Software Engineering

Prof Ravi Prakash Gorthi

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

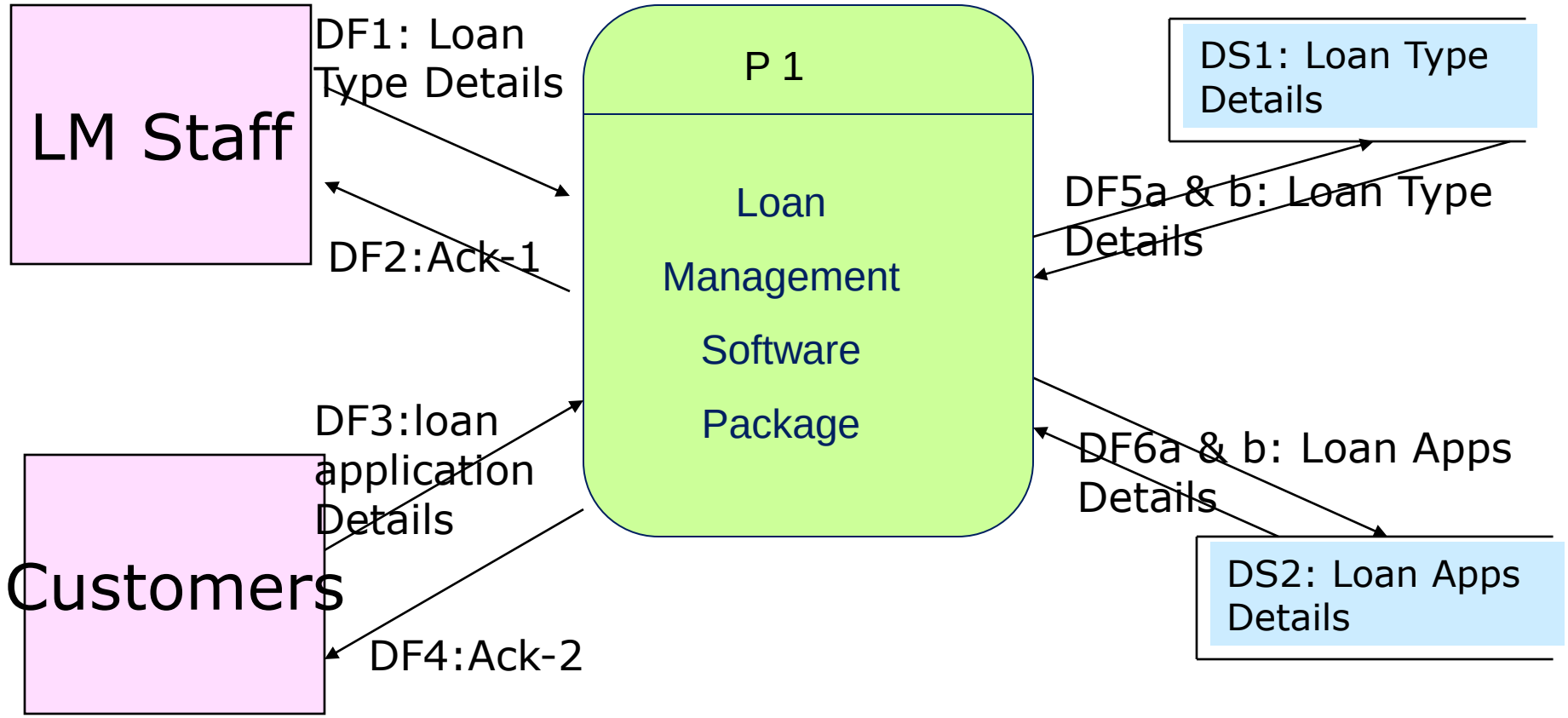
What do Data Flow Diagrams Offer?

- 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



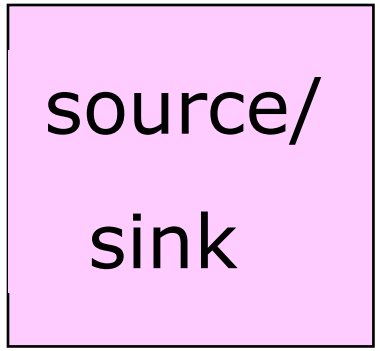
Systems Analysis Using DFDs

- 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

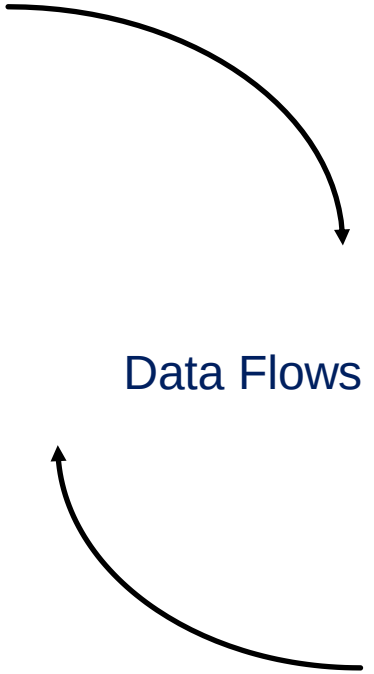
Sources/Sinks (external entities)

- 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!



