
- Iterative nature
  - revisions are common
- Decomposing into primitives (lowest level)
  - when to stop?

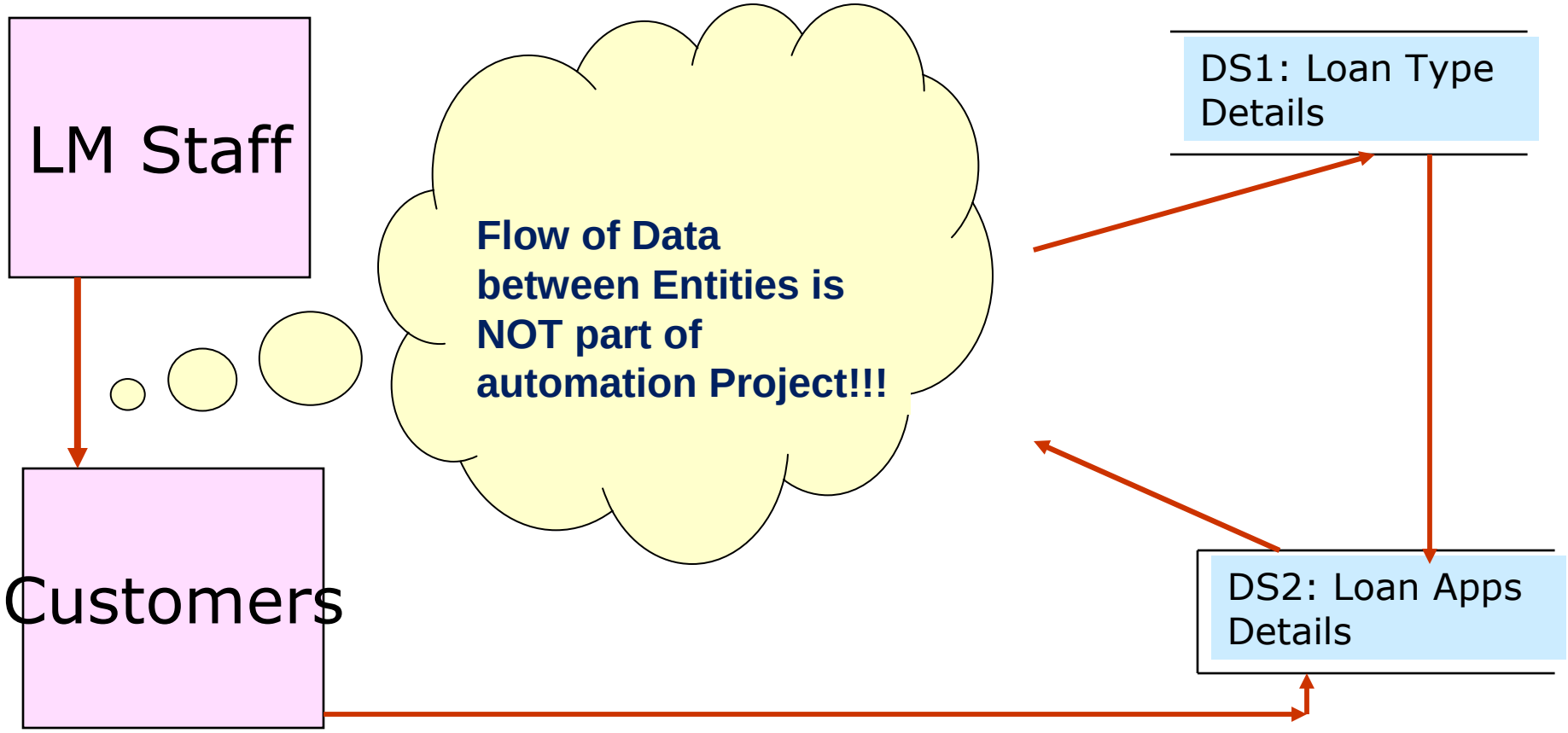
Level – 0 DFD


LM Staff
DF1: Loan Type Details
DF2:Ack-1
P 1
Loan Management Software Package
DS1: Loan Type Details
DF5a & b: Loan Type Details
DF3:loan application Details
DF4:Ack-2
Customers
DF6a & b: Loan Apps Details
DS2: Loan Apps Details

