Requirements Engineering

Software requirements engineering

Is a process of discovery, refinement, modeling, and specifying customer requirements. The challenge facing software engineers is deep:

How can we ensure that we have specified a system that properly meets the customer's needs and satisfies the customer's expectations? there is no foolproof answer to this difficult question, but a solid requirements engineering process is the best solution we currently have.

Who does it?

Both the software engineer and the customer take an active role in software requirements engineering.

Requirements engineering encompasses seven distinct tasks: inception, elicitation, elaboration, negotiation, specification, validation, and management.

1. Inception

At project inception, you establish a basic understanding of the problem, the people who want a solution, and the nature of the solution that is desired.

2. elicitation

The intent of elicitation is to transfer ideas from stakeholders to the software team smoothly and without delay.

3. Elaboration

The elaboration task focuses on developing a refined requirements model that identifies various aspects of software function, behavior, and information.

4. Negotiation

customers and users sometimes ask for more than can be achieved, and also propose conflicting requirements. These conflicts need to be corrected through the process of negotiation.

5. Specification

A software requirements specification (SRS) is a document that is created when a detailed description of all aspects of the software to be built must be specified before the project is to begin.

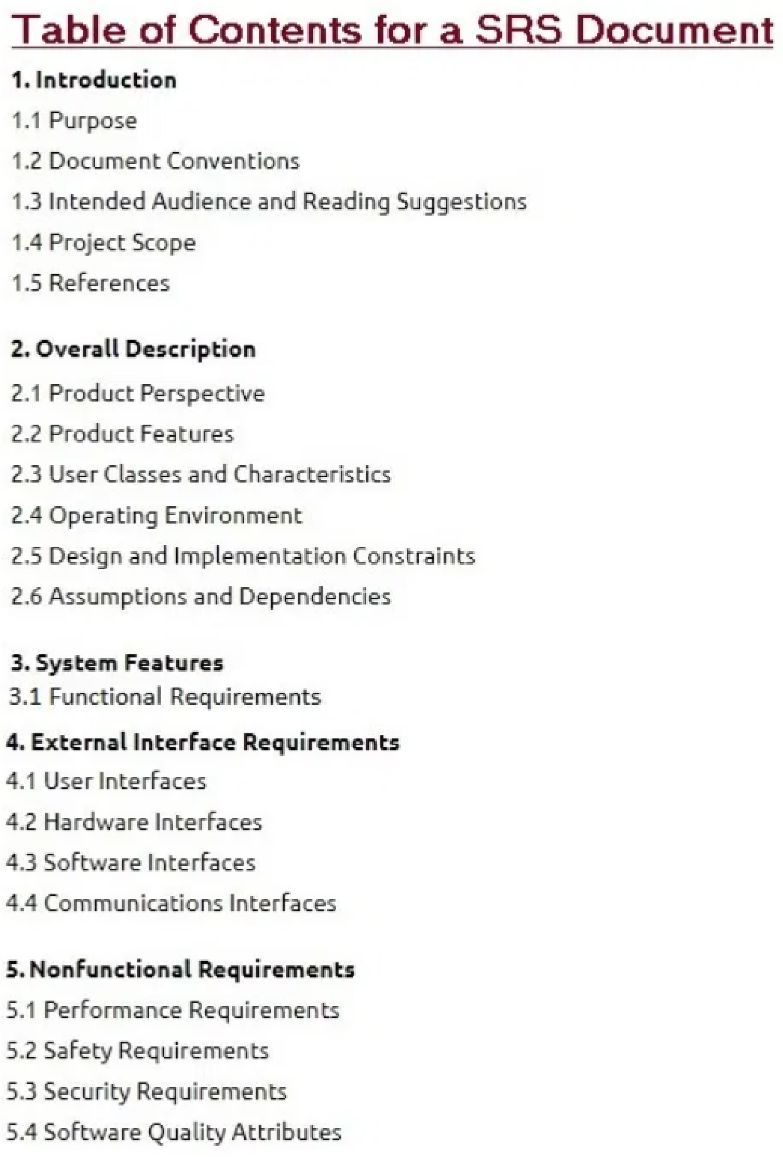


Table of Contents for a SRS Document

1. Introduction
1.1 Purpose
1.2 Document Conventions
1.3 Intended Audience and Reading Suggestions
1.4 Project Scope
1.5 References