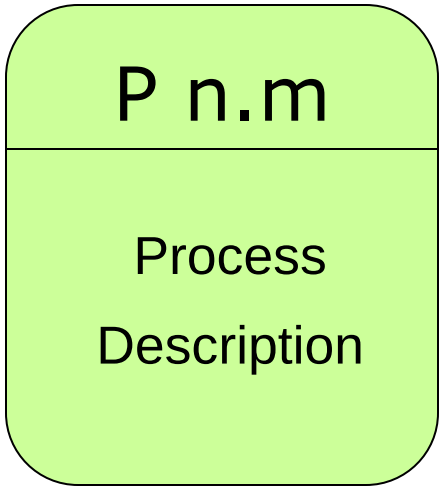
Processes

- 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

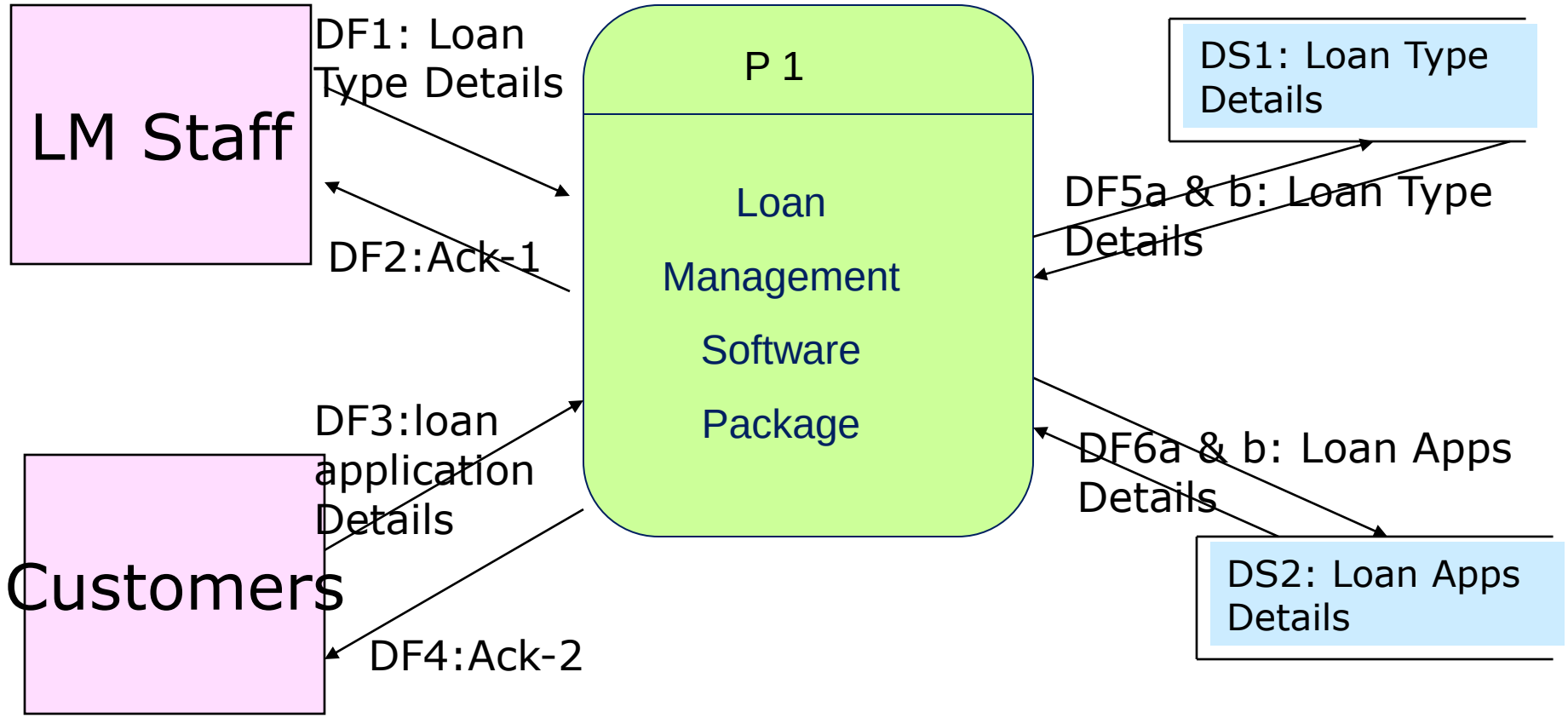
Different Types of DFDs

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

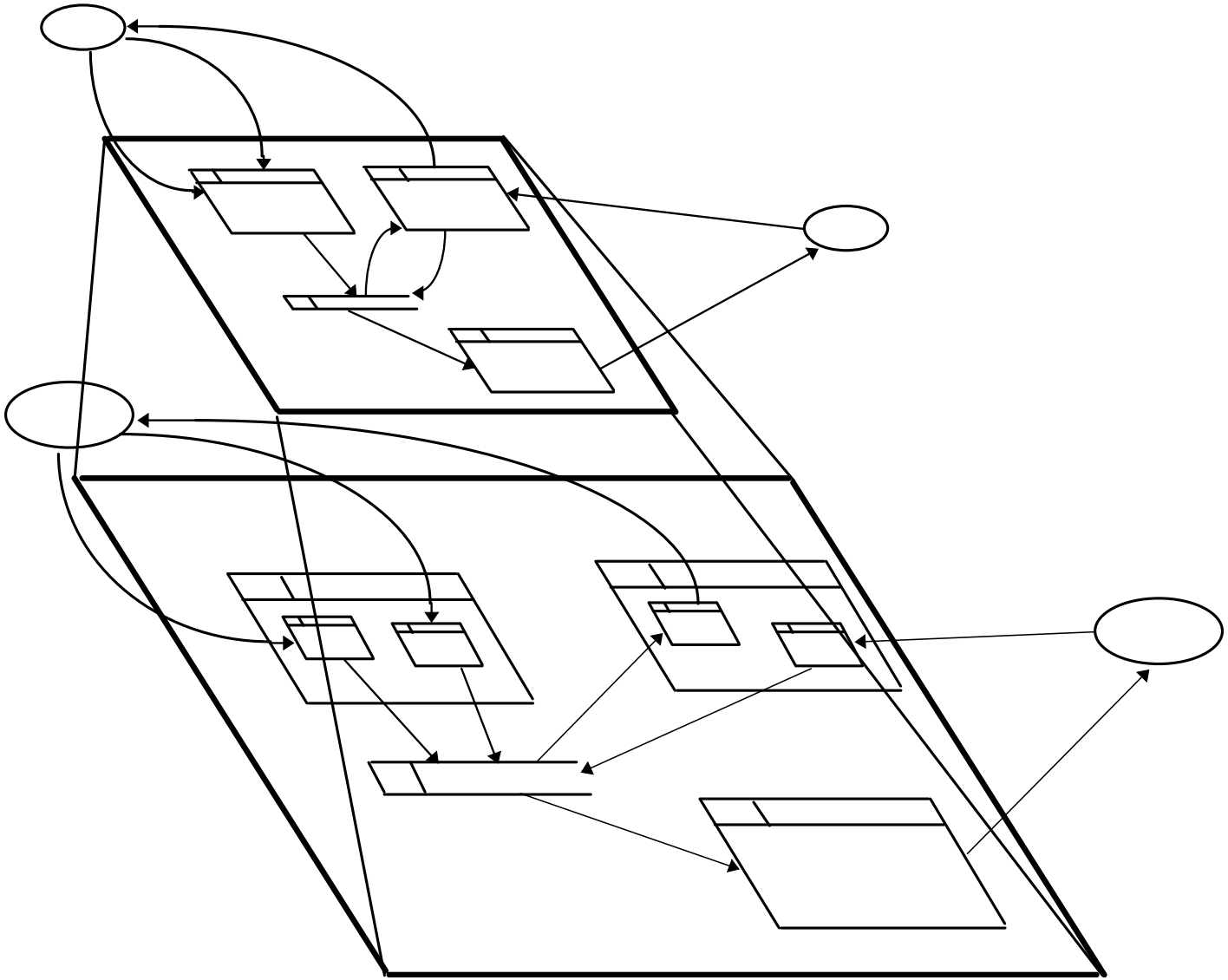