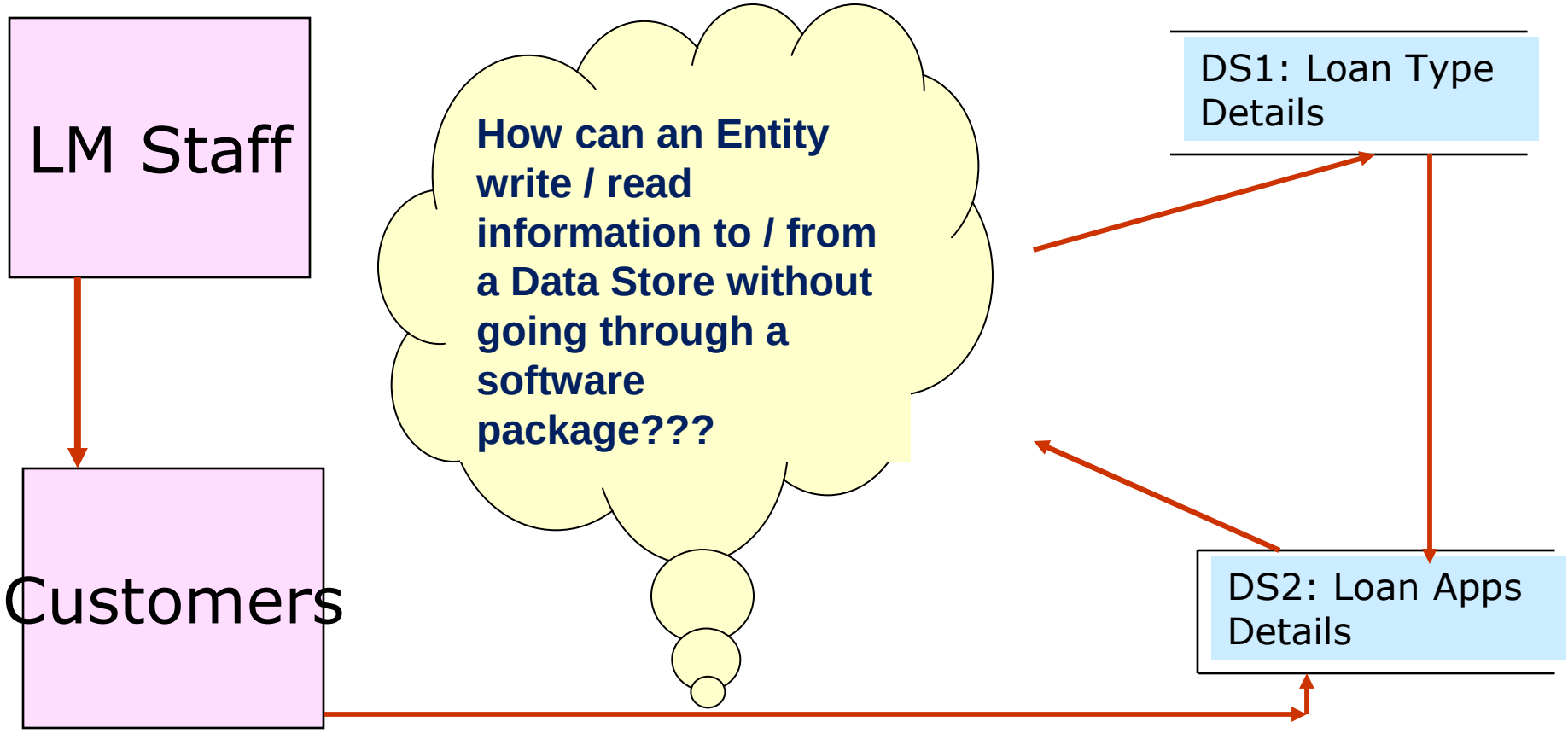
Incorrectness / Inconsistency in a DFD



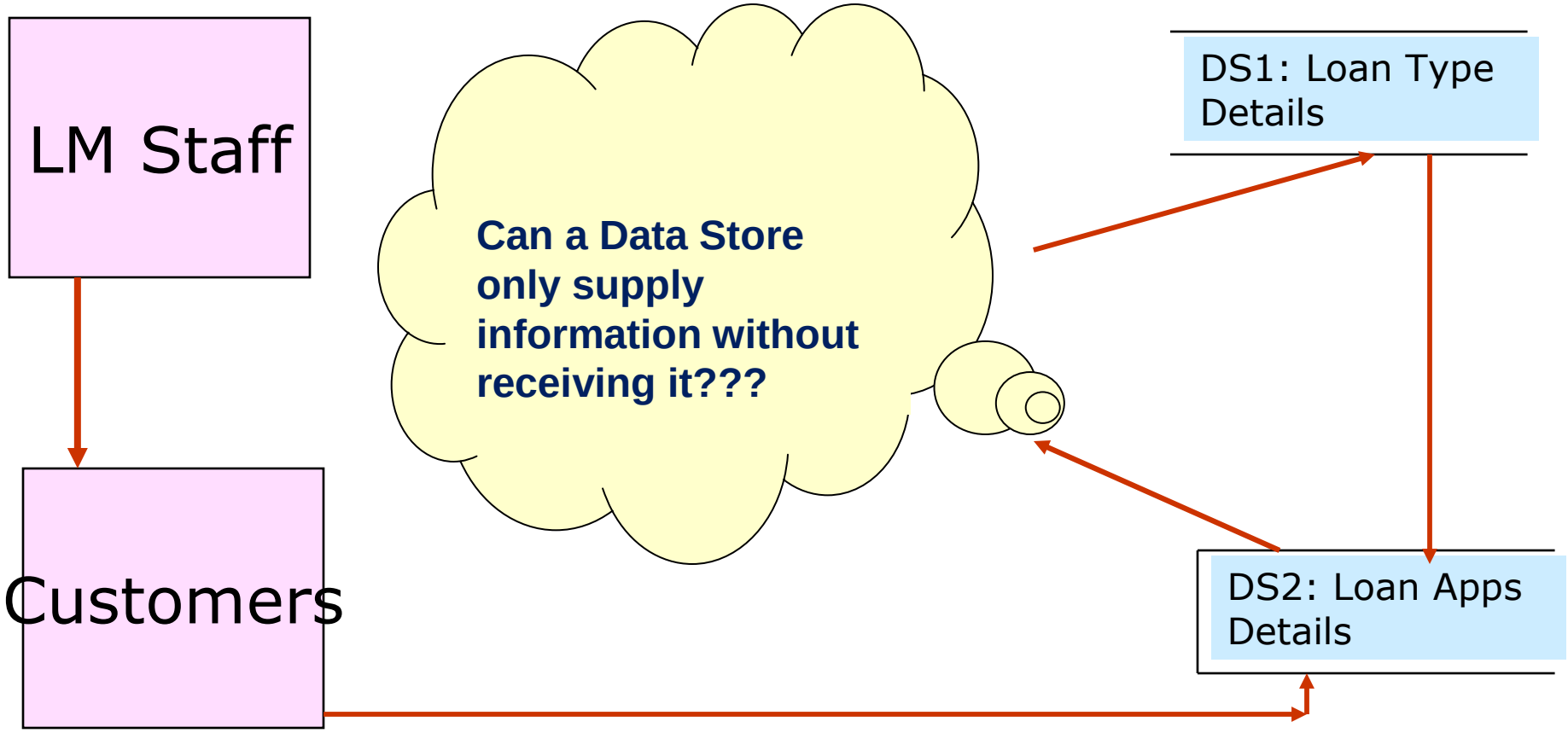
Incorrectness / Inconsistency in a DFD



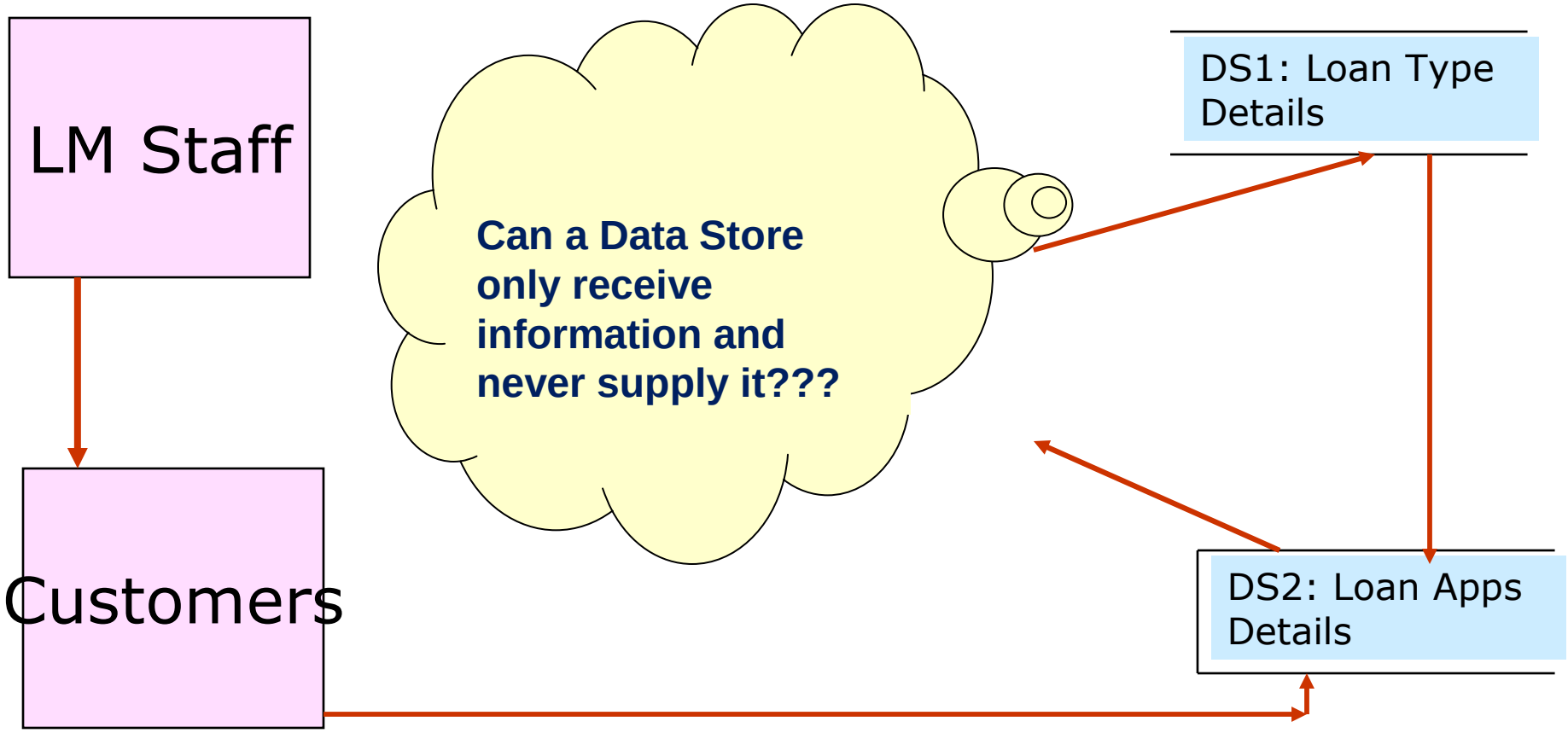
Incorrectness / Inconsistency in a DFD



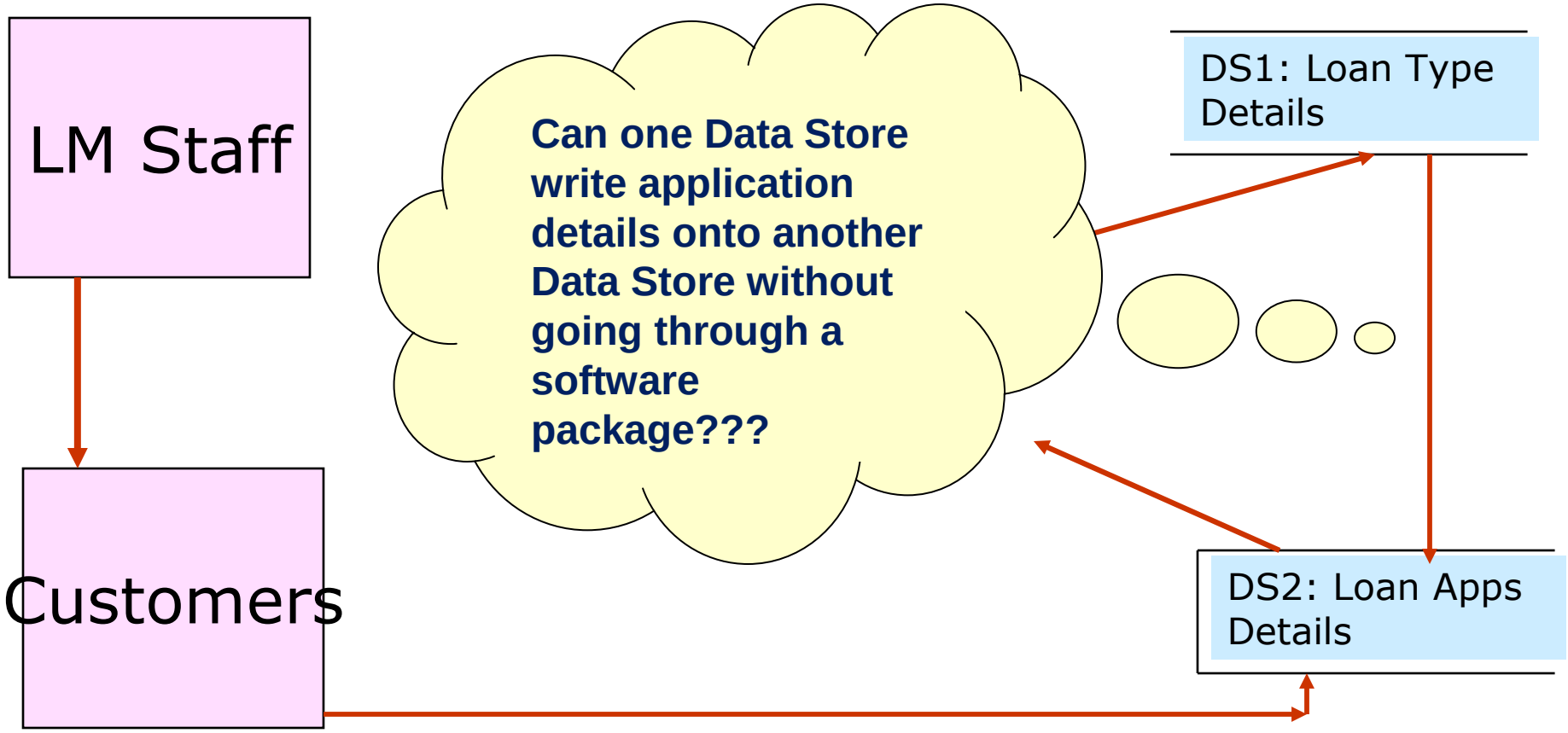
Incorrectness / Inconsistency in a DFD



Incorrectness / Inconsistency in a DFD



Incorrectness / Inconsistency in a DFD



Software Analysis Models

- ✓ Data Flow Diagrams
- ➢ Object Oriented Analysis
- ➢ Use-Case Activity Diagrams
- ➢ State Diagrams

Object Oriented Analysis

- Origin and Concepts / Principles / Techniques of Object Oriented Analysis
- Use Case Analysis
- Use Case Activity Diagrams
- State Diagrams

Object Oriented Analysis

Genesis:

- The O-O concepts mainly evolved during late 80's and early 90's
- By that time, there were more than 100 software systems of more than 1 m LOC;
- The predominant software engineering analysis model was DFDs of SSAD (Structured Systems Analysis and Design)
- DFDs used the concept of Data Flow through the Software Package and the principles of 'Process Decomposition' and 'Data Store Decomposition' to enhance the completeness, consistency and un-ambiguity of SRS

Object Oriented Analysis

- The field of Software Engineering achieved modularity through,
  - Programs / sub-programs / functions
  - Local variables and parameter passing
  - Non-redundant relational data tables
- Overall this lead to a Software Package being made of 'loosely coupled, highly functionally coherent' software sub-programs

Still, there were some major concerns:

- Data usage (creation/ modification / deletion) was spread across many sub-programs
- There were no concepts / principles / techniques to reason with data independence

Object Oriented Analysis

Concepts, principles and techniques of **Object Oriented Approach**

- Software automates information processing
- A wonderful abstraction in the field of 'information processing' is
  - **Object,**
  - **its own data,**
  - **its responsibilities to its users, and**
  - **its interactions with other objects**

- Examples:
  - Loan
