EDU
EAST DELTA
UNIVERSITY

CSE 411
Software Engineering and System Analysis and Design

Topic 5: Software Requirements

Software Requirement Specifications

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS)

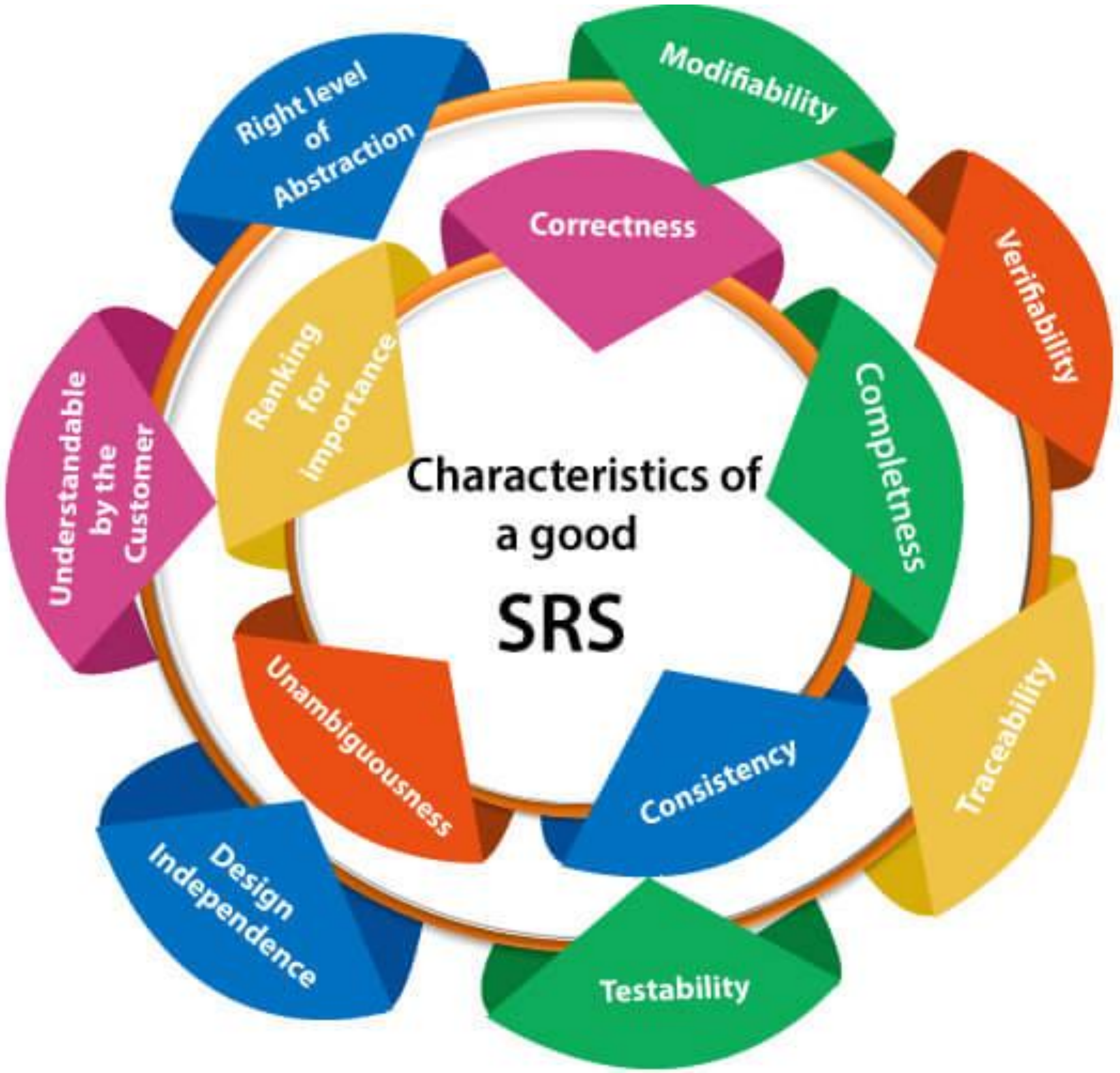
SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements.

It serves several goals depending on who is writing it.

- First, the SRS could be written by the client of a system.
- Second, the SRS could be written by a developer of the system.

The two methods create entirely various situations and establish different purposes for the document altogether.

- The first case, SRS, is used to define the needs and expectation of the users.
- The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

Characteristics of a good SRS
Correctness
Modifiability
Verifiability
Completeness
Traceability
Consistency
Testability
Unambiguousness
Design Independence
Understandable by the Customer
Ranking for importance
Right level of Abstraction

Characteristics

Imp

Following are the features of a good SRS document:

1. **Correctness**: User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

2. **Completeness**: The SRS is complete if, and only if, it includes the following elements:

- All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces are present.
- Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

3. Consistency: The SRS is consistent if, and only if, no subset of individual requirements described in its conflict.

4. Unambiguousness: SRS is unambiguous when every fixed requirement has only one interpretation.

5. Ranking for importance and stability: The SRS is ranked for importance and stability of each requirement. Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable.

