Requirements Engineering

Instructor: Mehroze Khan

Requirements

- Standish (1995) asked the survey respondents to explain the causes of the failed projects. The top factors were reported to be
    1. Incomplete requirements (13.1%)
    2. Lack of user involvement (12.4%)
    3. Lack of resources (10.6%)
    4. Unrealistic expectations (9.9%)
    5. Lack of executive support (9.3%)
    6. Changing requirements and specifications (8.7%)
    7. Lack of planning (8.1%)
    8. System no longer needed (7.5%)

Requirements Engineering

- The **requirements** for a system are the descriptions of the **services** that a system should provide and the **constraints** on its operation.
- These requirements reflect the needs of customers for a system that serves a certain purpose such as controlling a device, placing an order, or finding information.
- The process of finding out, analyzing, documenting and checking these services and constraints is called **requirements engineering (RE).**

User Requirements

- User requirements are **statements**, in a natural language plus **diagrams**, of what services the system is expected to provide to system users and the constraints under which it must operate.
- The user requirements may vary from **broad statements** of the system features required to **detailed, precise descriptions** of the system functionality.

System Requirements

- System requirements are more **detailed descriptions** of the software system’s functions, services, and operational constraints.
- The **system requirements document** (sometimes called a functional specification) should define exactly what is to be implemented.

Example

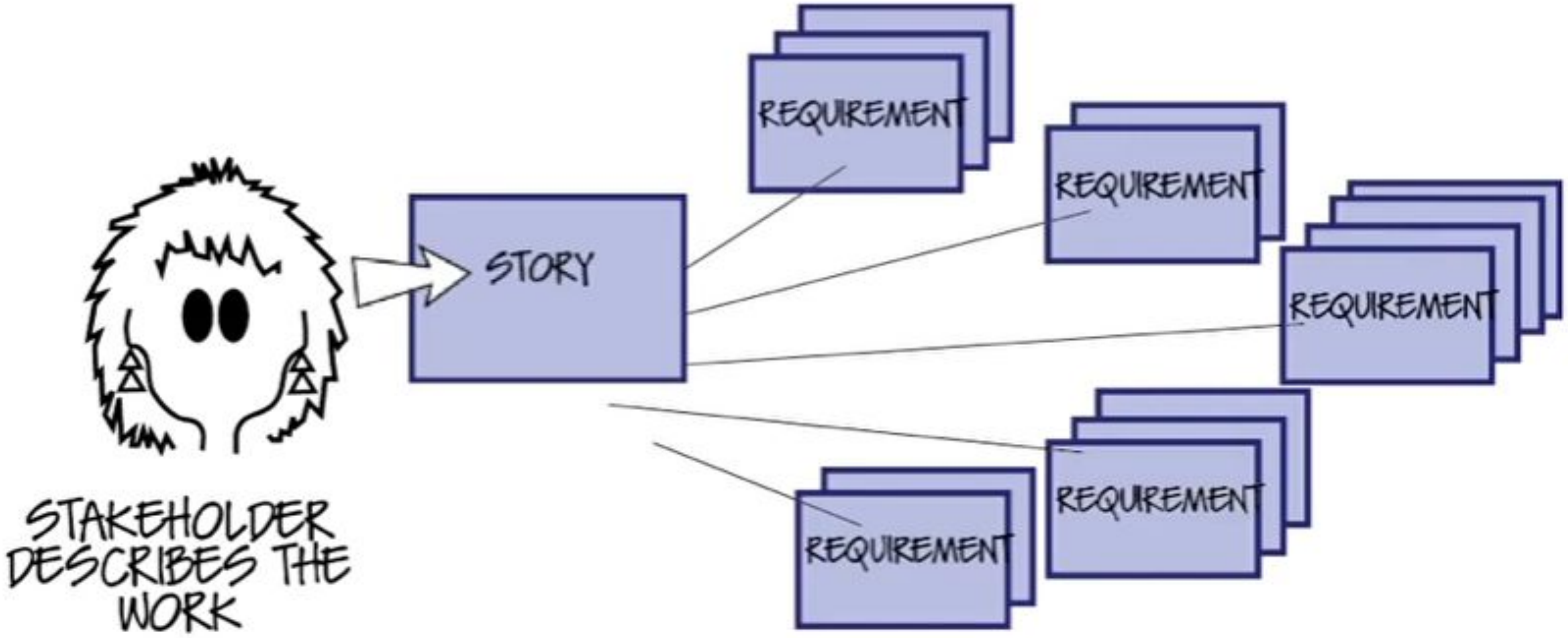
User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Requirement Engineering



Writing User Story

- **Template:**

As a < type of user >, I want < some goal > so that < some reason >

- As a forgetful user, I want a straightforward way to reset my password in case I forget it.

Acceptance Criteria:

- On the login page, the user clicks on the "Forgot Password" link.
- The user is prompted to enter their registered email address.
- After entering the email, the user receives a password reset link via email.
- Clicking the reset link directs the user to a page where they can securely set a new password.
- The user successfully logs in with the new password.
- If the user enters an unregistered email, an appropriate error message is displayed.
- The reset link expires after a reasonable time or upon successful use.

Writing User Story

- As an online gamer, I want to have a multiplayer option so that I can play online with friends.
- As a design team lead, I want to organize assets, so I can keep track of multiple creative projects.
- As an e-commerce shopper, I want to filter my searches so I can find products quickly.
- As a trainer, I want my profile to list my upcoming classes and include a link to a detailed page about each so that prospective attendees can find my courses.
- As a site visitor, I want to access old news that is no longer on the home page, so I can access things I remember from the past.