  - Book
  - Order, etc

Object Oriented Analysis

Object Oriented Approach to Software Engineering looks at All,

- Models of SE
- Phases of SE

from the point of view of **'information objects'** being taken through their **own natural life cycle**:

- Loan details being created, viewed, modified, de-activated, re-activated, monthly-status-being-presented, closed, etc.
- Book details being crated, books being reserved, issued, returned, deleted, etc

Object Oriented Analysis

Object Oriented Approach

- Three scientists,
  - **Grady Booch, James Rumbaugh and Ivar Jacobson**

  are known to have taken the O-O Technology to the best of depth and breadth, independently with common concepts and principles but with varied methodologies, notations and techniques

- There was public sentiment that they come together and Unify the Approach

Object Oriented Analysis

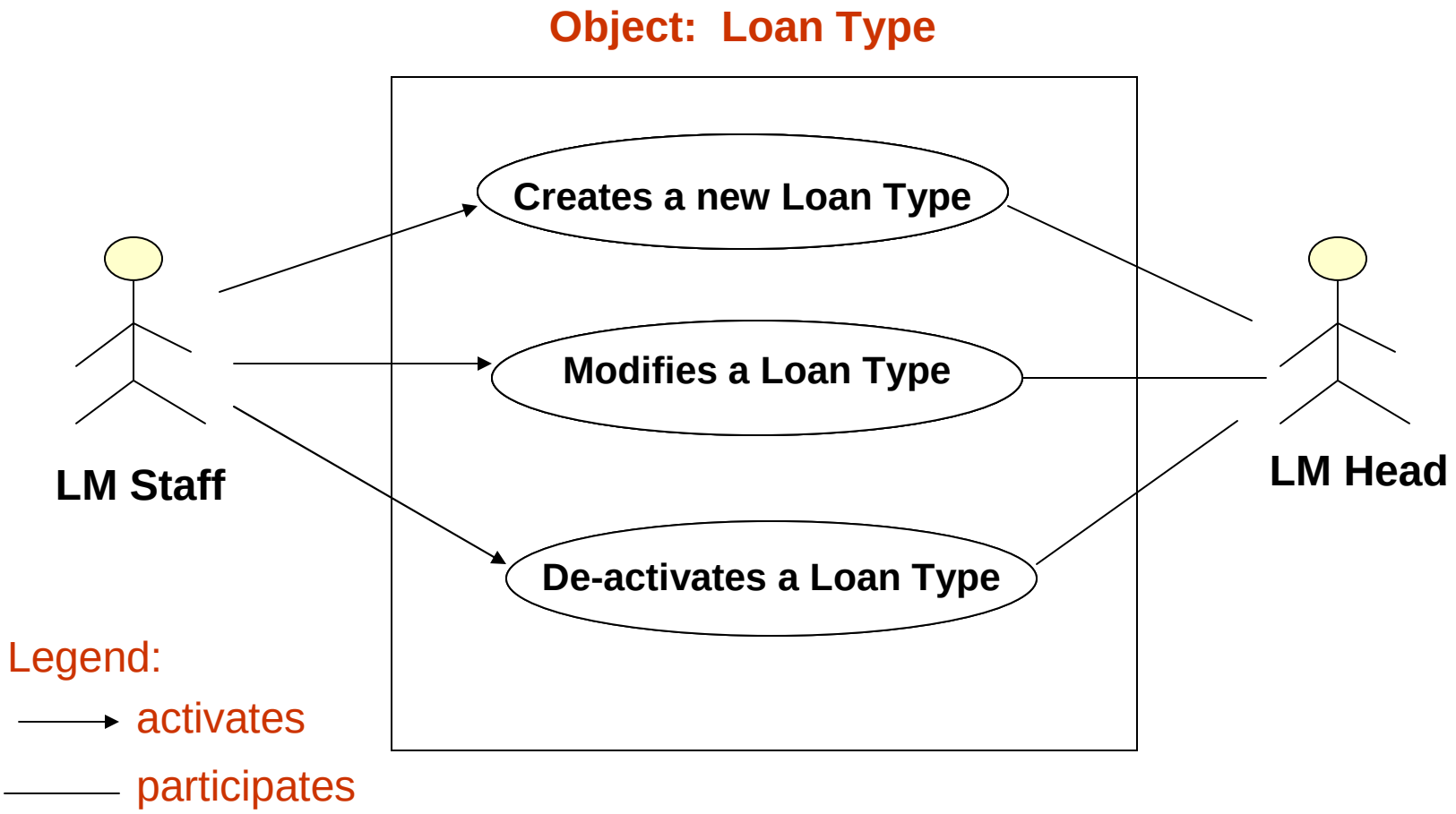
Object Oriented Approach

➔ Rational Software company brought them together which lead to **Unified Approach (UA)** and **Unified Modeling Language (UML)**

➔ UA and UML adapted Ivar Jacobson's **Use Case Approach**, **Use Case Analysis** to Object Oriented Analysis

Object Oriented Analysis

Use Case Approach to analyze SRS: Consider the SRS of 'Loan Management'



Object Oriented Analysis

Object Oriented Approach: Properties of an Object

- Business Operations revolve around Objects
- Objects have attributes (data of their own)
- Objects have life of their own (are created, persisted, modified, deleted) and allow business operations being performed on them
- Objects are saved onto / retrieved from the permanent storage during the business operations
- A business operation can involve more than one distinct Object and these Objects generally interact (give and take information) among themselves during the business operations; Objects have a behavior

Object Oriented Analysis

From a given SRS:

- Identify Objects and attributes of each object
- Find Use Cases: Business Operations Performed by Users on each Object
- Categorize the Users
- Elaborate Use Cases: Unravel the Scenarios of each Use Case