6. **Modifiability**: SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.

7. **Verifiability**: SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

8. **Traceability**: The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

9. Design Independence: There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

10. Testability: An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

11. Understandable by the customer: An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.

12. The right level of abstraction: If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas, for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

Properties of a good SRS document:

Concise: The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.

Structured: It should be well-structured. A well-structured document is simple to understand and modify.

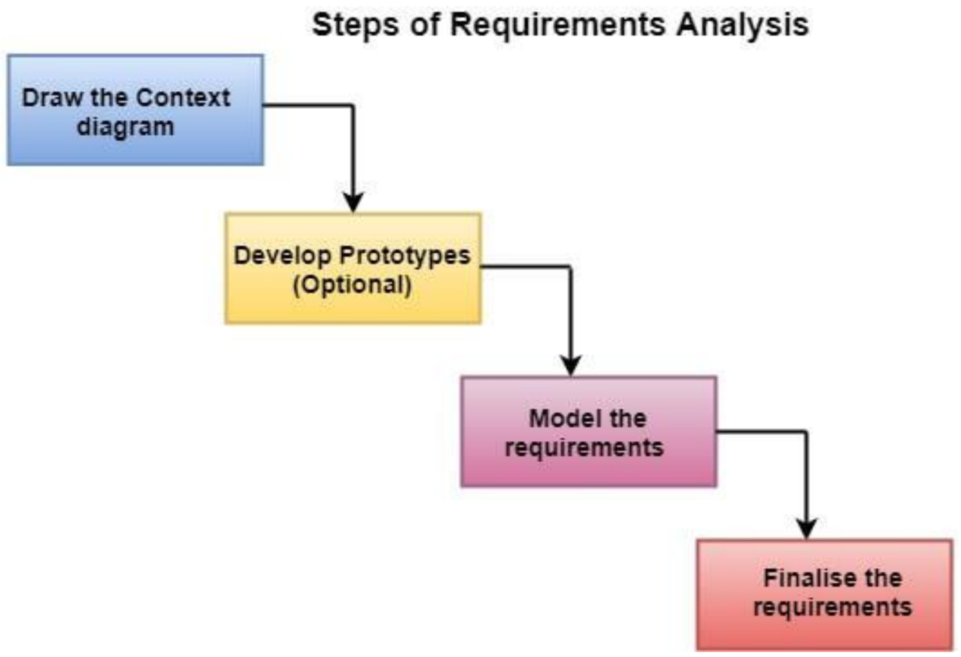
Black-box view: It should only define what the system should do and refrain from stating how to do these.

Conceptual integrity: It should show conceptual integrity so that the reader can simply understand it.

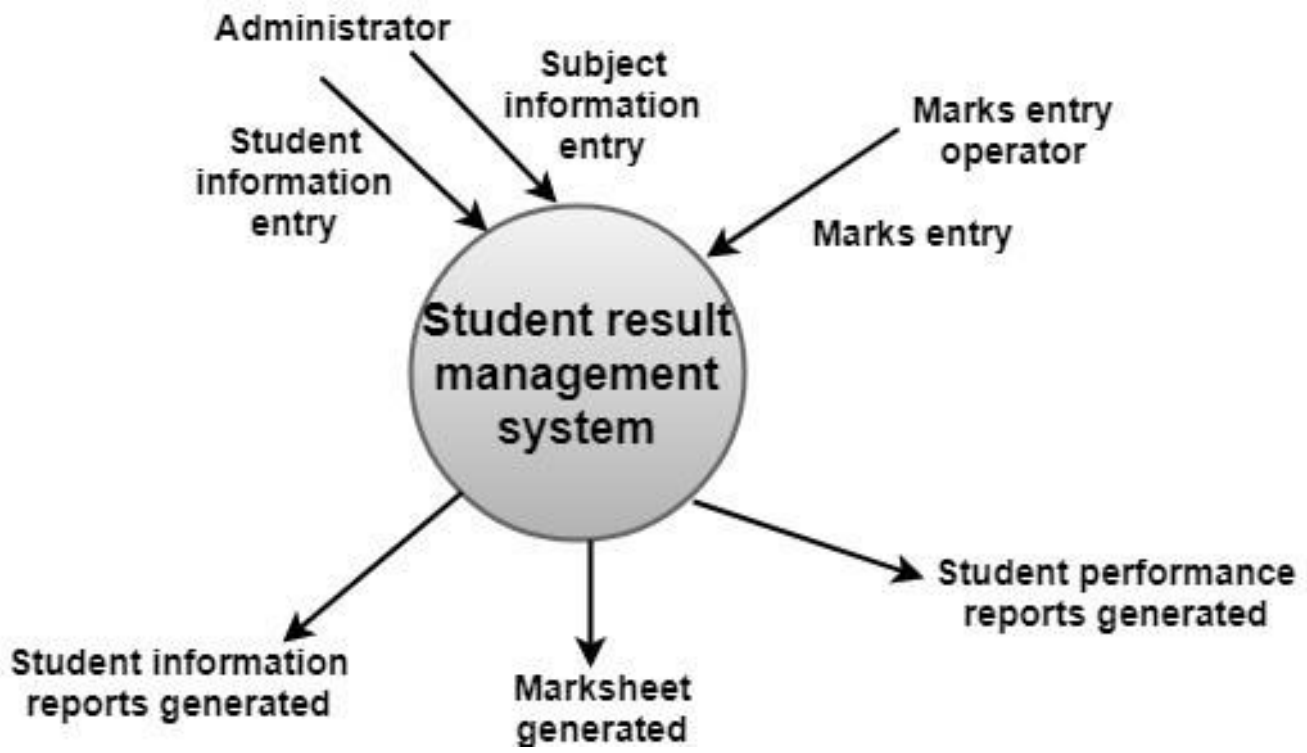
Verifiable: All requirements of the system, as documented in the SRS document, should be correct.