2. Overall Description
2.1 Product Perspective
2.2 Product Features
2.3 User Classes and Characteristics
2.4 Operating Environment
2.5 Design and Implementation Constraints
2.6 Assumptions and Dependencies

3. System Features
3.1 Functional Requirements

4. External Interface Requirements
4.1 User Interfaces
4.2 Hardware Interfaces
4.3 Software Interfaces
4.4 Communications Interfaces

5. Nonfunctional Requirements
5.1 Performance Requirements
5.2 Safety Requirements
5.3 Security Requirements
5.4 Software Quality Attributes

6. Validation

Requirements validation examines the specification to ensure that all software requirements have been stated clearly; in a consistent manner, and errors have been detected and corrected.

The primary requirements validation mechanism is the technical review.

7. Requirements management

is a set of activities that help the project team identify, control, and track requirements and changes at any time as the project proceeds.

Analysis Concepts And Principles

Requirements analysis is a software engineering task that bridges the gap between system-level requirements engineering and software design

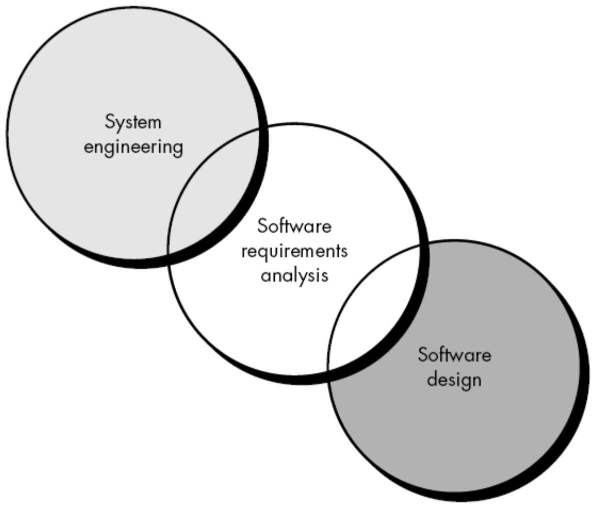


Figure 7.1: Requirements Analysis as a bridge.

Who does it?

A software engineer (sometimes called an "analyst") builds the model using requirements elicited from the customer.

Why is it important?

If you don't analyze, it's highly likely that you'll build a very elegant software solution that solves the wrong problem. The result is wasted time and money, personal disappointment, and unhappy customers.

Requirements Modeling Approaches

A- structured analysis, considers data and the processes that transform the data as separate entities.

B- object-oriented analysis, focuses on the definition of classes and the manner in which they collaborate with one another to affect customer requirements.

The Elements of the analysis model:

1- Scenario-based models e.g. (use cases).

2- Class models e.g. (class diagrams, collaboration diagrams).

3- Flow models e.g. (DFDs, data models).

4- Behavioral models e.g. (state diagrams, sequence diagrams).

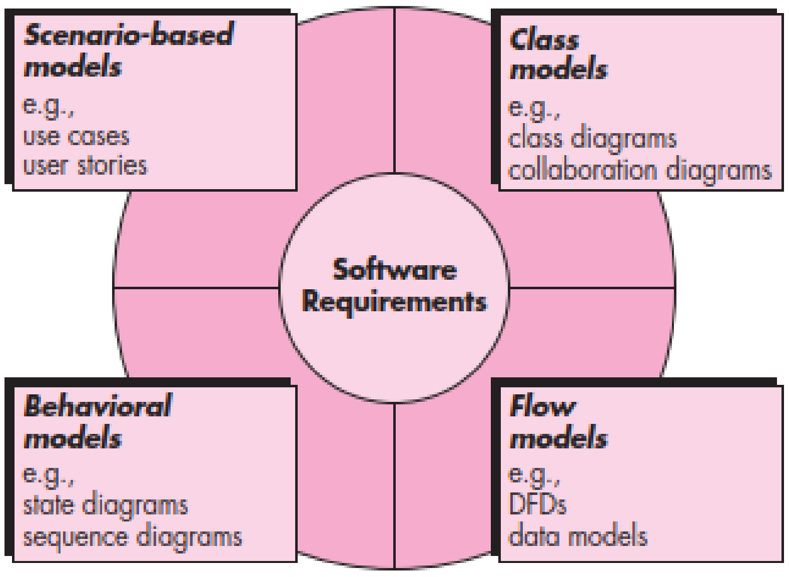


Figure 7.2: Elements of the analysis model

1. Scenario-based modeling

requirements modeling with UML begins with the creation of scenarios in the form of use cases, activity diagrams, and swimlane diagrams.