Object Oriented Analysis

Object Oriented Approach: An Important Question:

- How to identify 'Objects' in a business operation?
- There are Important guidelines, but NO scientific technique
  - Noun-Phrases in an SRS that describe information (data) being created, persisted, modified, viewed / presented / reported, deleted and have a life of its own
- Examples:
  - Objects: Loan, Book, PNR, Flight, Customer, etc
  - Non-objects: Log-in, Date-of-Journey, Ticket-Amount

Object Oriented Analysis

Object Oriented Approach: An Important Question:

- How to identify Actions Performed on an Object and the Actors who perform those Actions?
- There are Important guidelines, but NO scientific technique
  - Verb-Phrases in an SRS that describe operations performed on the Objects
- Examples:
  - Loan: Loan Application is created by Citizen;
  - Loan: Loan-Application is validated by LM Staff;
  - Loan: Loan-Application is Approved or Rejected by LM Head

Object Oriented Analysis

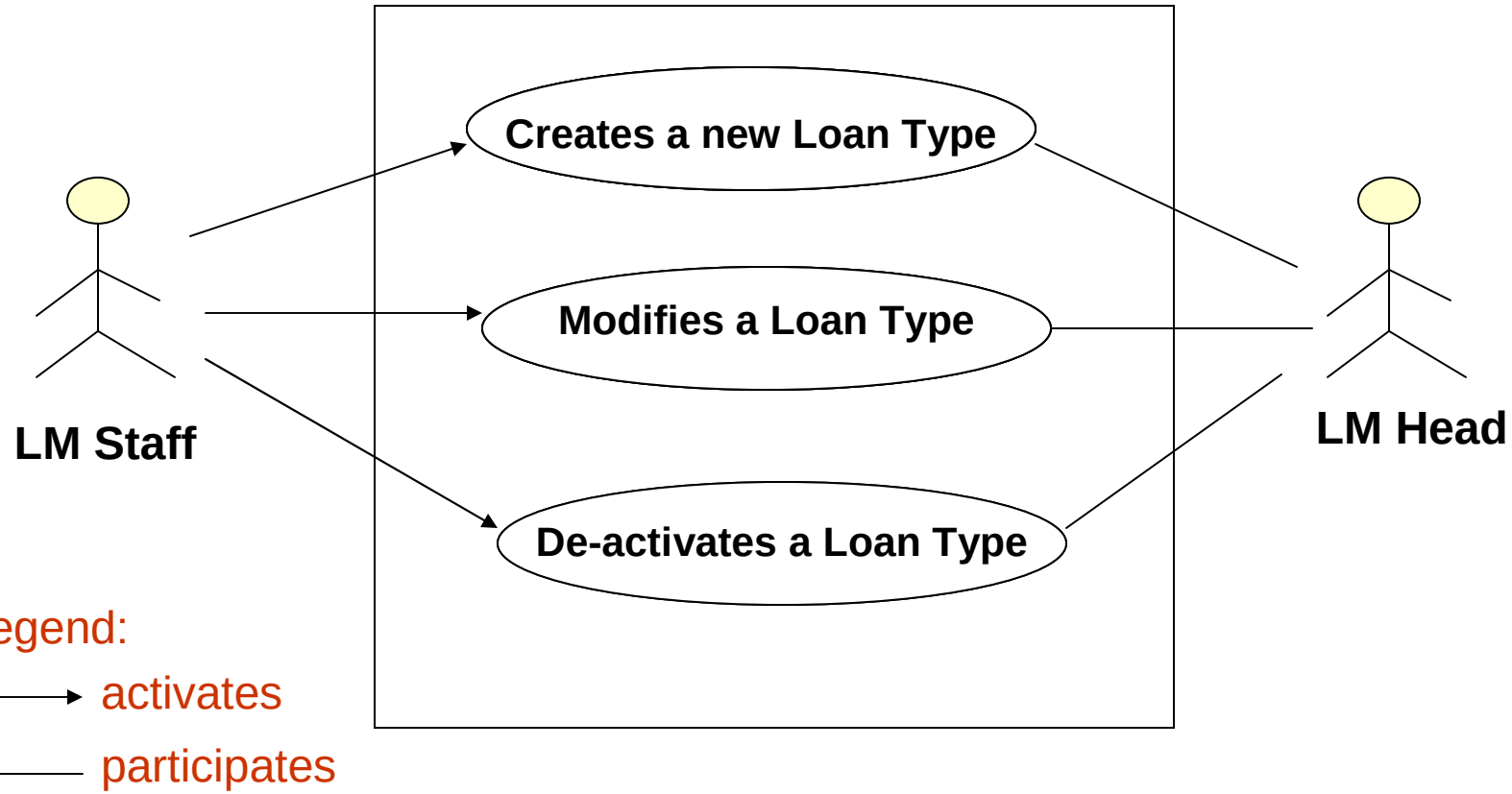
Object Oriented Approach: Use Case Analysis

- Some categories of Users **'trigger'** an action on an object; Other users **'participate'** in the operations being performed on an object
  - E.g.: Citizen 'creates' a Loan-Application Object through a triggering action; LM Staff participate and validate the Loan-Application Object;
  - E.g.: LM Staff trigger 'modification' to a Loan-Type Object; LM Head participates and 'approves / rejects' the modifications;

Object Oriented Analysis

Use Case Approach to analyze SRS: Consider the SRS of 'Loan Management'

Object: Loan Type



Object Oriented Analysis

Object Oriented Approach: Use Case Analysis

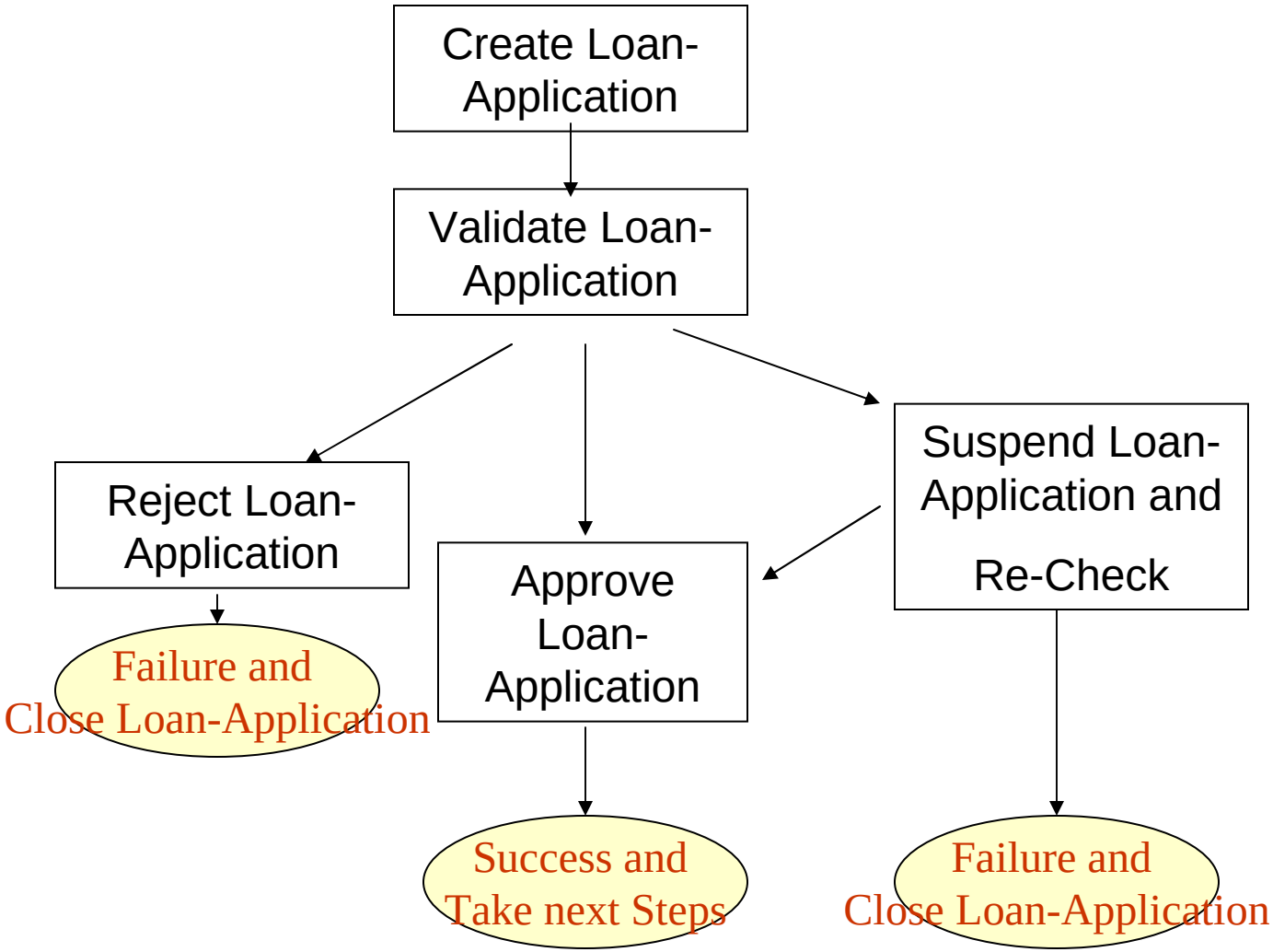
- ✓ Identify Objects, their attributes and Operations to be performed on the Objects by different Actors