Requirements Analysis

- **Requirement analysis** is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system.
- After the completion of the analysis, it is expected that the understandability of the project may improve significantly.


Steps of Requirements Analysis
Draw the Context diagram
Develop Prototypes (Optional)
Model the requirements
Finalise the requirements

- **Draw the context diagram:** The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world. It identifies the entities outside the proposed system that interact with the system.
- The context diagram of student result management system is given below:


Administrator
Student information entry
Subject information entry
Marks entry operator
Marks entry
Student result management system
Student information reports generated
Marksheet generated
Student performance reports generated

- **Development of a Prototype (optional):** One effective way to find out what the customer wants is to construct a prototype, something that looks and preferably acts as part of the system they say they want.

- We can use their feedback to modify the prototype until the customer is satisfied continuously. Hence, the prototype helps the client to visualize the proposed system and increase the understanding of the requirements.

- The prototype should be built quickly and at a relatively low cost.

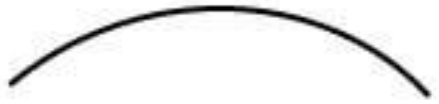
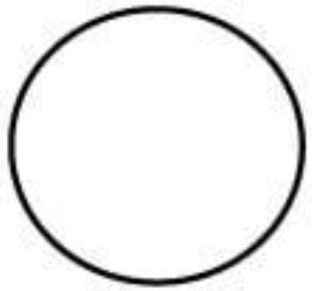
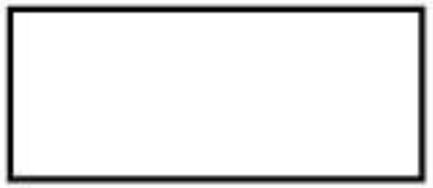
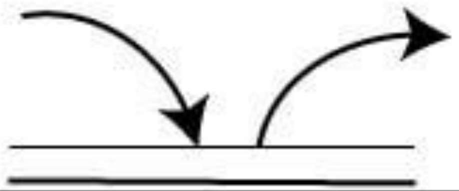
Model the requirements: This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships between them. The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements. Such models include the Data Flow diagram, Entity-Relationship diagram, Data Dictionaries, State-transition diagrams, etc.

Finalize the requirements: After modeling the requirements, we will have a better understanding of the system behavior. Now we finalize the analyzed requirements, and the next step is to document these requirements in a prescribed format.

Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infect, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

Rules for Data Flow Diagram

Data can not flow between two entities

Data flow must be from entity to a process or a process to an entity. There can be multiple data flows between one entity and a process.

Data can not flow between two data stores

Data flow must be from data store to a process or a process to an data store. Data flow can occur from one data store to many process.

Rules for Data Flow Diagram

Data can not flow directly from an entity to data store

Data Flow from entity must be processed by a process before going to data store and vice versa.

A process must have at least one input data flow and one output data flow

Every process must have input data flow to process the data and an output data flow for the processed data.

Rules for Data Flow Diagram

A data store must have at least one input data flow and one output data flow

Every data store must have input data flow to store the data and an output data flow for the retrieved data.

All the process in the system must be linked to minimum one data store or any other process.

0-level DFDM

It represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows.