<u>Use case diagram</u>

It simply describes and displays the relation or interaction between the users and the system.

Here, we will understand the design use case diagram for the library management system. Some scenarios of the system are as follows:

1. User who registers himself as a new user initially is regarded as a staff or student for the library system.
- For the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user.
- After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to the cardholder or user.
2. After getting the library card, a new book is requested by the user as per their requirement.

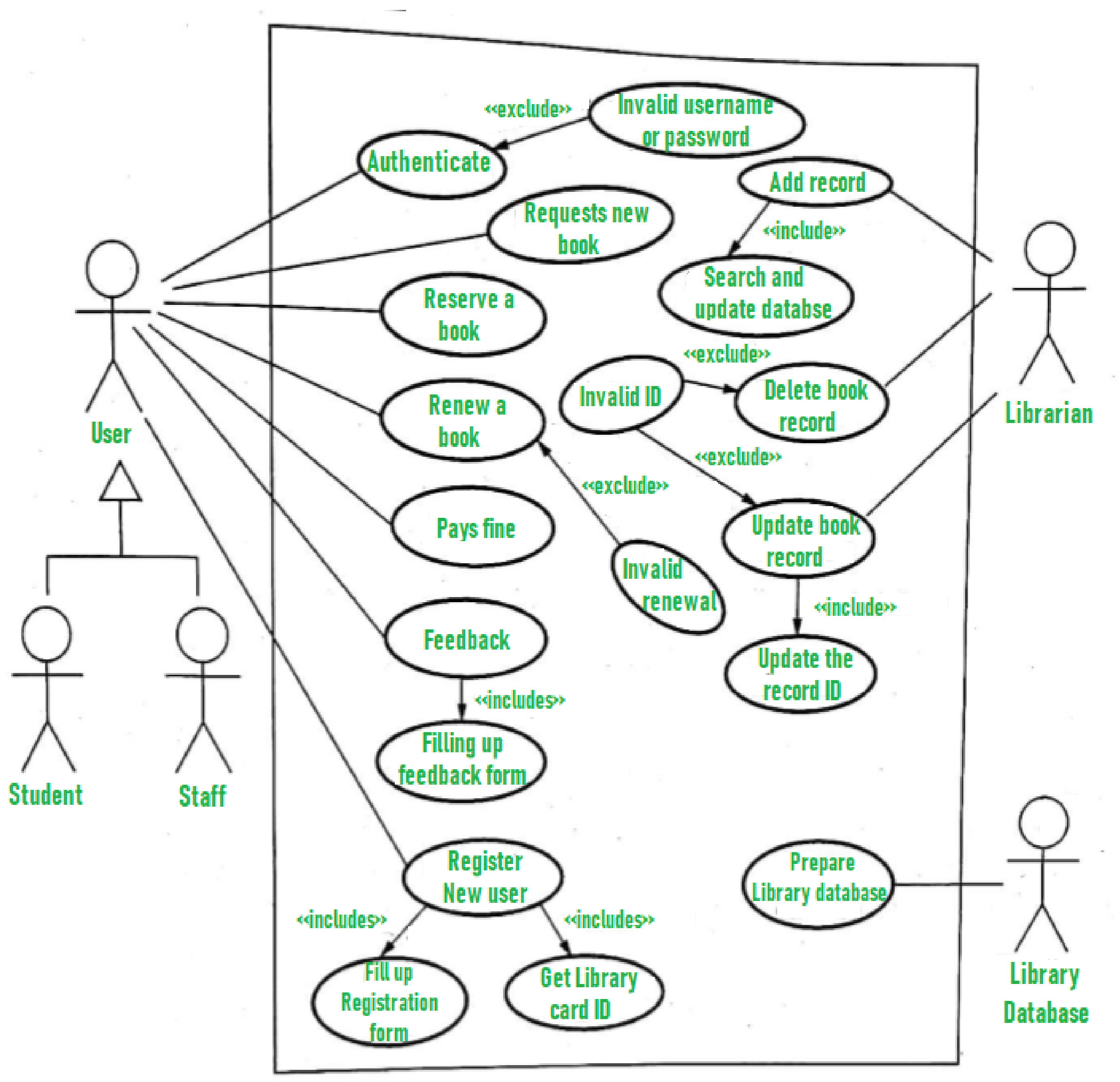
3. After, requesting, the desired book or the requested book is reserved by the user which means no other user can request for that book.
4. Now, the user can renew a book which means the user can get a new due date for the desired book if the user has renewed them.
5. If the user somehow forgets to return the book before the due date, then the user pays a fine. Or if the user forgets to renew the book by the due date, then the book will be overdue and the user pays a fine.
6. User can fill out the feedback form available if they want to.
7. Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing a book or returning the book, or paying fine.
8. Librarian also deletes the record of a particular student if the student leaves the college. If the book no longer exists in the library, then the record of the particular book is also deleted.
9. Updating database is the important role of Librarian.