- ➢ **Elaborate** each business Operation to be performed on each Object as an ordered sequence of unit-level activities
  - Use Case Activity Diagrams describe the minutest-level (unit-level) activities to be performed on an object and its response (behavior towards the actions) as part of a business operation on that object

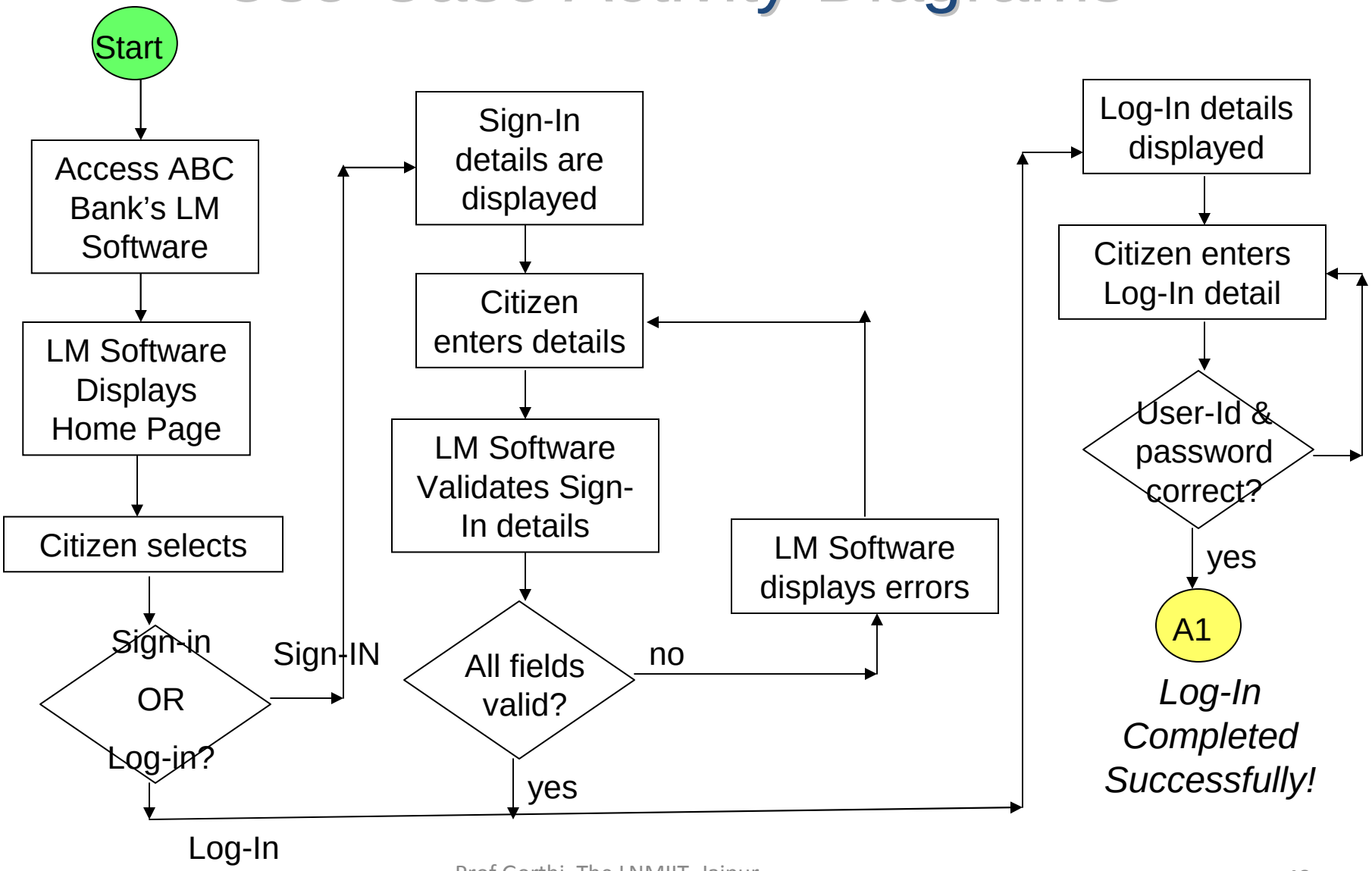
ROLE OF USE CASE Activity Diagrams

- All possible responses of an Object to a triggering action are modeled as a directed cyclic graph
- Use Case Activity Diagrams are helpful when the situation is complicated with multiple, alternate possibilities of response from the Object

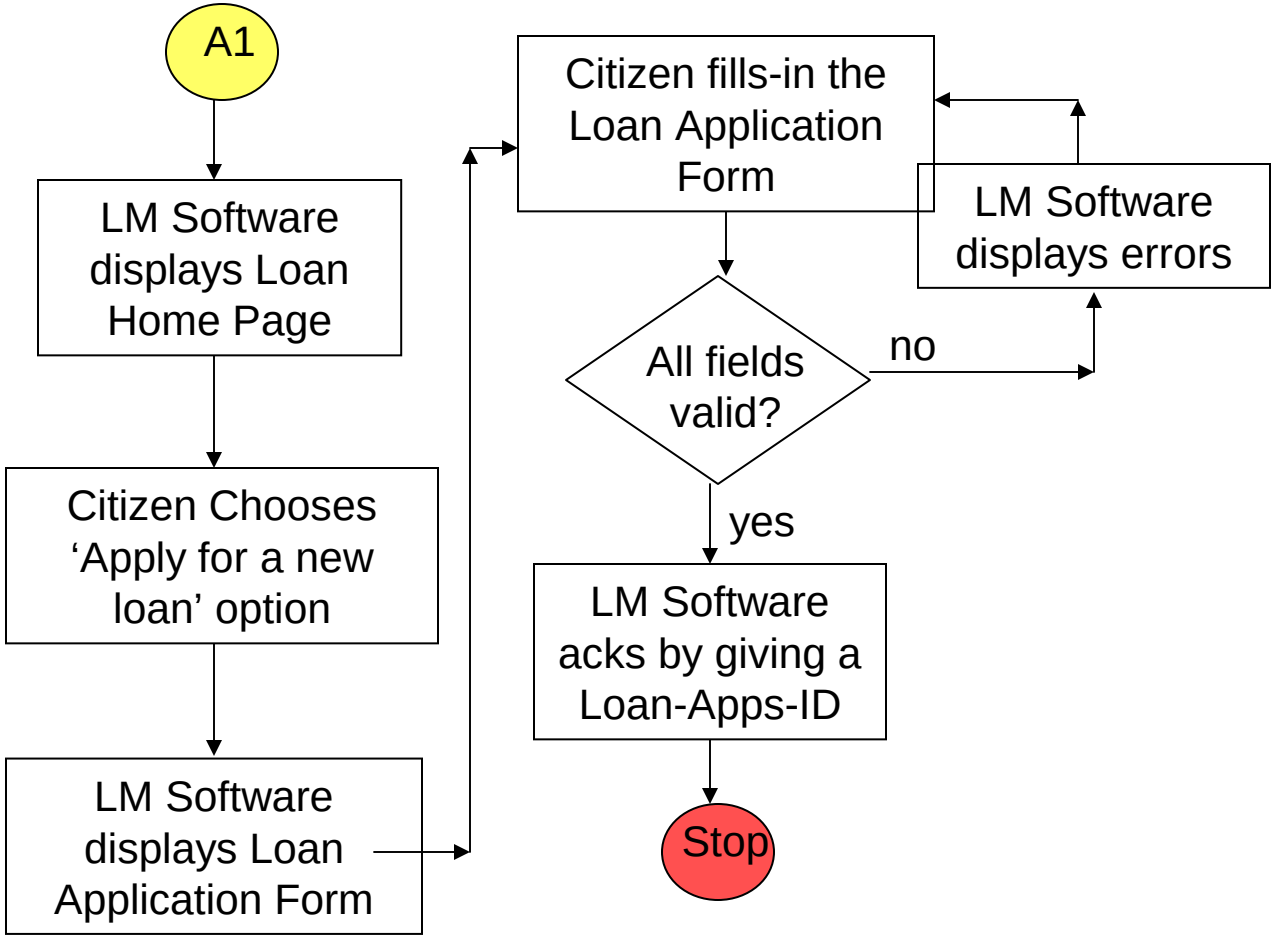
Elaborating Use Cases as Use-Case Activity Diagrams