Requirements Engineering Tasks

- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Management

Requirements Engineering Tasks

- **Inception**: understand problem, people, nature of solution, effectiveness of communication.
- **Elicitation**: Ask questions about objectives, targets, detailed requirements
- **Elaboration**: Identify software function, behavior, information. Develop Requirements Model!!! Analysis???
- **Negotiation**: Resolve conflicts. Prioritize Requirements!!!
- **Specification**: Write document containing requirements models, scenarios etc.
- **Validation**: Ensure that all requirements have been stated, unambiguously!
- **Management**: Identify, control, track requirements and changes

Inception

- Identify: all stakeholders, measurable benefits of successful implementation, possible alternatives
- Ask questions stepwise, as early as possible, first meeting/encounter
- Possible questions at 1st step (stakeholders, overall goals and benefits):
  - Who is behind the request for this work?
  - Who will use this solution?
  - What will be the economic benefit of a successful solution?

Inception

- Possible questions at 2nd step (detailed understanding and customer perception about the solution):
  - What problems(s) will this solution address?
  - Can you show me (or describe) the business environment in which the solution will be used?
  - How do you characterize the ‘good’ output?
- Possible questions at 3rd step (effectiveness of communication):
  - Are you the right person to answer these questions?
  - Are my questions relevant to the problem that you have?
  - Can anyone else provide additional information?

Elicitation

- Get more detailed requirements.
- Customers do not always understand what their needs and problems are.
- It is important to discuss the requirements with everyone who has a stake in the system.
- Come up with agreement on what the requirements are
    - If we cannot agree on what the requirements are, then the project is doomed to fail

Elicitation

- Different Stakeholders are:
  - **Clients**: pay for the software to be developed
  - **Customers**: buy the software after it is developed
  - **Users**: use the system
  - **Domain experts**: familiar with the problem that the software must automate
  - **Market Researchers**: conduct surveys to determine future trends and potential customers
  - **Lawyers or auditors**: familiar with government, safety, or legal requirements
  - **Software engineers** or other technology experts

Elaboration

- Analyze, model, specify…
- Some Analysis Techniques
    - Data Flow Diagrams (DFD)
    - Use case Diagram
    - Object Models (ER Diagrams)
    - State Diagrams
    - Sequence Diagrams
    - Activity Diagrams

Requirements Modeling

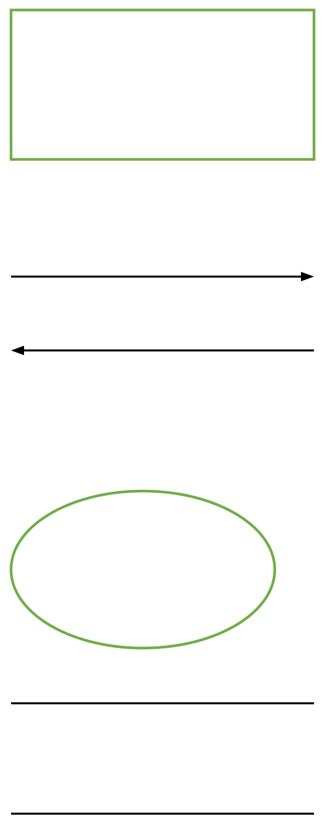
1. Scenario-based Models
   - User Stories
   - Use Case Diagram
2. Class-oriented Models
   - Class Diagram
   - CRC Cards
3. Behavioral Models
   - State Diagram
   - Sequence Diagram
4. Flow-oriented Models
   - Data Flow Diagram

Data Flow Diagram (DFD)

- A Data Flow Diagram (DFD) is a traditional **visual representation of the information flows within a system**.
- By drawing a Data Flow Diagram, you can tell the information provided by and delivered to someone who takes part in system processes, the information needed to complete the processes and the information needed to be stored and accessed.
- The DFD is presented in a **hierarchical fashion**. That is, the first data flow model (sometimes called a **level 0 DFD** or ***context diagram***) represents the system as a whole.
- Subsequent data flow diagrams refine the context diagram, providing increasing detail with each subsequent level.

Symbols used in DFD

- **Square Box:** A square box defines **external entities** (source or sink) of the system. Source supplies data to system and sink receives data from system.
- **Arrow or Line:** An arrow identifies the **data flow** i.e. it gives information to the data that is in motion. Connects processes, external entities and data stores.
- **Circle or bubble chart:** It represents a **process** that gives us information. It is also called processing box.
- **Open Rectangle:** An open rectangle is a **data store** where data is stored. Data can be read from or written to the data store