We have three main actors in our system:

- **Librarian or Admin**: Mainly responsible for adding and modifying books, book items, and users. The Librarian can also issue, reserve, and return book items.
- **User**: All members can search the catalog, as well as check-out, reserve, renew, and return a book.
- **Library Database:** Mainly responsible for prepare library database.

Here are the top use cases of the Library Management System:

- Add/Remove book: To add, remove or modify a book or book item.
- Register new account membership: To add a new member.
- Check-out book: To borrow a book from the library.
- Reserve book: To reserve a book which is not currently available.
- Renew a book: To reborrow an already checked-out book.
- Return a book: To return a book to the library which was issued to a member.



2. flow-oriented modeling

Although data flow-oriented modeling is considered an outdated technique by some software engineers, it is a core modeling activity in structured analysis. Although the data flow diagram (DFD) is not a formal part of UML, it can be used to complement UML diagrams.

2.1 Data Flow Diagram (DFD)

As information moves through software, it is modified by a series of transformations. A data flow diagram is a graphical representation that

depicts information flow and the transforms that are applied as data move from input to output. A few simple guidelines can aid during the derivation of a data flow diagram:

(1) the level 0 data flow diagram should depict the software/system as a single bubble

(2) primary input and output should be carefully noted

(3) refinement should begin by isolating candidate processes, data objects, and stores to be represented at the next level

(4) all arrows and bubbles should be labeled with meaningful names

(5) information flow continuity must be maintained from level to level, and

(6) one bubble at a time should be refined.

The basic form of a data flow diagram, also known as a data flow graph or a bubble chart, is illustrated as following:

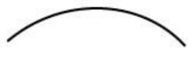
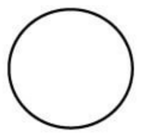
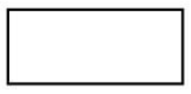
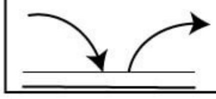
Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.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

Figure 7.3 : show notations of DFD

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:

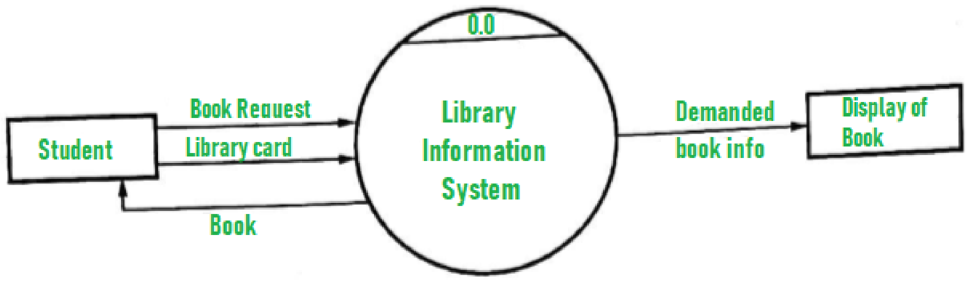
- Book will be the output as the book demanded by the student will be given to them.
- Information of demanded book should be displayed by the library information system that can be used by the student while selecting the book which makes it easier for the student.