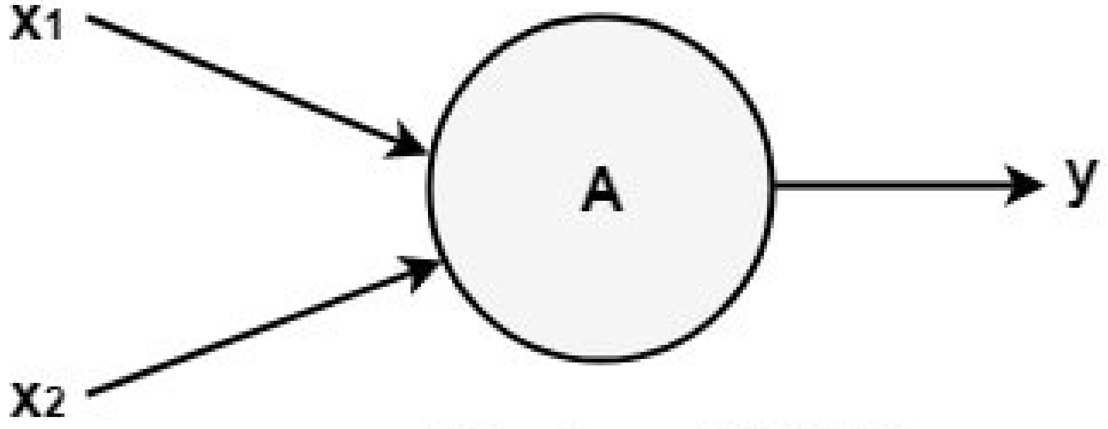


Fig: Level-0 DFD.

DFD for Library Management System

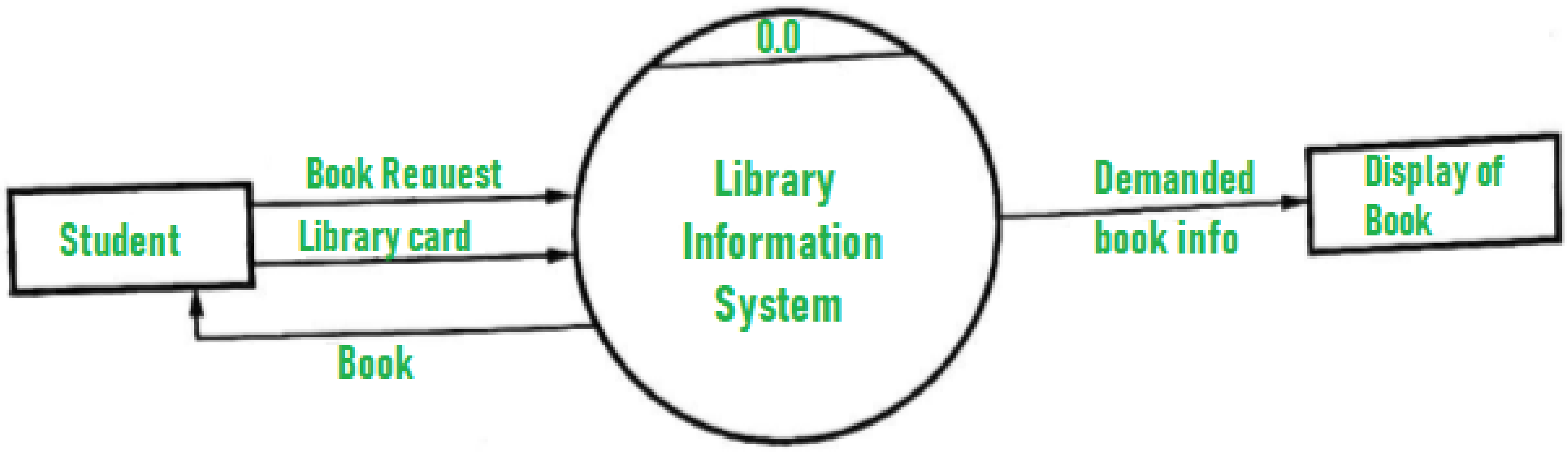
Data Flow Diagram (DFD) depicts the flow of information and the transformation applied when a data moves in and out from a system. The overall system is represented and described using input, processing and output in the DFD. The inputs can be:

- **Book request when a student requests for a book.**
- **Library card when the student has to show or submit his/her identity as a proof.**

The overall processing unit will contain the following output that a system will produce or generate:

- **Book will be the output as the book demanded by the student will be given to them.**
- **Information of demanded book should be displayed by the library information system that can be used by the student while selecting the book which makes it easier for the student.**

DFD for Library Management System (LEVEL 0)