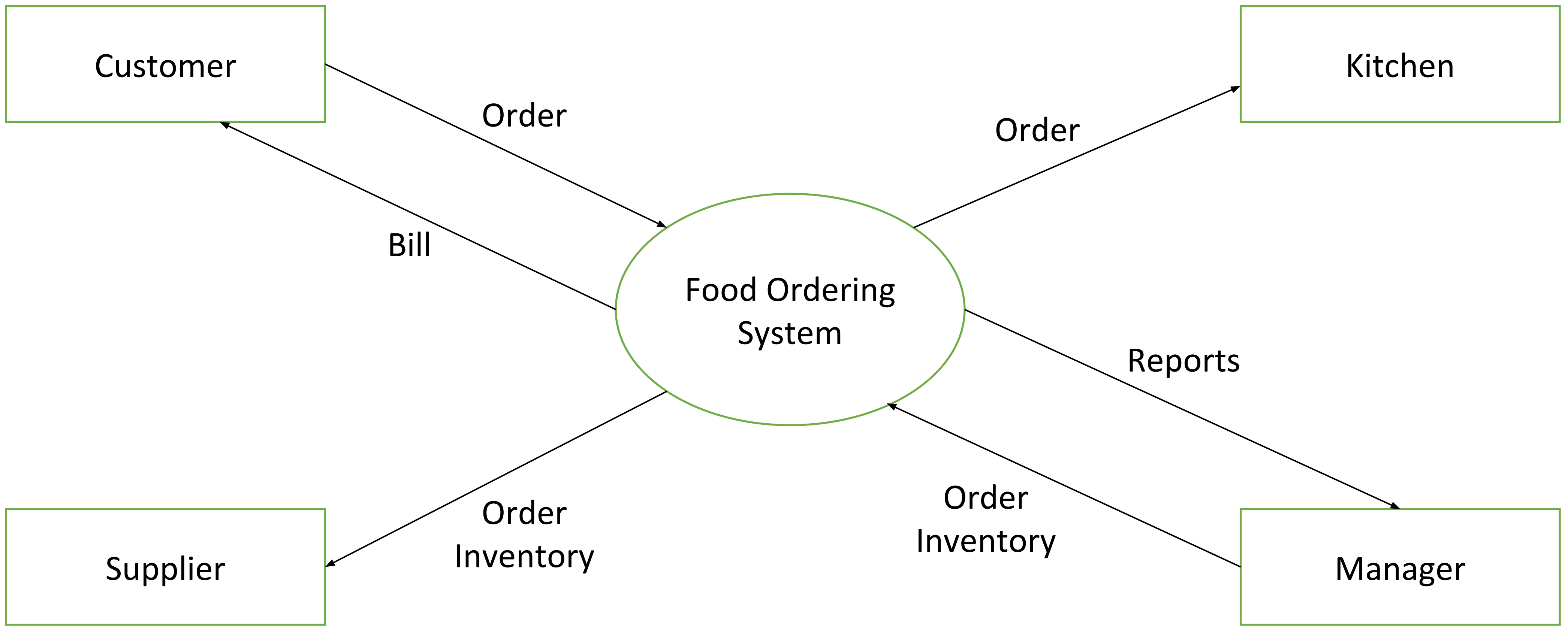
Level-0 DFD

- A context diagram is a data flow diagram that only shows the **top level**, otherwise known as Level 0.
- At this level, there is only **one visible process node** that represents the functions of a complete system regarding how it interacts with external entities.
- Context DFD is the entrance of a data flow model. It contains one and only one process and does not show any data store.
- A verb in requirements narrative is potentially a process, a noun is either data, or external entity, or data store.

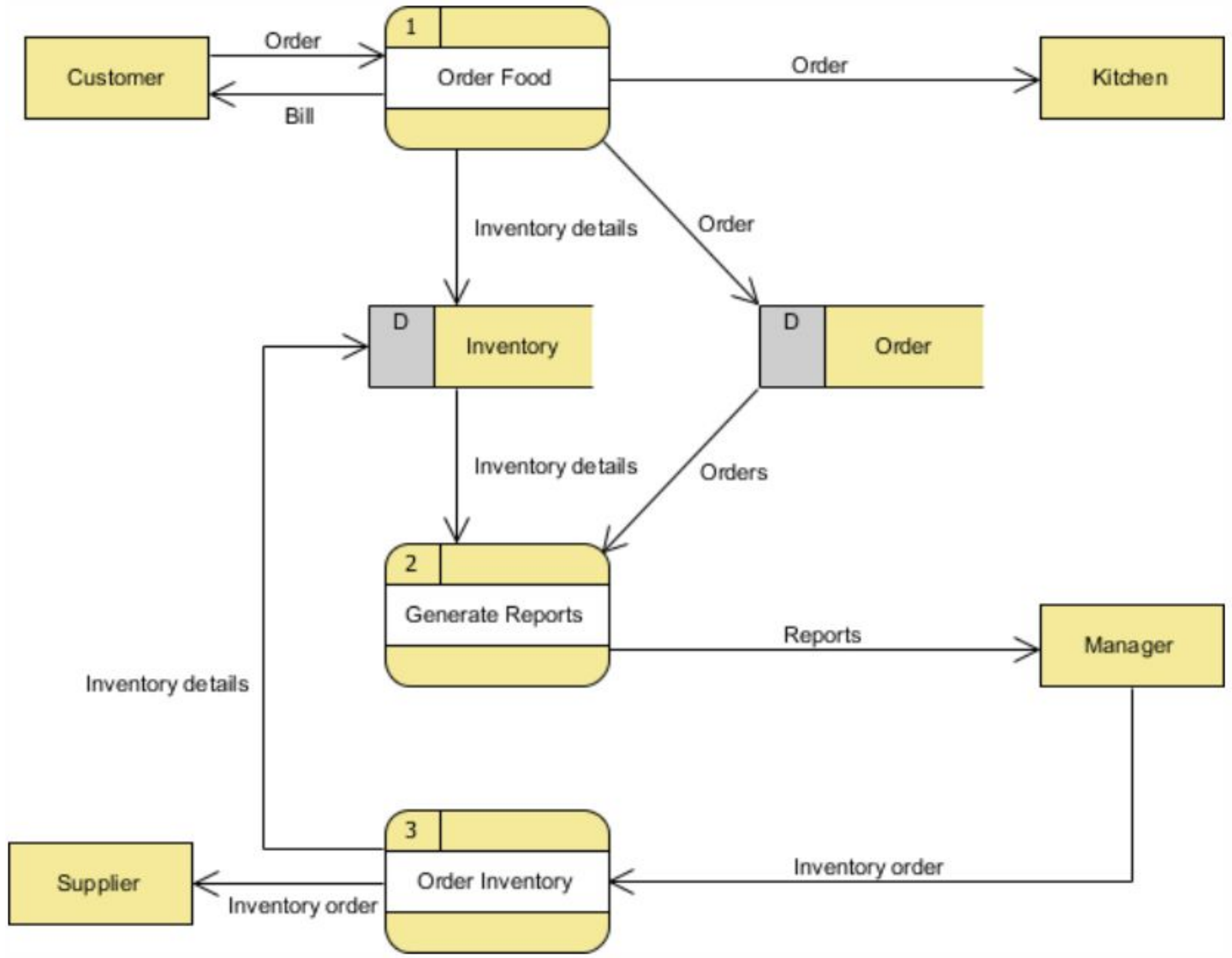
Level-0 DFD (Food Ordering System)