Levels of DFD

DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created. Levels of DFD are as follows:

- 0-level DFD.
- 1-level DFD.
- 2-level DFD.

<u>Level 0 DFD</u>

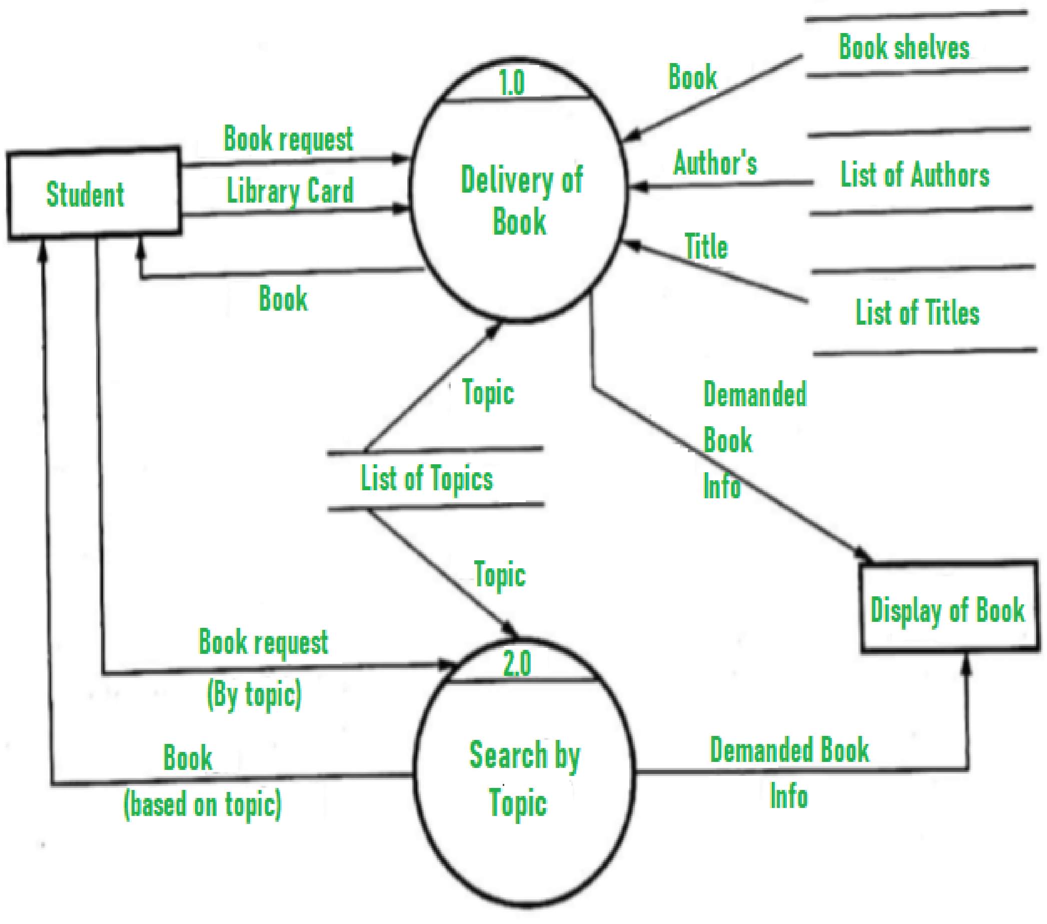


Level 1 DFD

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

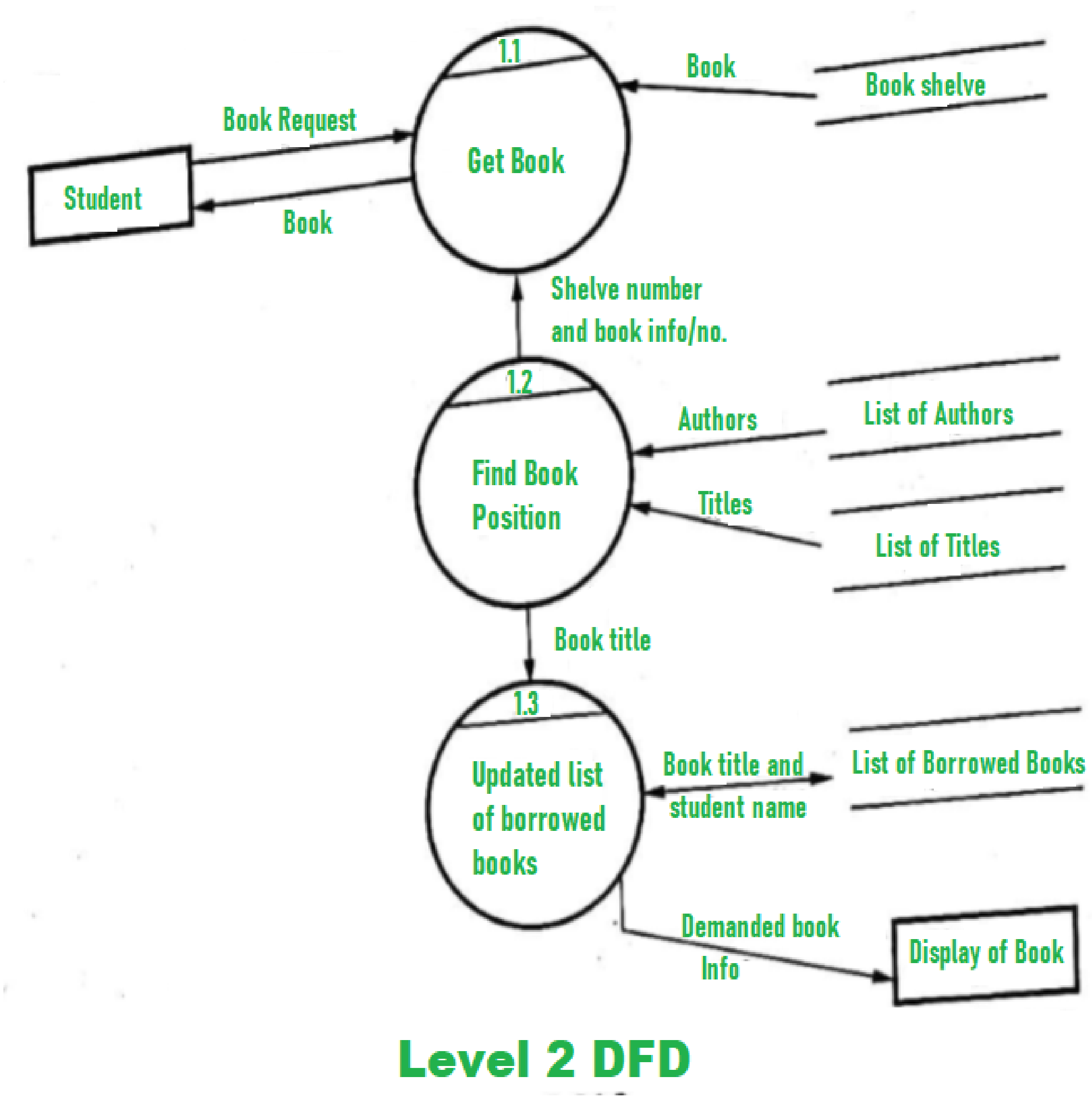
- Book delivery.
- Search by topic.

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.



Level 1 DFD

Level 2 DFD



2.2 THE DATA DICTIONARY

Data Dictionary is the major component in the structured analysis model of the system. It lists all the data items appearing in DFD. A data dictionary in Software Engineering means a file or a set of files that includes a database's metadata. The data dictionary is different from the database. It does not contain tables. It only contains information about the tables and resources if any data table is inherited from some where. The data dictionary is created when we create a new database.