Elaborating Use Cases as Use-Case Activity Diagrams



Elaborating Use Cases as Use-Case Activity Diagrams

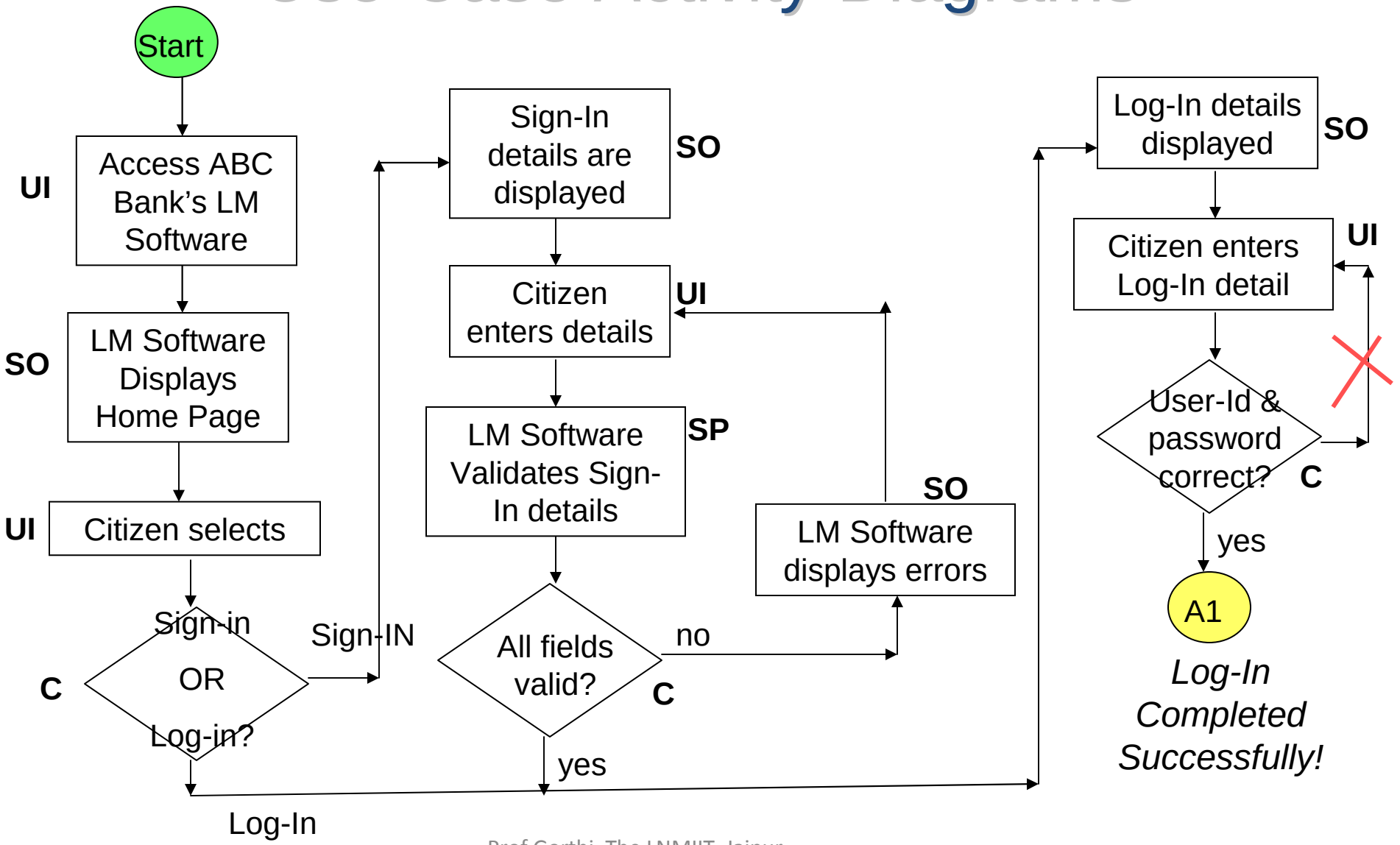


Improvements to Use Case Activity Diagrams

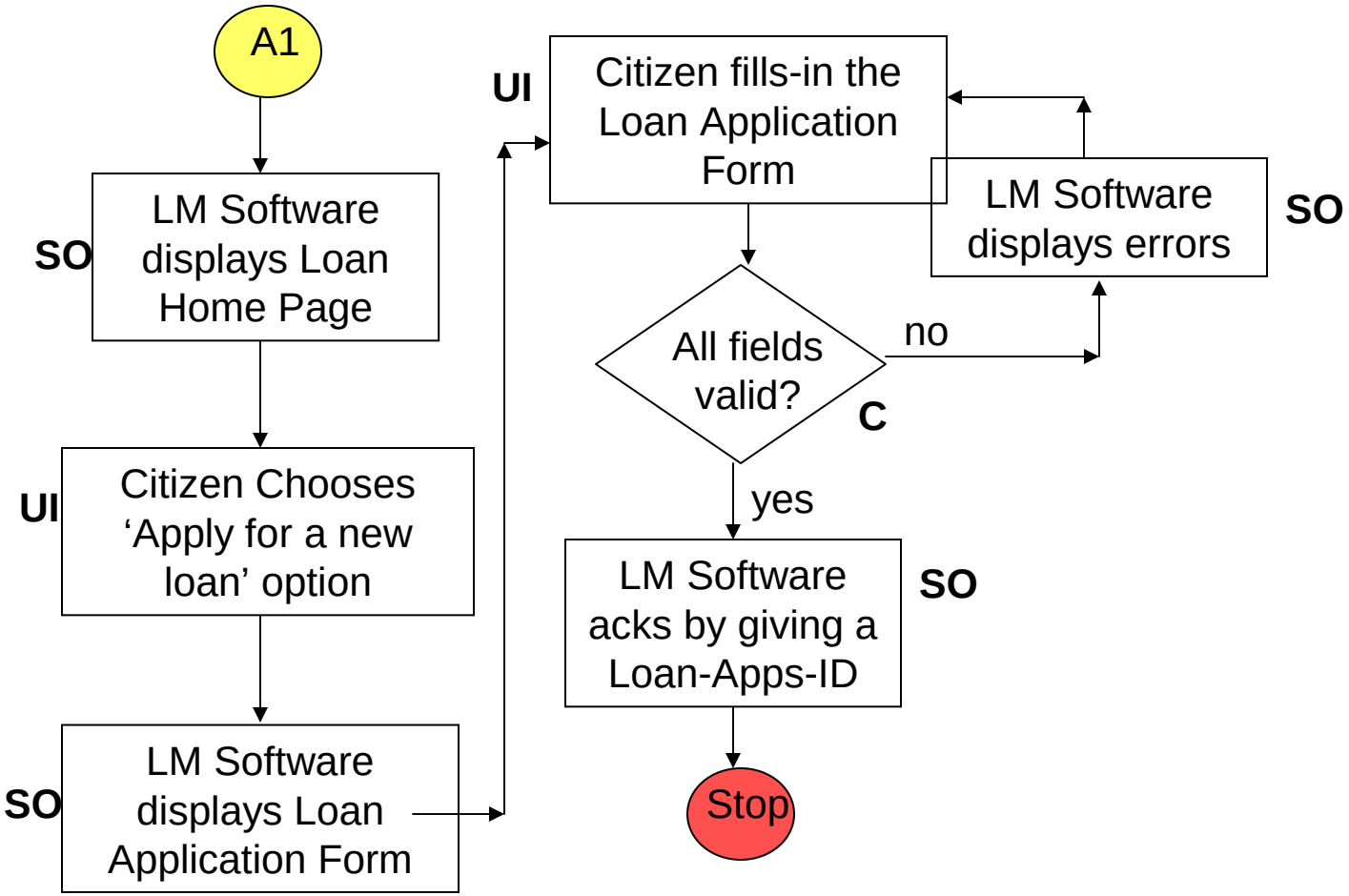
Ravi Gorthi et al, Infosys [2007] created a major enhancement to the UML Use Case Activity Diagrams:

- They defined & brought-in a concept called '**Unit of Behavior'** of a Software Package
- UoB :: <UI, SP, C, SO>
  - UI:: User Inputs
  - SP:: Software Processes
  - C:: Checks
  - SO: Software Outputs