- It contains a process that represents the system to model, in this case, the "*Food Ordering System*".
- It also shows the participants who will interact with the system, called the external entities. In this example, the *Supplier*, *Kitchen*, *Manager*, and *Customer* are the entities who will interact with the system.
- In between the process and the external entities, there is data flow (connectors) that indicate the existence of information exchange between the entities and the system.

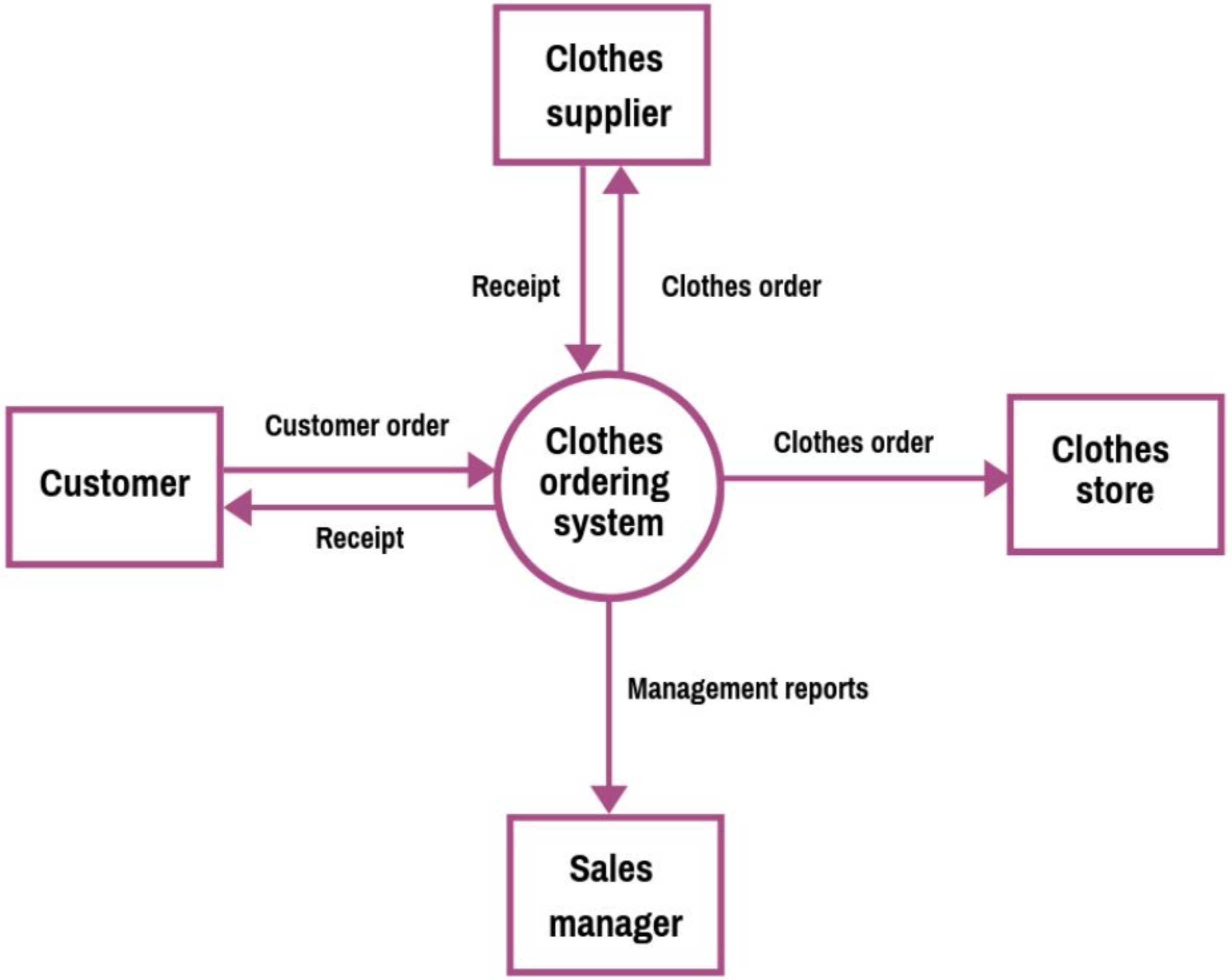
Level-0 DFD (Food Ordering System)