Drawing a Level-0 Diagram

- 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

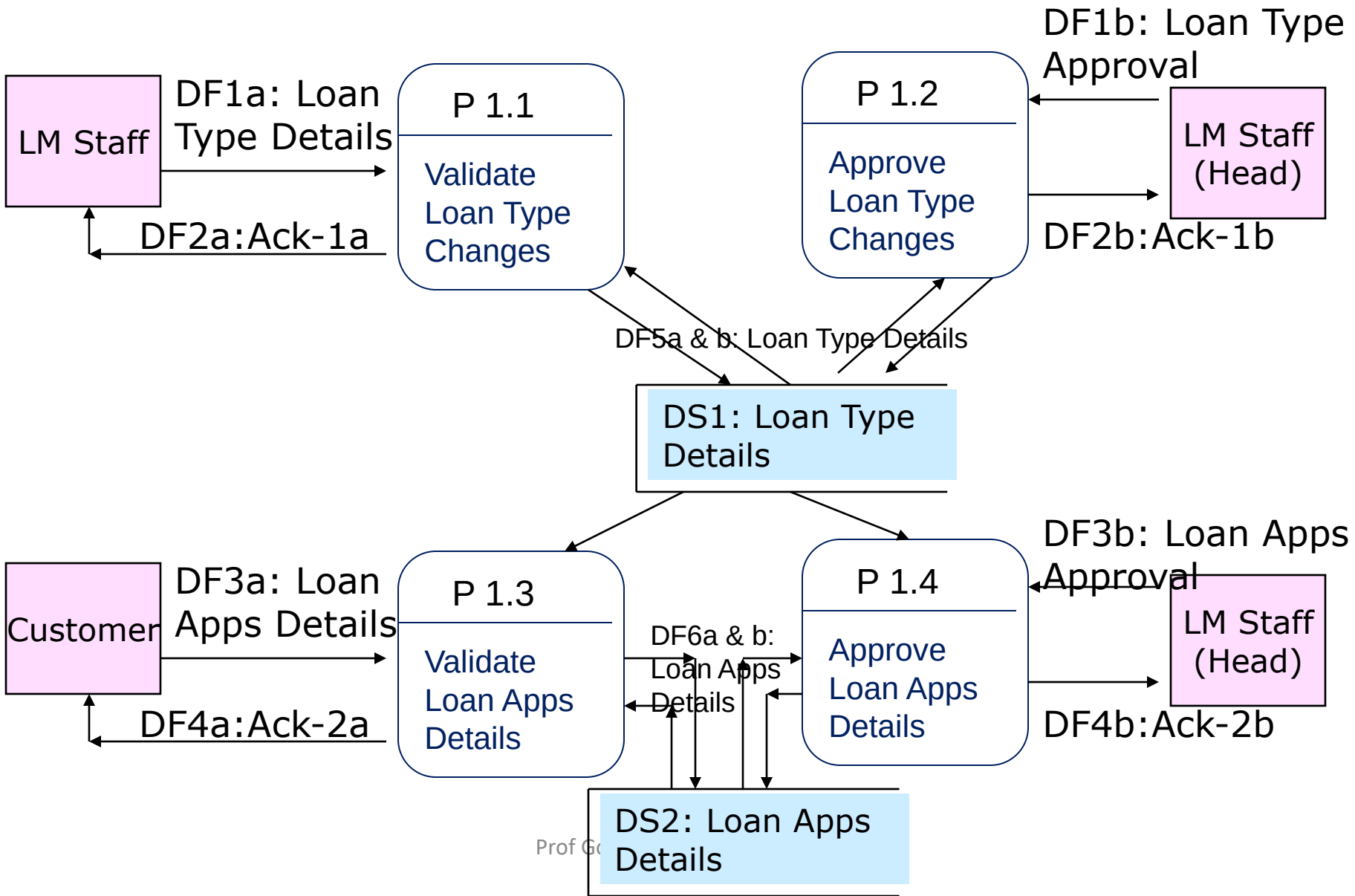
Level – 0 DFD


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
DF6a & b: Loan Apps Details
Customers
DS2: Loan Apps Details