Elaborating Use Cases as Use-Case Activity Diagrams



Elaborating Use Cases as Use-Case Activity Diagrams



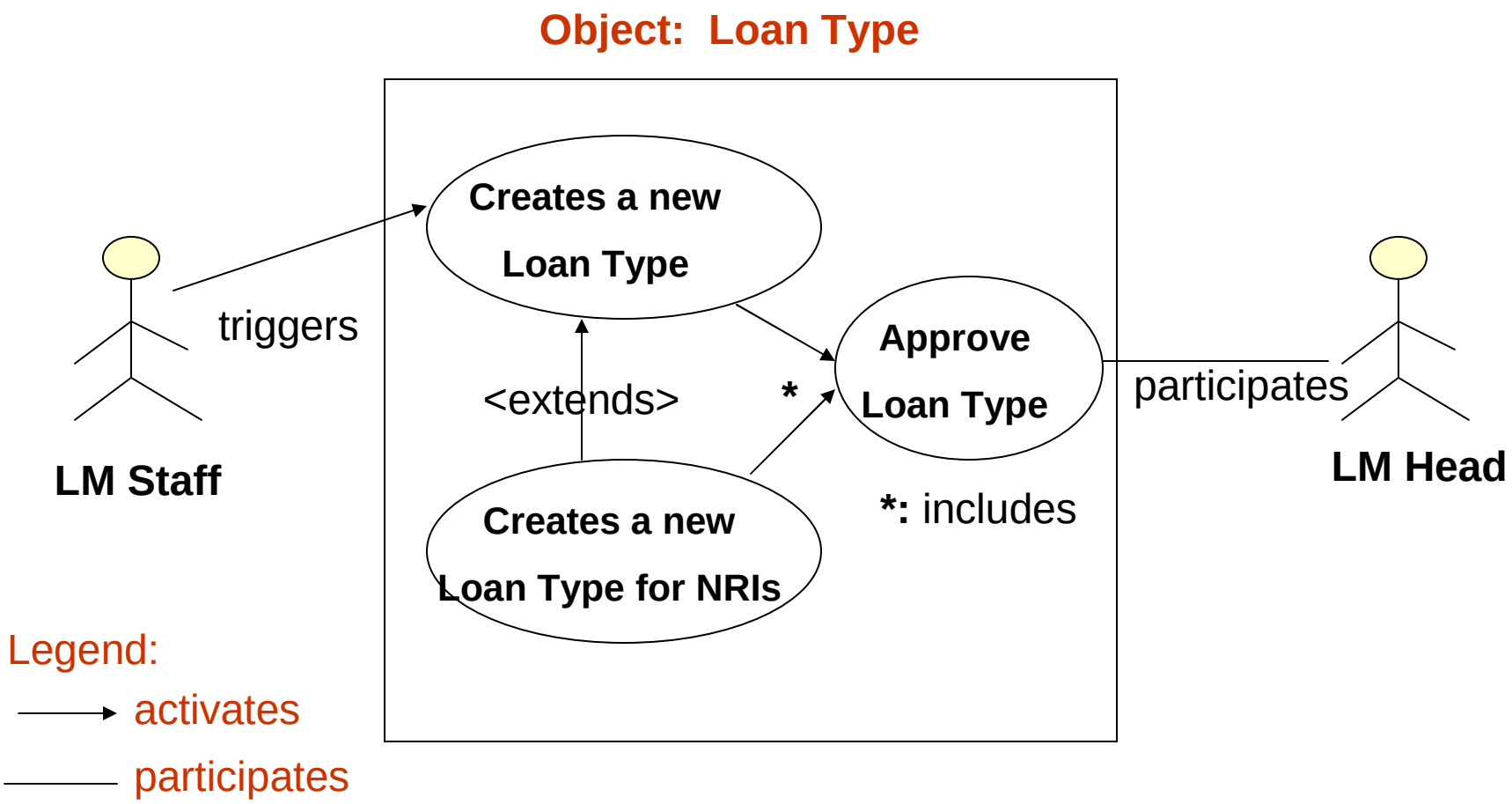
Object Oriented Analysis

Object Oriented Approach: Use Case Analysis

- ✓ Identify Objects, their attributes and Operations to be performed on the Objects by different Actors
- ✓ **Elaborate** each business Operation to be performed on each Object as an ordered sequence of unit-level activities using Use-Case Activity Diagrams
- ➢ Refinement of Use Cases using the concepts of "extensions" and "inclusions"