Level-1 DFD (Food Ordering System)



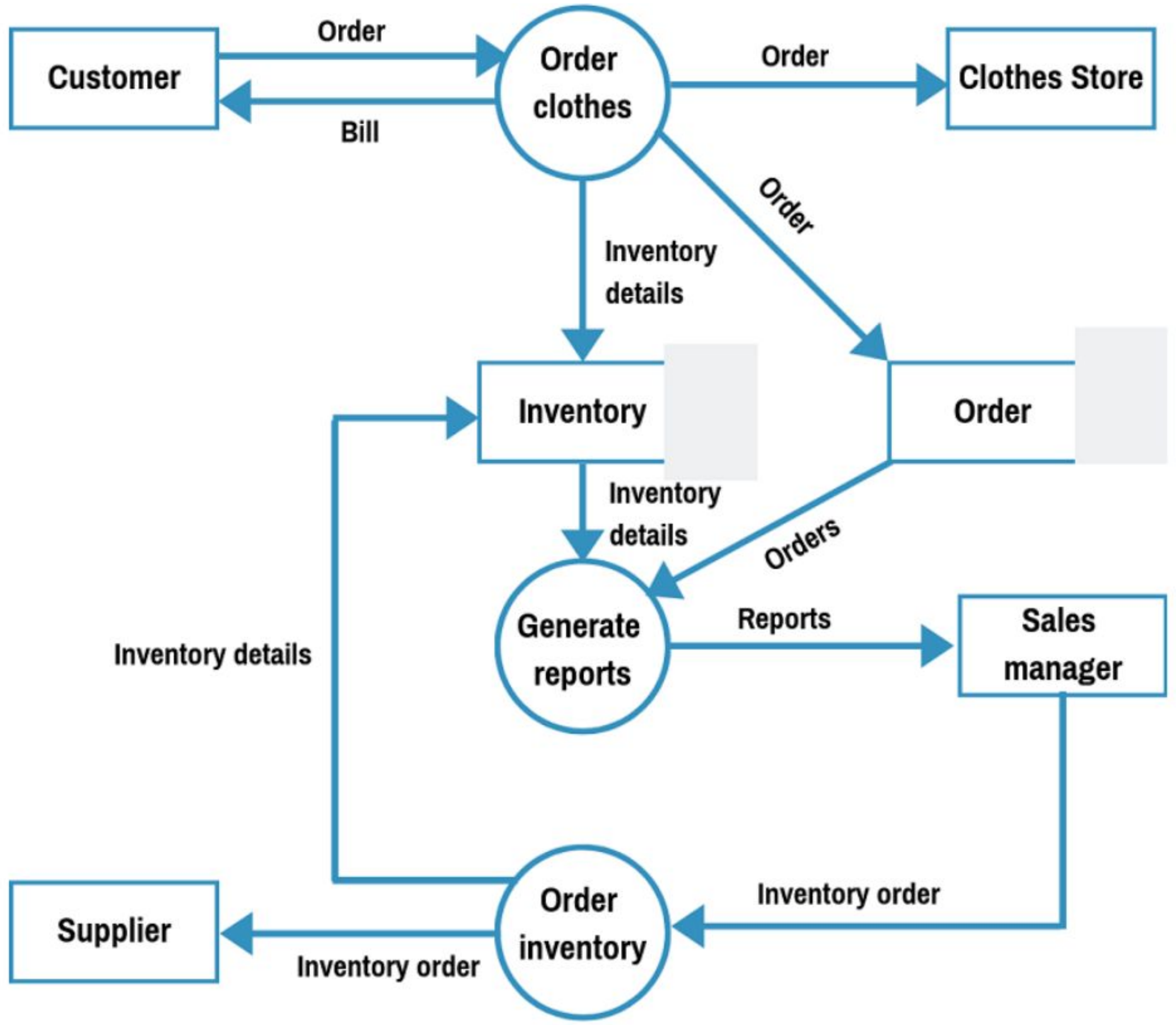
Level-0 DFD (Clothes Ordering System)



Level-1 DFD (Clothes Ordering System)

- **Step 1:** Define the processes.
- The three processes are: *Order Clothes, Generate Reports, and Order Inventory.*
- **Step 2:** Create the list of all external entities.
- The external entities are *Customer, Clothes Store, Sales Manager, and Supplier*
- **Step 3:** Create the list of the data stores.
- These are *Order and Inventory*
- **Step 4:** Create the list of the data flows
- Data flows are *Order, Bill, Order, Order, Inventory details, Inventory details, Orders, Reports, Inventory Order, Inventory Order, Inventory details.*
- Now, connect the rectangles with arrows signifying the data flows.
- If data flows both ways between any two rectangles, create two individual arrows.
- **Step 5:** Create the diagram.