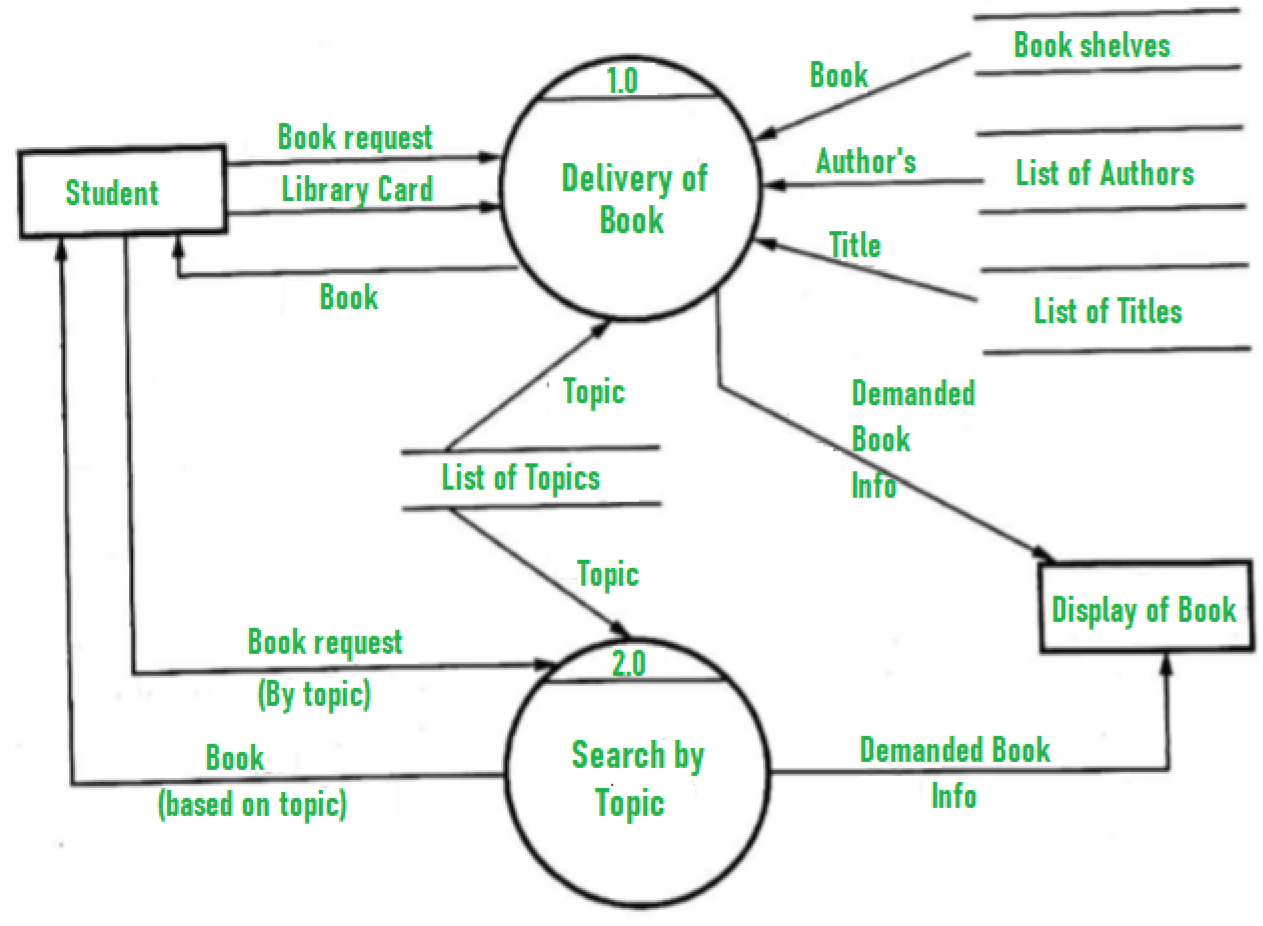
DFD for Library Management System (LEVEL 1)

At this level, the system has to show or exposed with more details of processing. The processes that are important to be carried out are:

- **Book delivery**
- **Search by topic**

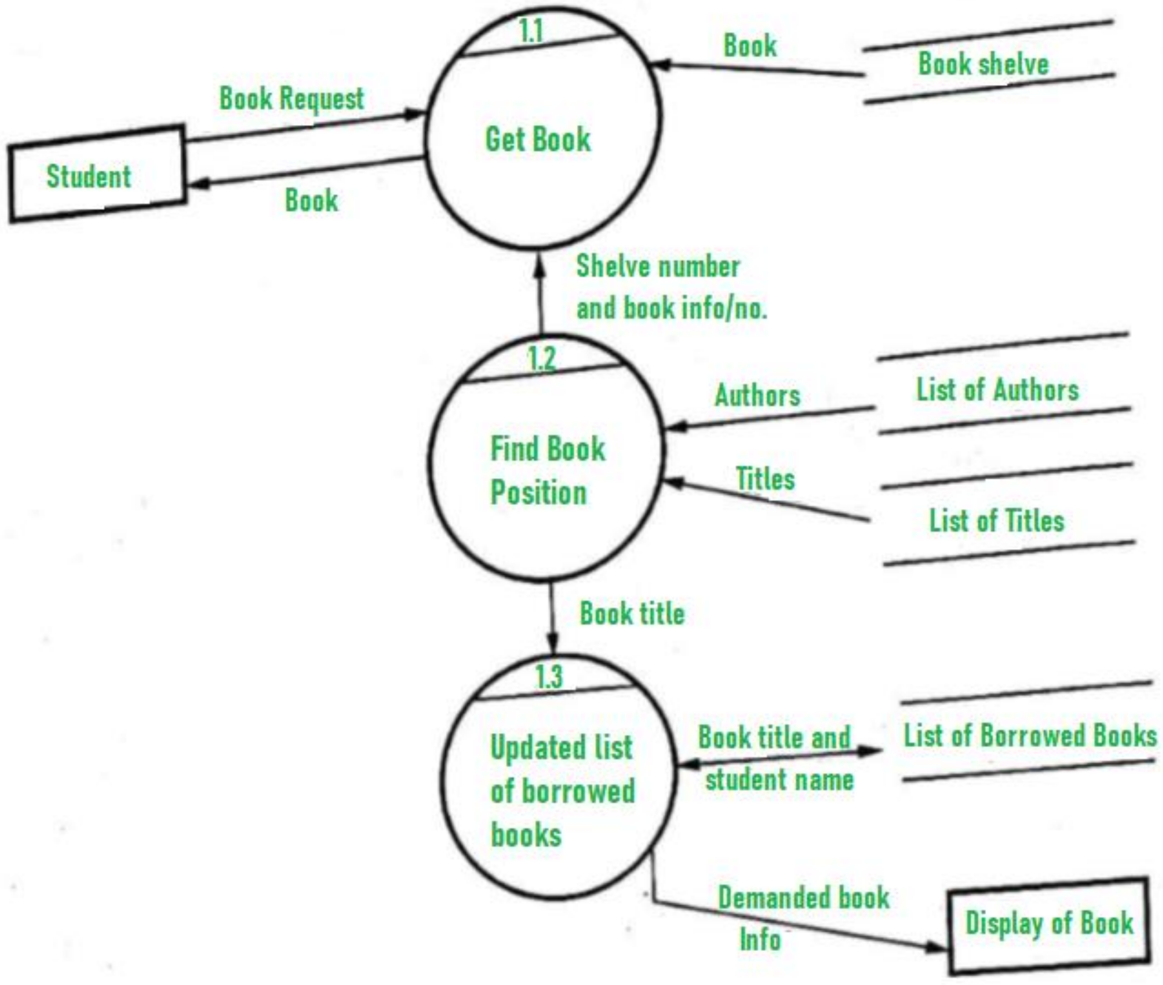
List of authors, List of Titles, List of Topics, the bookshelves from which books can be located are some information that is required for these processes. Data store is used to represent this type of information.

DFD for Library Management System (LEVEL 1)