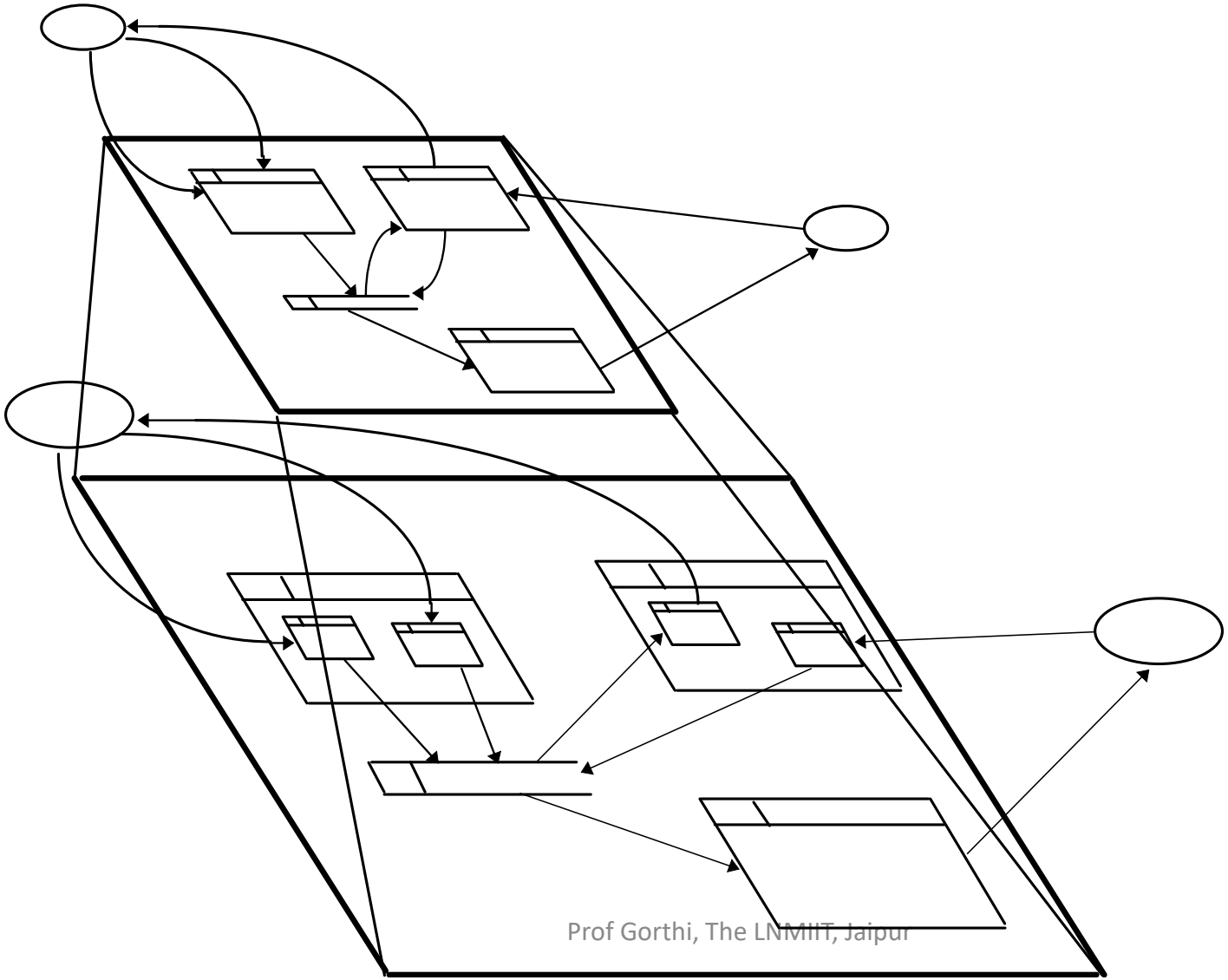
Decomposing the Level-0 DFD



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 Staff (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 Staff (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