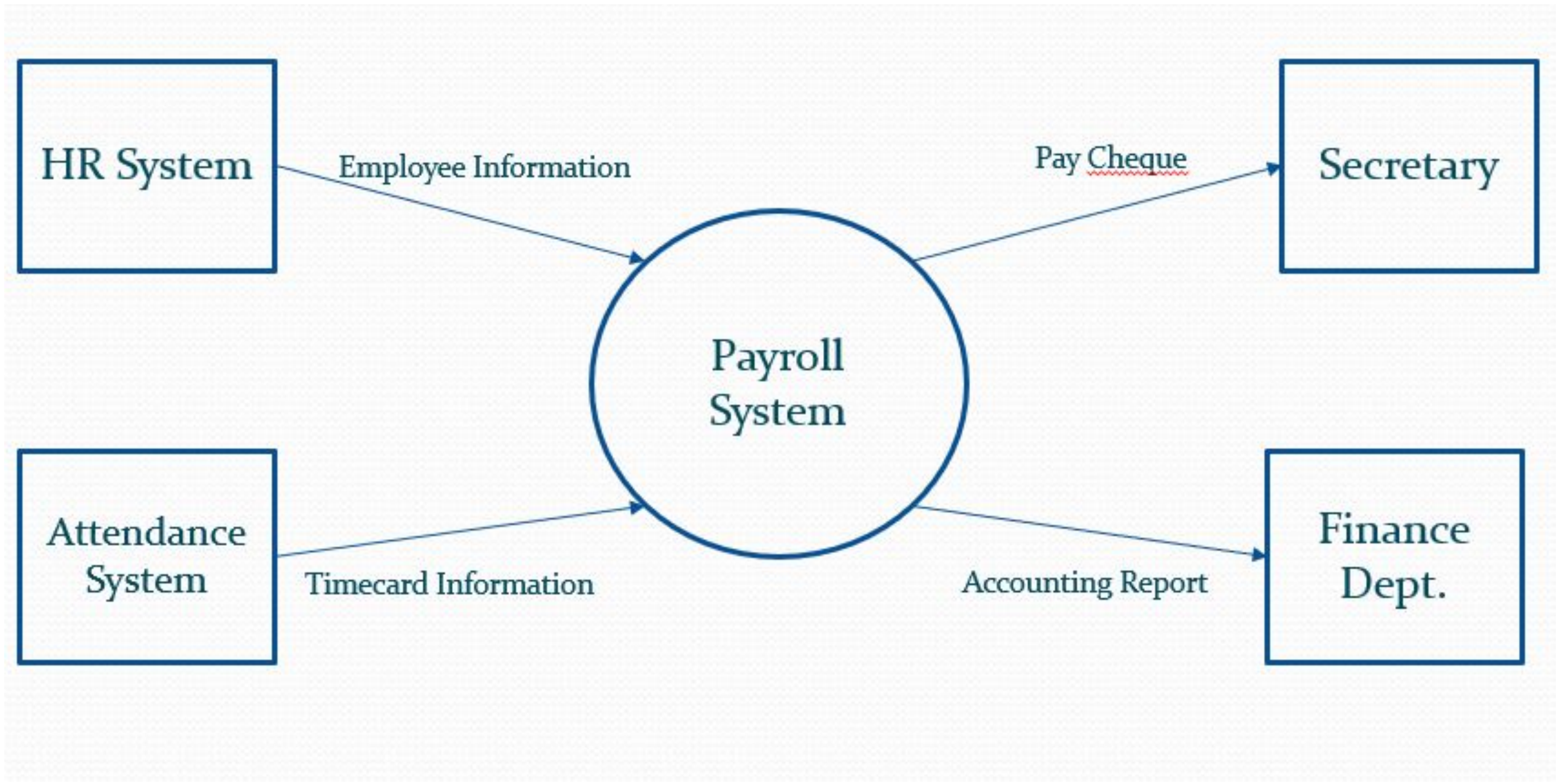
Level-1 DFD (Clothes Ordering System)