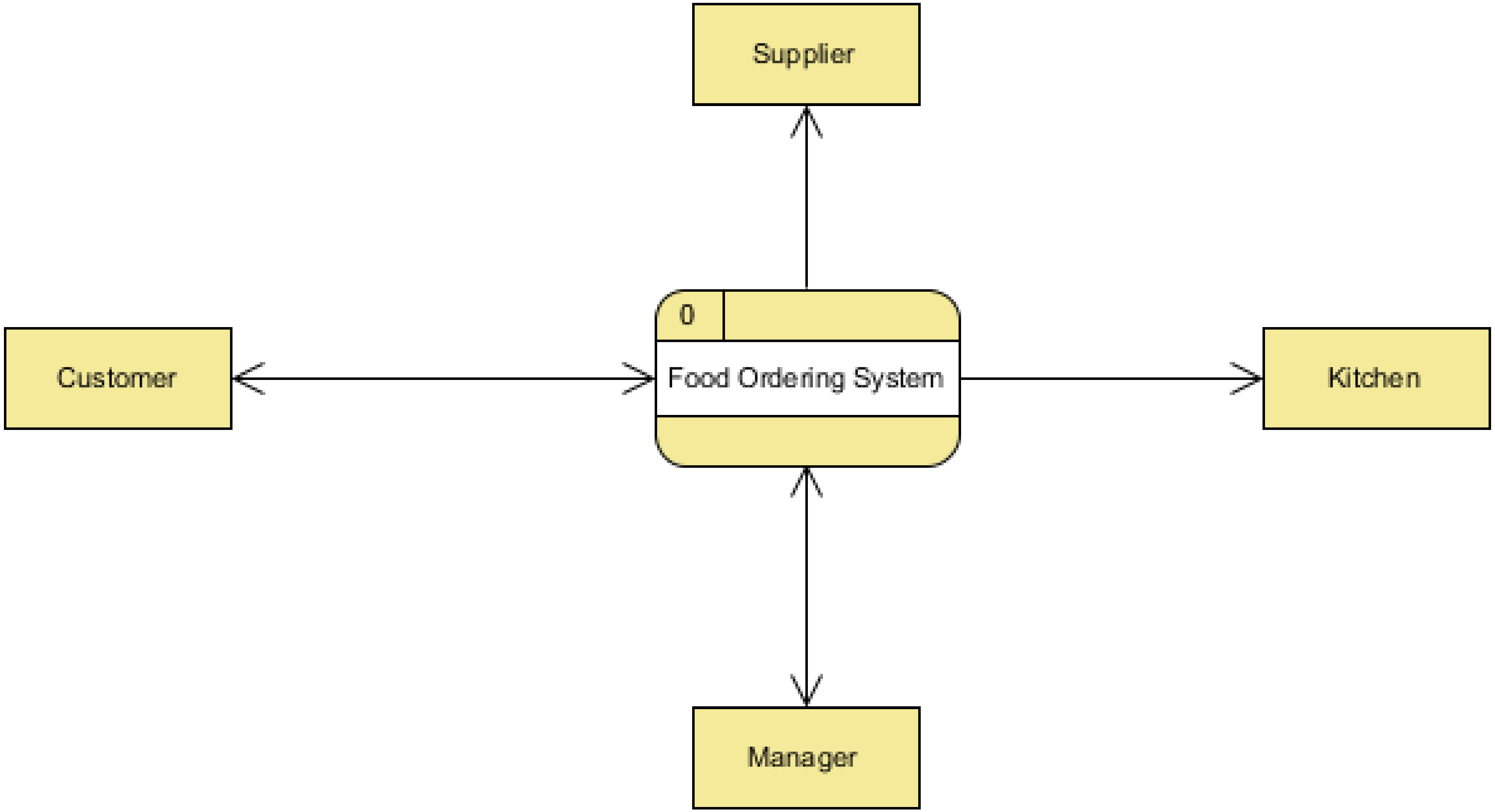
Level 1 DFD

DFD for Library Management System (LEVEL 2)