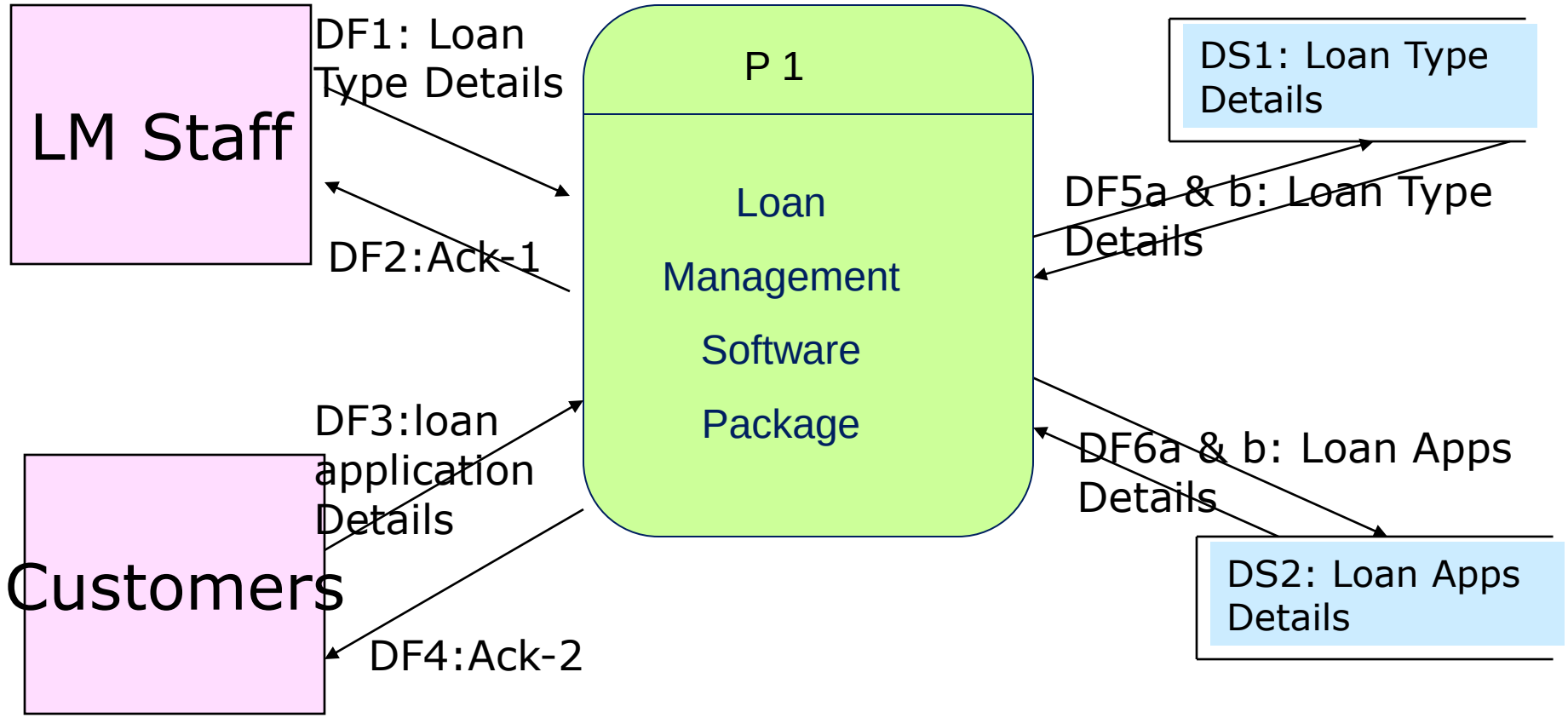
Data Store Description

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	

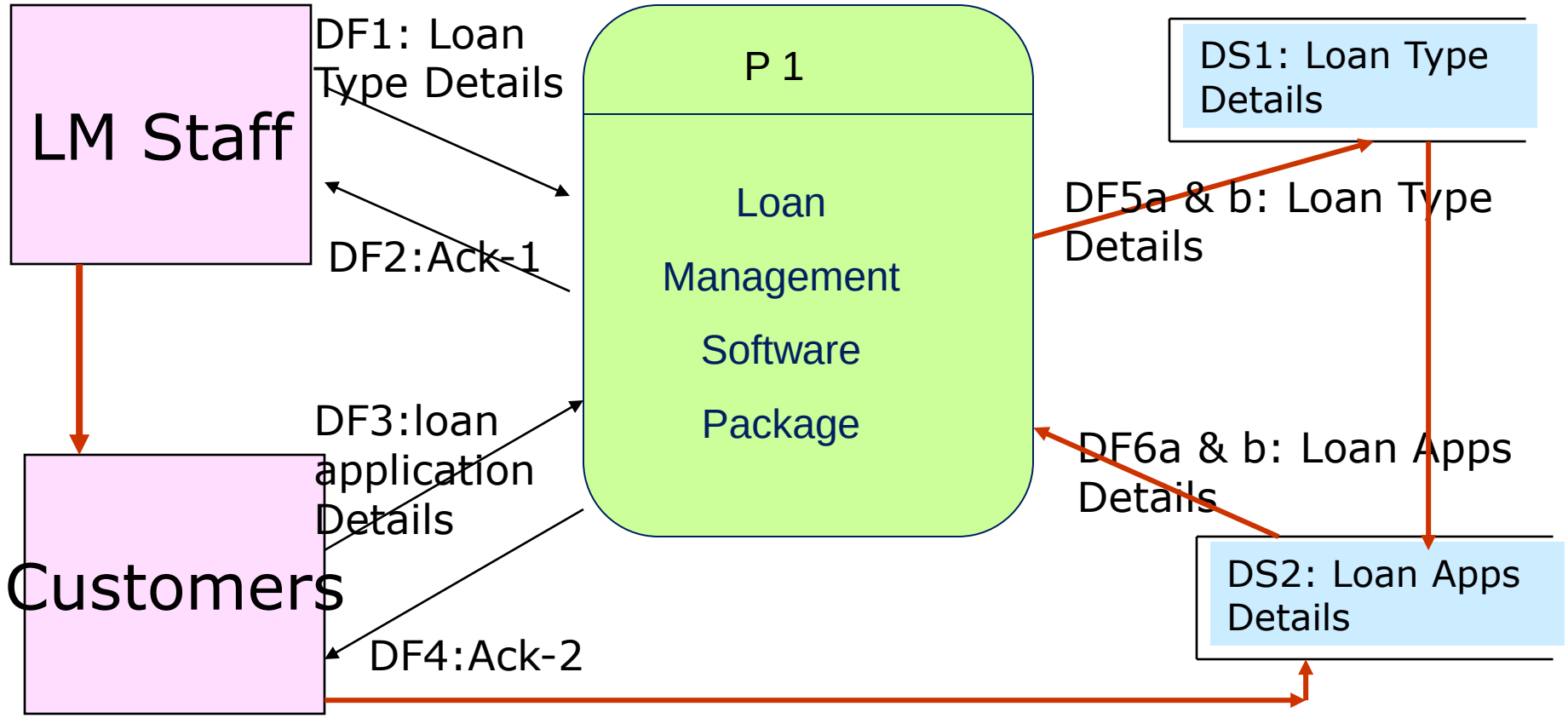
Quality Guidelines

- 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?