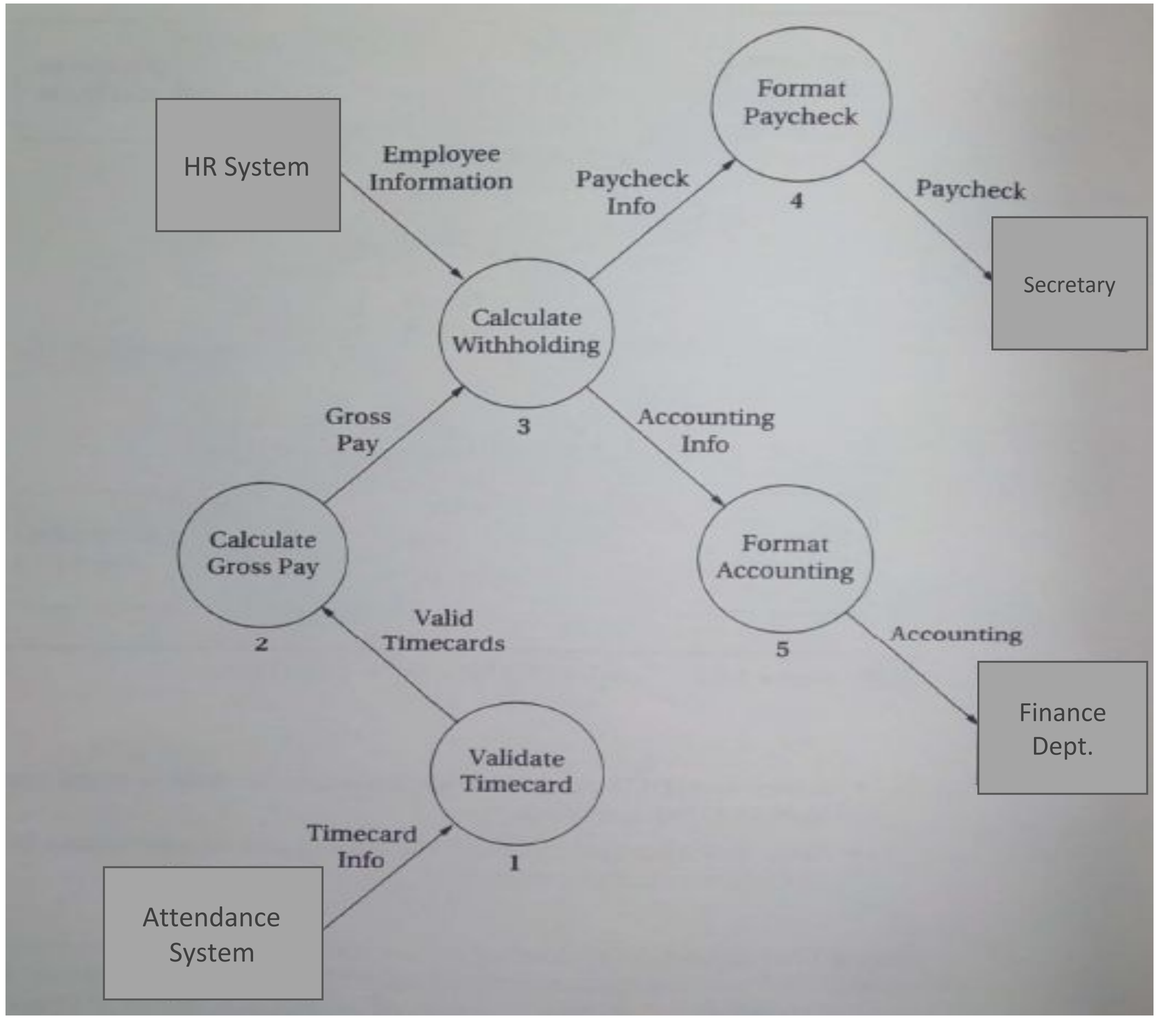
Level-1 DFD (Clothes Ordering System)

- Level-0 DFD contains only one process and does not illustrate any data store. This is the main difference with level 1 DFD.
- Level 1 DFD breaks down the main process into subprocesses that can then be seen on a deeper level. Also, level 1 DFD contains data stores that are used by the main process.

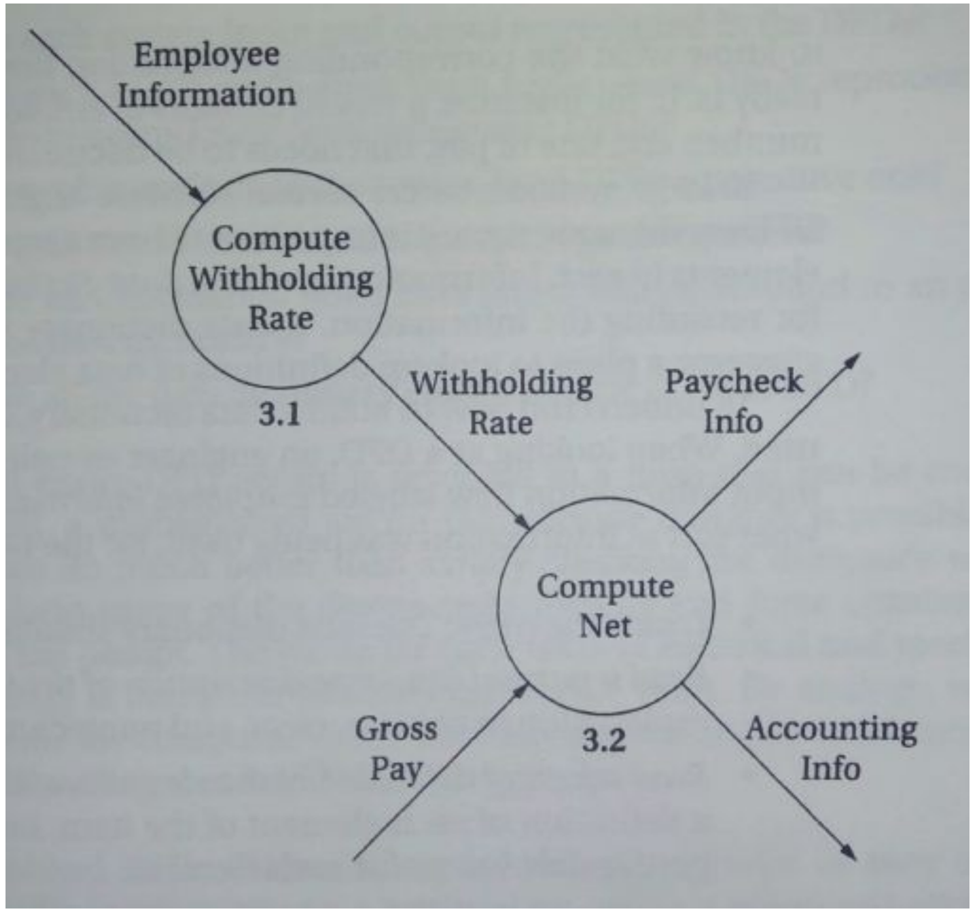
Level-0 DFD (Payroll System)