<u>**EXAMPLE:**</u> To illustrate the use of the data dictionary, we return to the DFD for the ***library management system***. The data dictionary entry for the **Librarian (Admin)** is as follows:

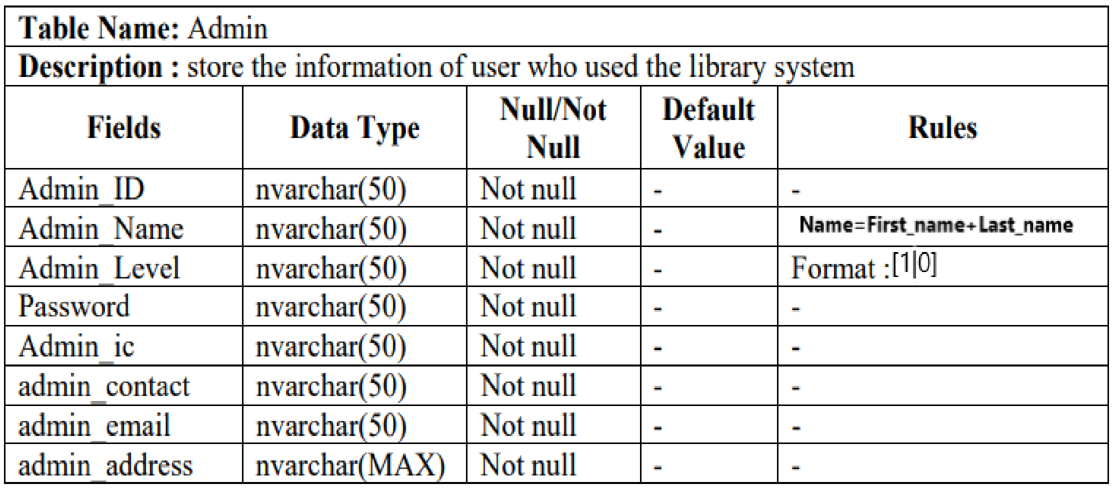
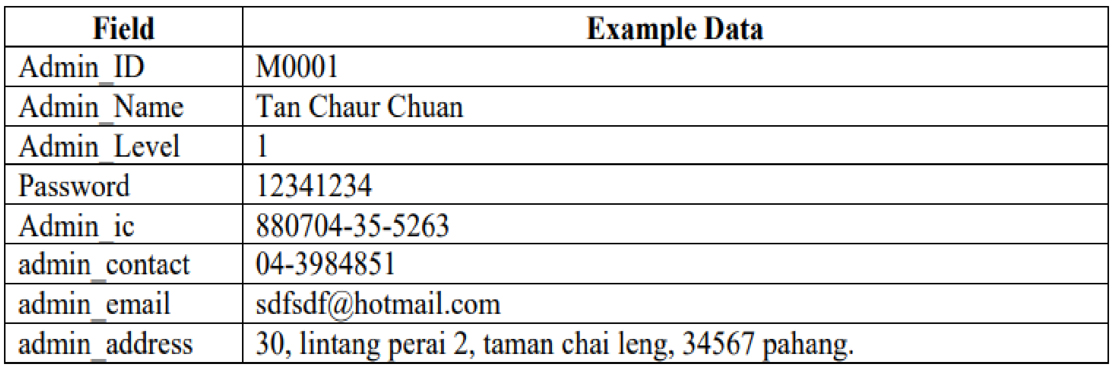


Table Name: Admin				
Description : store the information of user who used the library system				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
Admin_ID	nvarchar(50)	Not null	-	-
Admin_Name	nvarchar(50)	Not null	-	Name=First_name+Last_name
Admin_Level	nvarchar(50)	Not null	-	Format :[1|0]
Password	nvarchar(50)	Not null	-	-
Admin_ic	nvarchar(50)	Not null	-	-
admin_contact	nvarchar(50)	Not null	-	-
admin_email	nvarchar(50)	Not null	-	-
admin_address	nvarchar(MAX)	Not null	-	-



Field	Example Data
Admin_ID	M0001
Admin_Name	Tan Chaur Chuan
Admin_Level	1
Password	12341234
Admin_ic	880704-35-5263
admin_contact	04-3984851
admin_email	sdfsdf@hotmail.com
admin_address	30, lintang perai 2, taman chai leng, 34567 pahang.

The notation used to develop a content description is noted in the following table:

Composition	=	Is composed of
Sequence	+	Represents AND
Selection	[|]	Represents OR
Repetition	$\{\}^n$	Represents n repetitions or repetition for n times
Parentheses	()	Optional data that may or may not be present
Comment	*...*	Delimits a comment or commented information

<u>Importance of data dictionary</u>:

- It provides developers to use different terms to refer to the same data.
- It helps to understand the database for new joiners or new database users.
- All the information is assembled in one place as databases are huge and have a number of tables.