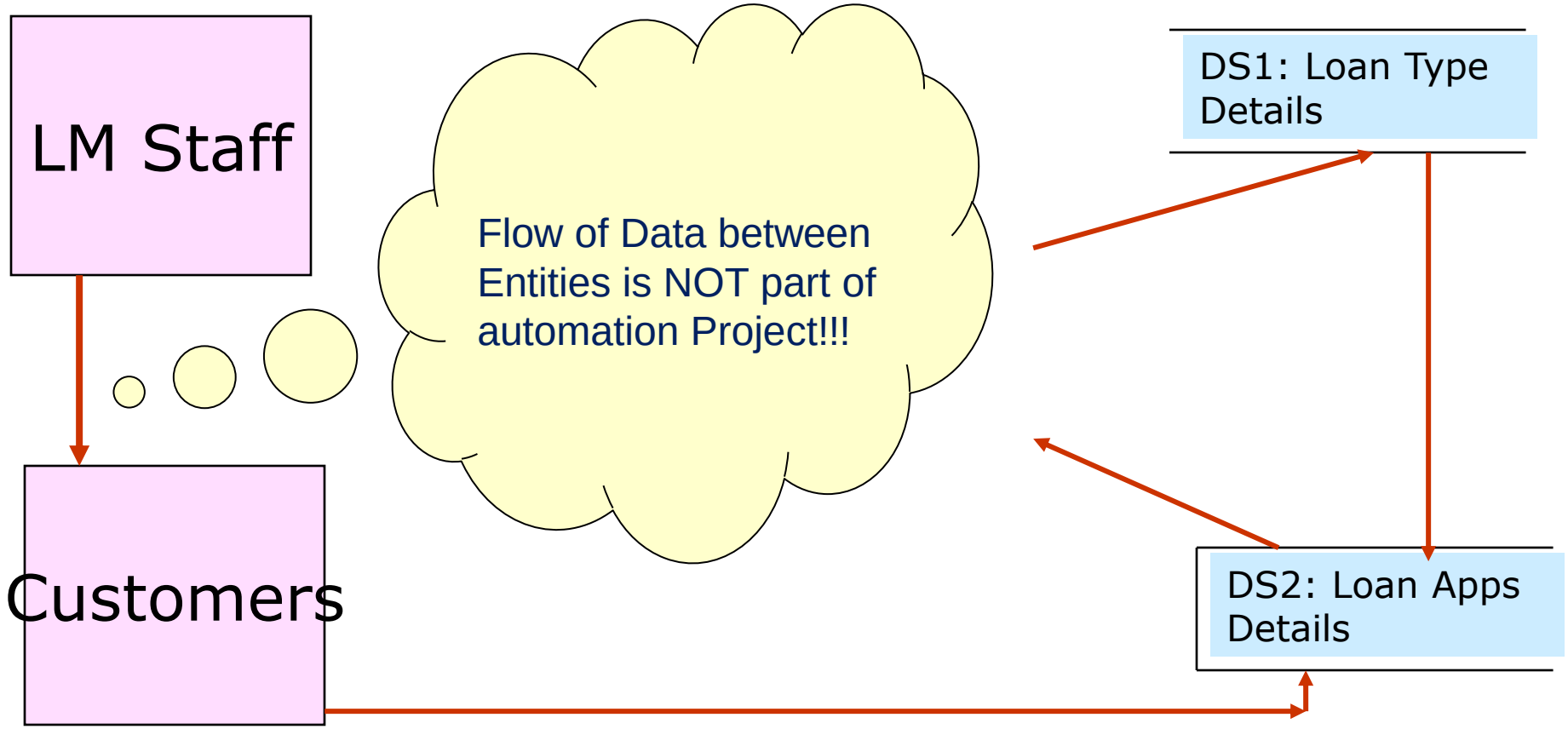
Level – 0 DFD


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

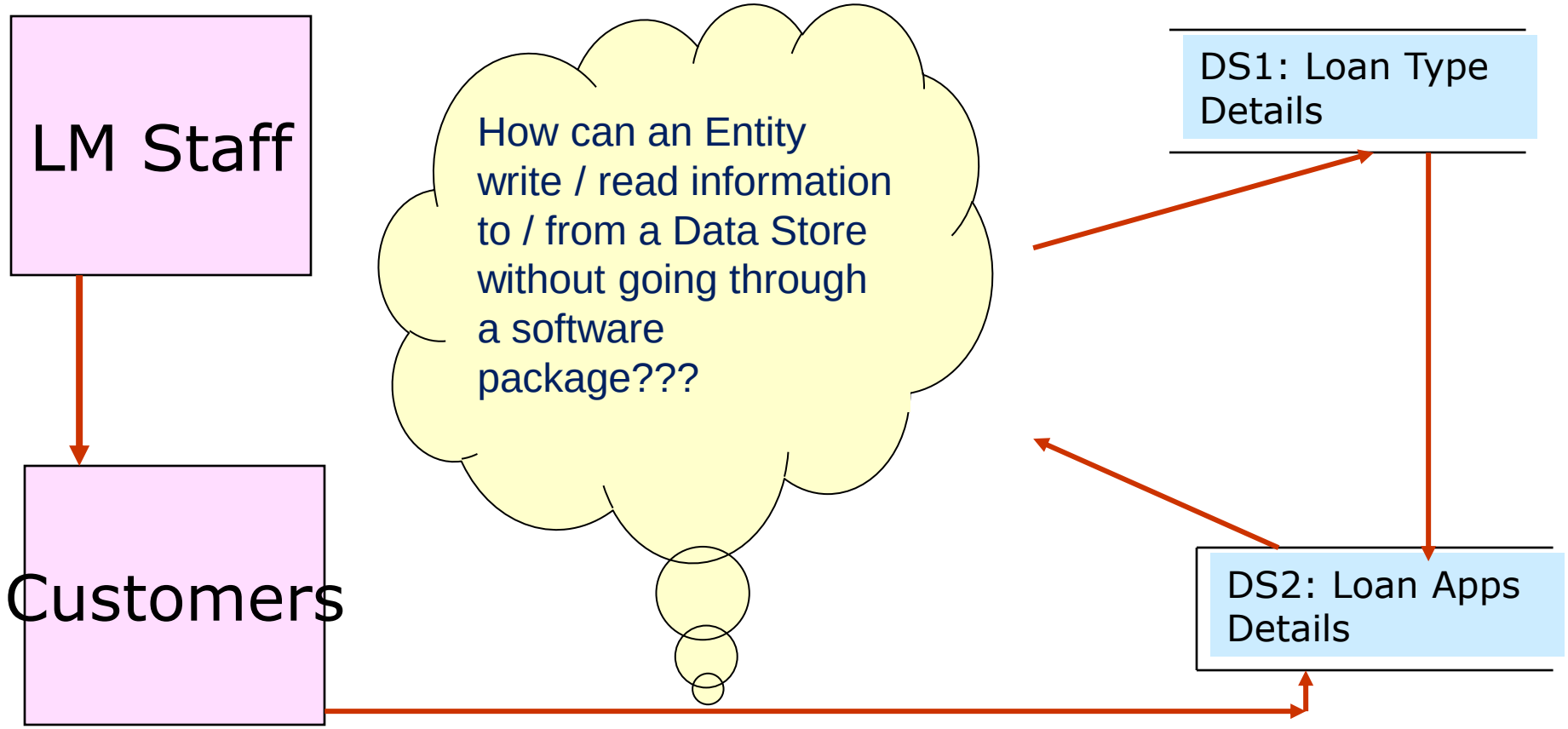