Level-1, Level-2 DFD (Payroll System)



Level-1



Level-2

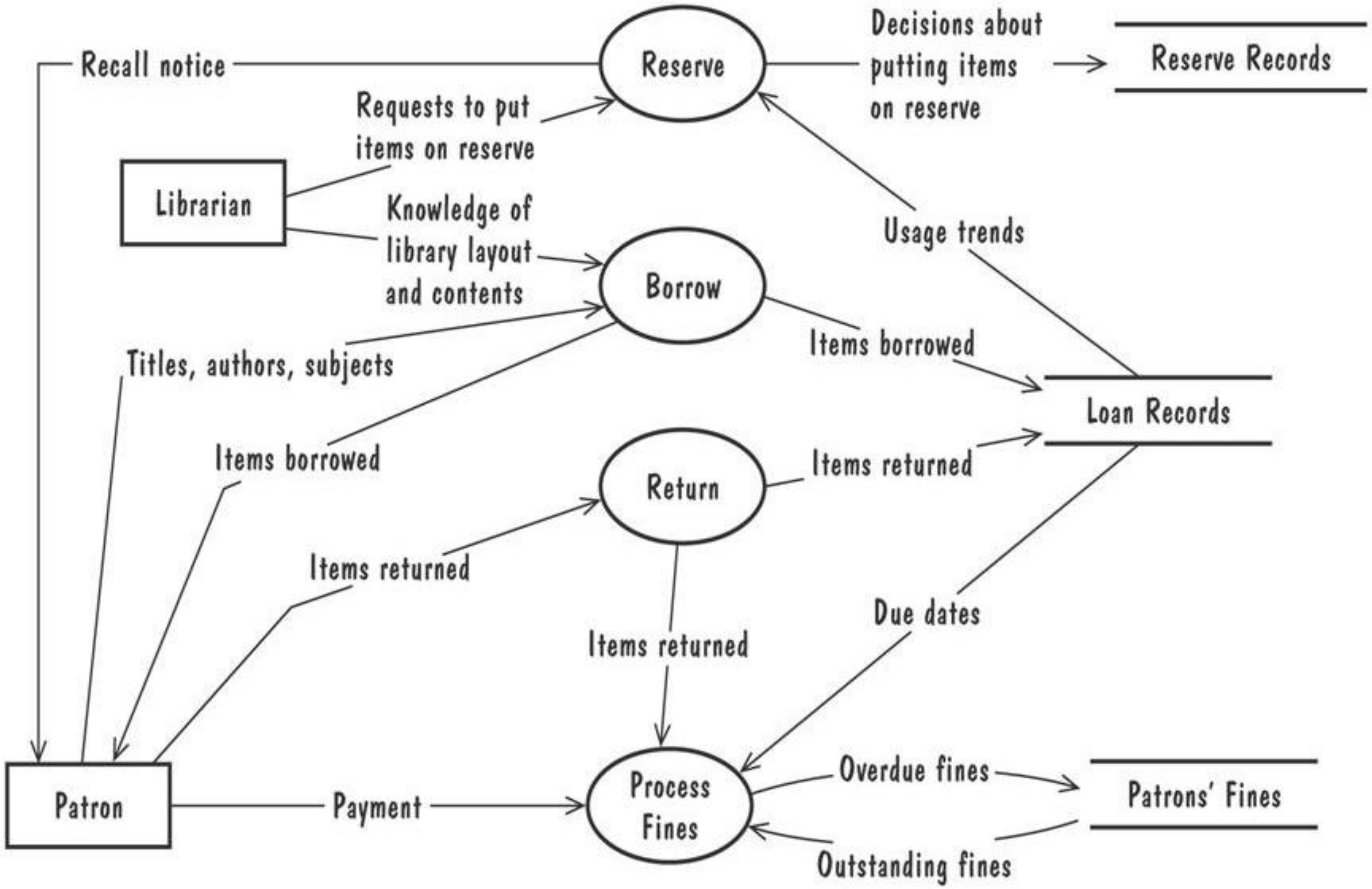
Library Management System

- Consider a library management system where circulation services allow the Patron to borrow or return a book/item/publication without involvement of Librarian, so the system needs to process these requests. The Librarian controls which items can be borrowed and how can the items be shown to the patrons. The system processes fines also when a borrowed item is returned after the due date. The librarian is able to reserve an item if required and a recall notice is sent to the patron as a consequence. The record of reserved items is kept so that the librarian can view which items have been reserved already.

Library Management System

- Consider a library management system where circulation services allow the Patron to borrow or return a book/item/publication without involvement of Librarian, so the system needs to process these requests. The Librarian controls which items can be borrowed and how can the items be shown to the patrons. The system processes fines also when a borrowed item is returned after the due date. The librarian is able to reserve an item if required and a recall notice is sent to the patron as a consequence. The record of reserved items is kept so that the librarian can view which items have been reserved already.

Library Management System (Level-1 DFD)



References

1. Roger S. Pressman, Software Engineering A Practitioner's Approach, 9th Edition. McGrawHill
2. Shari PFleeger, Joanne Atlee, Software Engineering: Theory and Practice, 4th Edition
3. Roger S. Pressman, Software Engineering A Practitioner's Approach, 5th Edition. McGrawHill