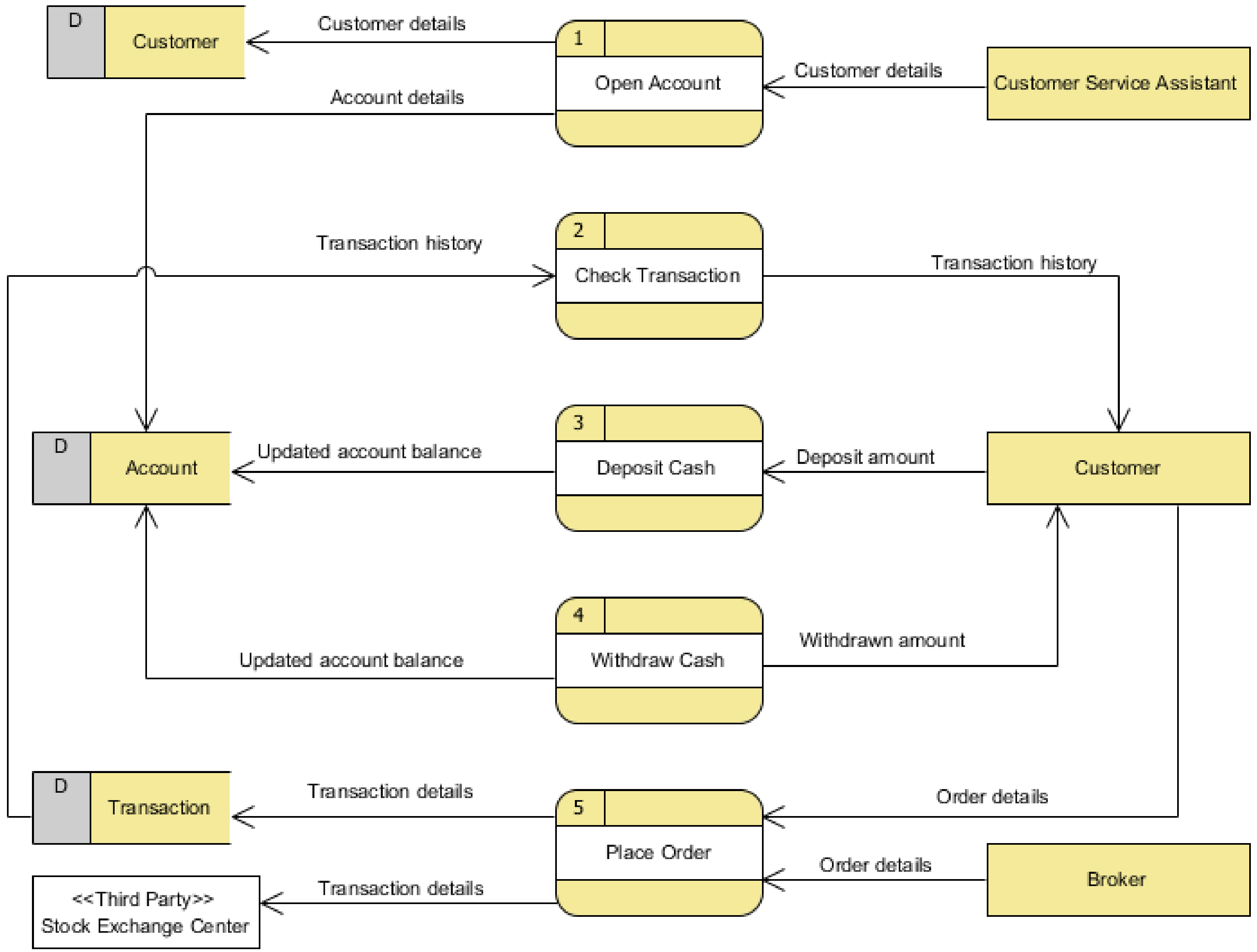


Level 2 DFD

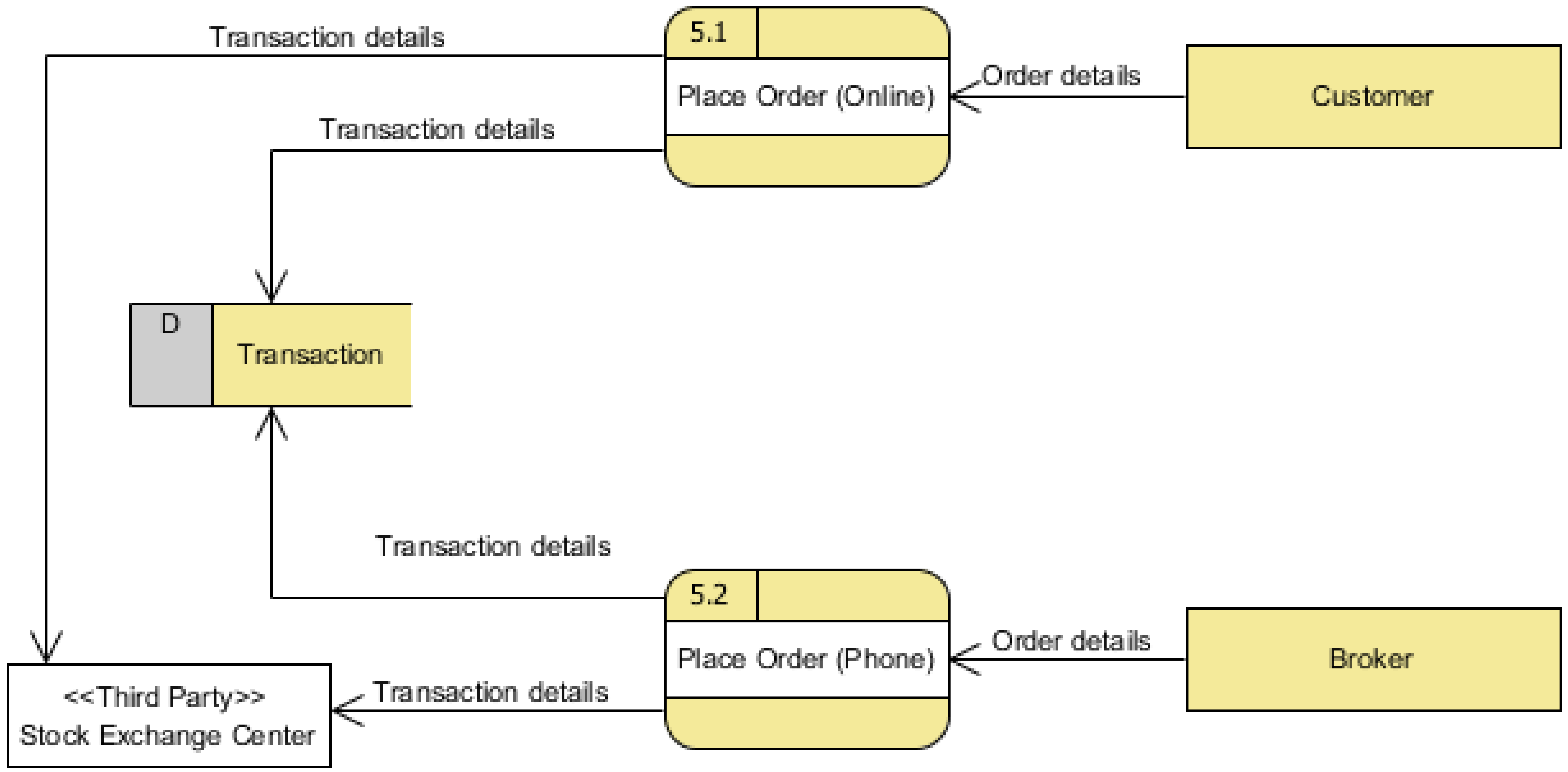
DFD for Food Ordering System (LEVEL 0)



DFD for Food Ordering System (LEVEL 1)



DFD for Food Ordering System (LEVEL 2)



DFD Exercise

- DFD Example - Customer Service C
- DFD Example - Food Ordering C
- DFD Example - Securities Trading C /Broker
- DFD Example - Supermarket App
- DFD Example - Vehicle Maintenance Depot Mechanics / Customer
- DFD Example - Video rental

https://www.visual-paradigm.com/tutorials/data-flow-diagram-example-cs-system.jsp

End of Topic 5