Incorrectness / Inconsistency in a DFD



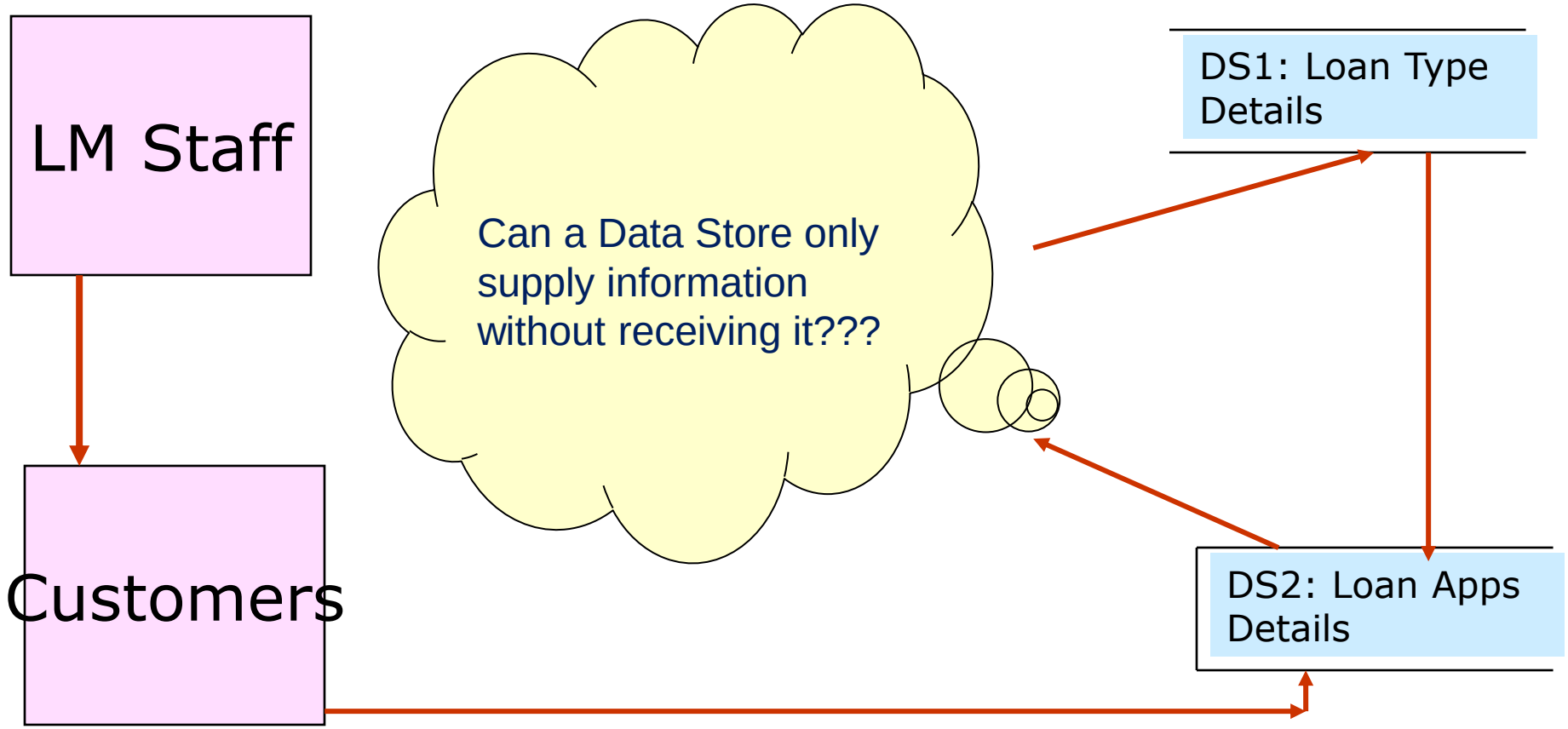
Incorrectness / Inconsistency in a DFD



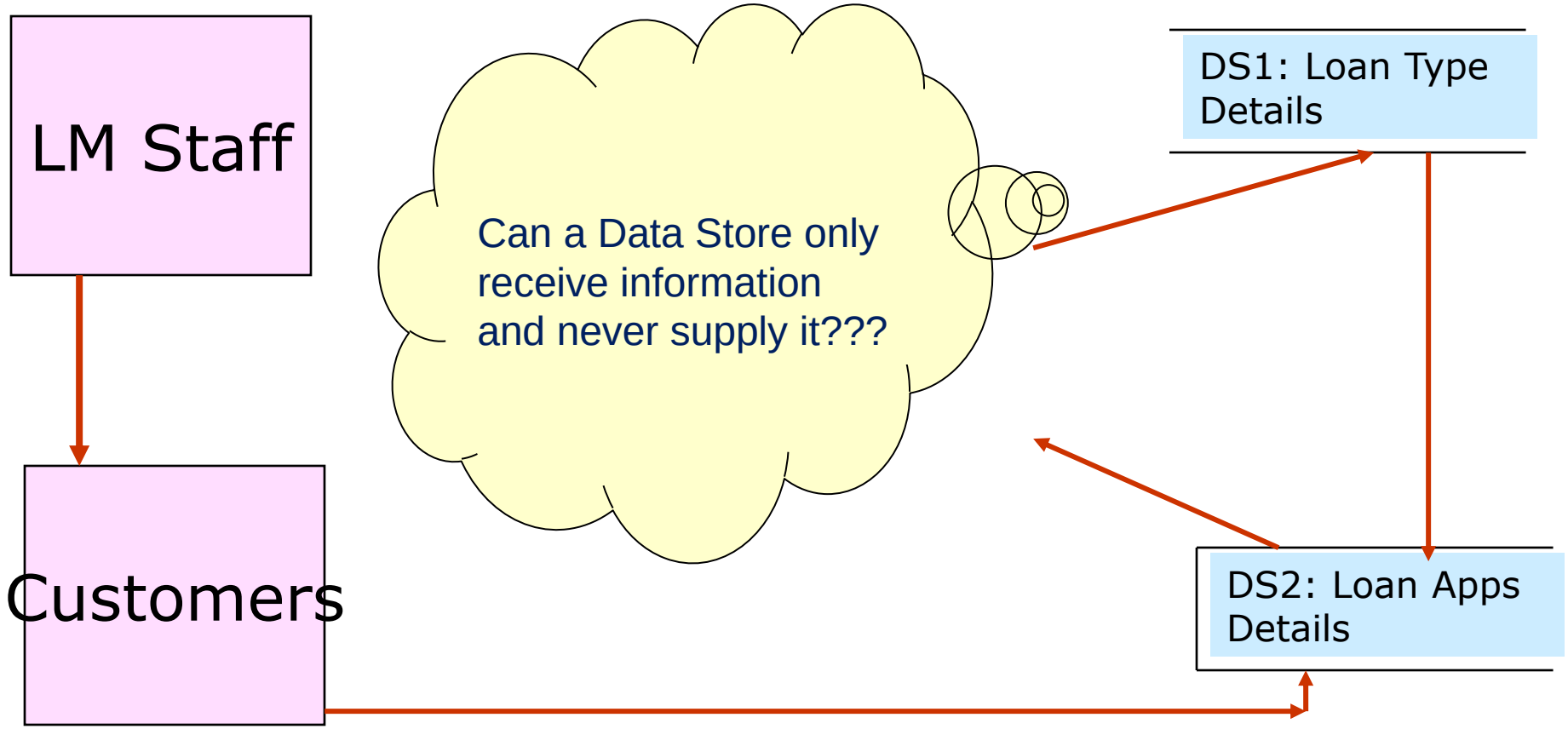
Incorrectness / Inconsistency in a DFD



Incorrectness / Inconsistency in a DFD



Incorrectness / Inconsistency in a DFD



Incorrectness / Inconsistency in a DFD