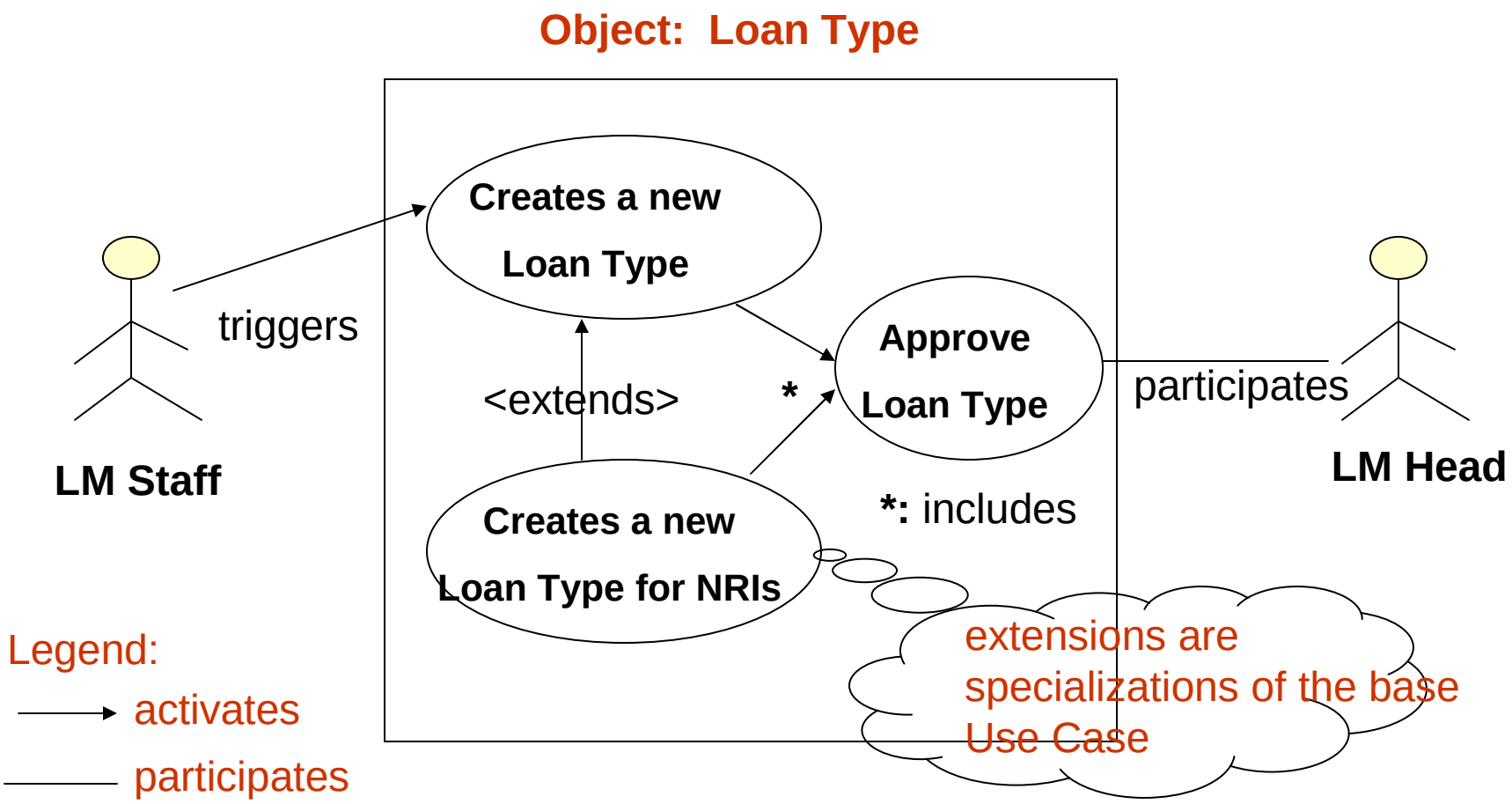
Object Oriented Analysis

Concepts of “extensions” and “inclusions” in Use-Cases



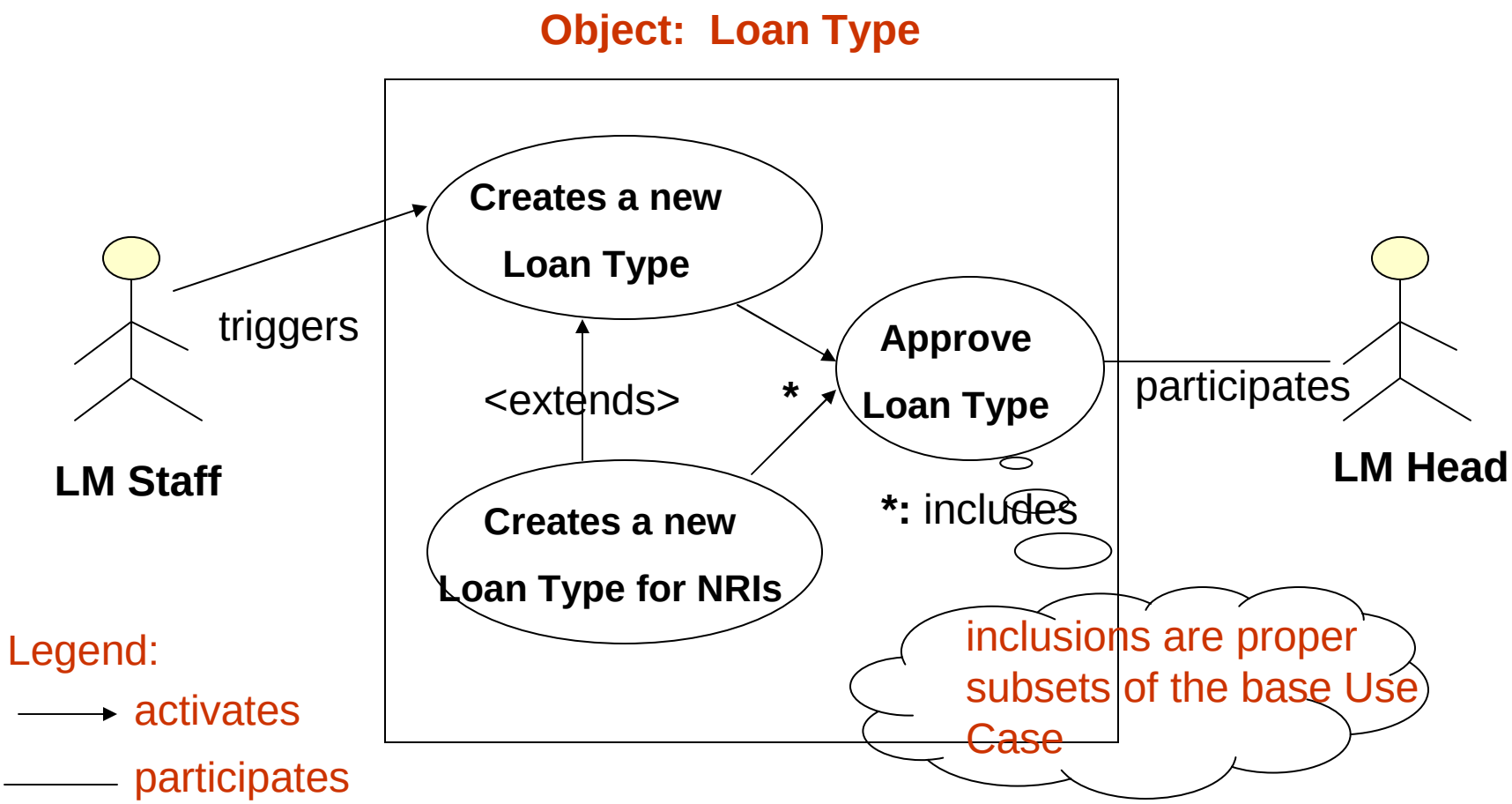
Object Oriented Analysis

Concepts of “extensions” and “inclusions” in Use-Cases



Object Oriented Analysis

Concepts of “extensions” and “inclusions” in Use-Cases



Object Oriented Analysis

Object Oriented Approach: Use Case Analysis

- ✓ Identify Objects, their attributes and Operations to be performed on the Objects by different Actors
- ✓ **Elaborate** each business Operation to be performed on each Object as an ordered sequence of unit-level activities using Use-Case Activity Diagrams
- ✓ Refinement of Use Cases using the concepts of "extensions" and "inclusions"

Avoid "Paralysis of Analysis"!!!

Software Analysis Models

- ✓ Data Flow Diagrams
- ✓ Object Oriented Analysis
- ✓ Use-Case Activity Diagrams
- ➢ State Diagrams

UML State Diagrams

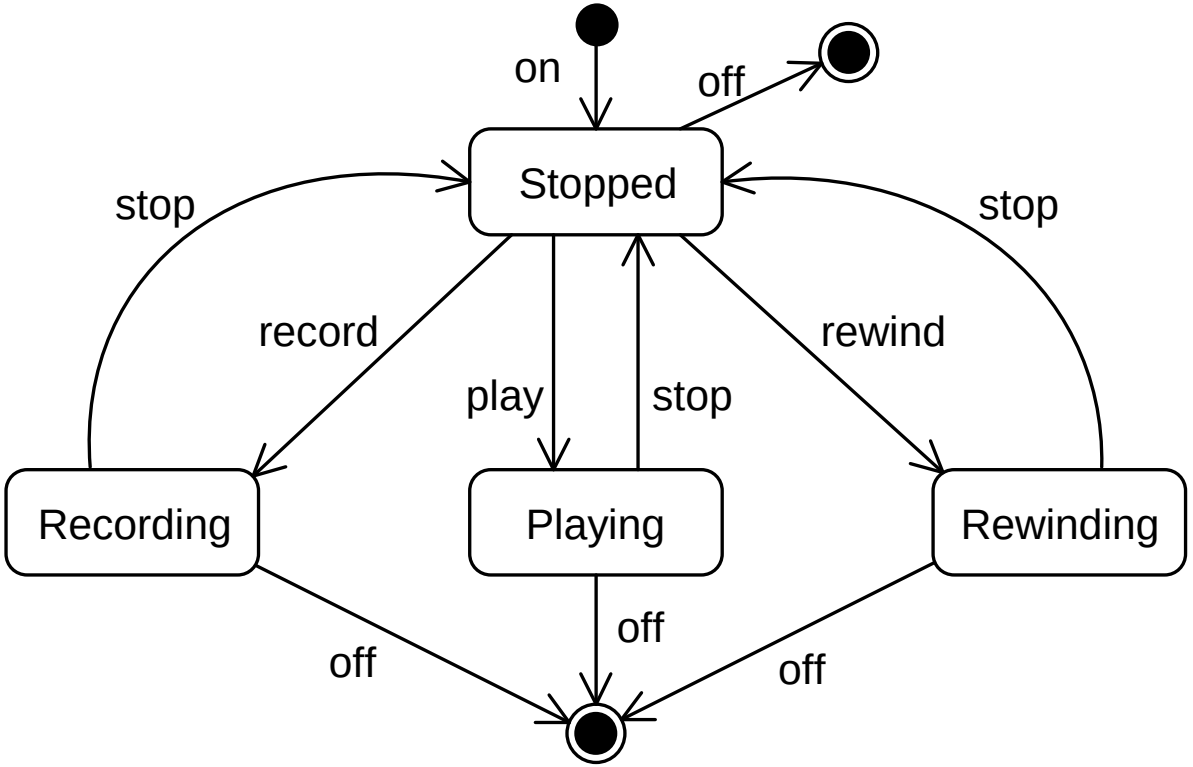
State Diagrams are typically used to analyze embedded software programs

- UML state diagram concepts & notation
- Heuristics for making good state diagrams

UML State Diagrams

- A ‘state’ is a mode or condition of being.
- An ‘event’ is a noteworthy occurrence of stimuli or input at a particular time.
- A ‘transition’ is a change from one state to another.

UML State Diagrams



UML State Diagrams

Concepts: Use of Finite Automaton

- A formal model that abstracts states and state transitions
- A finite state machine or finite automaton specification must
  - Describe all states unambiguously (names)
  - Describe all transitions by stating the source and destination states and the triggering event
  - Designate an initial state and optionally one or more ‘halt’ states

UML State Diagrams

State Diagram:

- Represent states by rounded rectangles containing the state name
- Represent transitions by solid arrows labeled with a transition string
- Transition string
  - Describes triggering circumstances
  - Actions that result
- Initial state designates the initial state
- Optional final states represents halting states

UML State Diagrams

UML State Diagrams: Consistency Checks:

- Make one initial state in every state diagram (including nested state diagrams).
- Check that no event labels two or more transitions from a state.
- Check that no arrow leaves a final state.
- Check for black holes (dead states) and white holes (unreachable states).

Software Analysis Models

- ✓ Data Flow Diagrams
- ✓ Object Orinted Analysis
- ✓ Use-Case Activity Diagrams
- ✓ State Diagrams

Summary:

Data Flow Diagrams enable the SE professionals to

- ✓ Analyze the flow of data in the TBD software in a 'top-down' way
- ✓ Identify the input and output data elements, data stores and unit-level processes that transform input data into output

Software Analysis Models

- ✓ Data Flow Diagrams
- ✓ Object Orinted Analysis
- ✓ Use-Case Activity Diagrams
- ✓ State Diagrams

Summary:

O-O Analysis enables the SE professionals to

- ✓ View the TBD software package as performing relevant business operations on business objects
- ✓ UML Use Case approach is a way to divide a complex set of business operations into distinct operations by different categories of users on various business objects

Software Analysis Models

- ✓ Data Flow Diagrams
- ✓ Object Orinted Analysis
- ✓ Use-Case Activity Diagrams
- ✓ State Diagrams

Summary:

O-O Analysis enables the SE professionals to

- ✓ While UML Use Case Activity Diagrams are typically quite useful to analyze Management Information Processing scenarios, UML State Diagrams facilitate analysis of embedded systems