

What is Structured Analysis?

Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.

It has following attributes –

- It is graphic which specifies the presentation of application.
- It divides the processes so that it gives a clear picture of system flow.
- It is logical rather than physical i.e., the elements of system do not depend on vendor or hardware.
- It is an approach that works from high-level overviews to lower-level details.

Structured Analysis Tools

During Structured Analysis, various tools and techniques are used for system development. They are –

- Data Flow Diagrams
- Data Dictionary
- Decision Trees
- Decision Tables
- Structured English
- Pseudocode

LOGICAL DATA MODEL VERSUS PHYSICAL DATA MODEL

LOGICAL DATA MODEL

Model that describes the data as much as possible, without regard to how they will be physical implemented in the database

Defines the data elements and their relationships

Data Architects and business analysts create logical data model

The objective of logical data model is to develop a technical map of rules and data structures

Simpler than the physical data model

PHYSICAL DATA MODEL

Model that represents how the actual database is built

Allows developing the actual database

Database Administrators and developers create physical data model

The objective of physical model is to implement the actual database

Complex than the logical data model

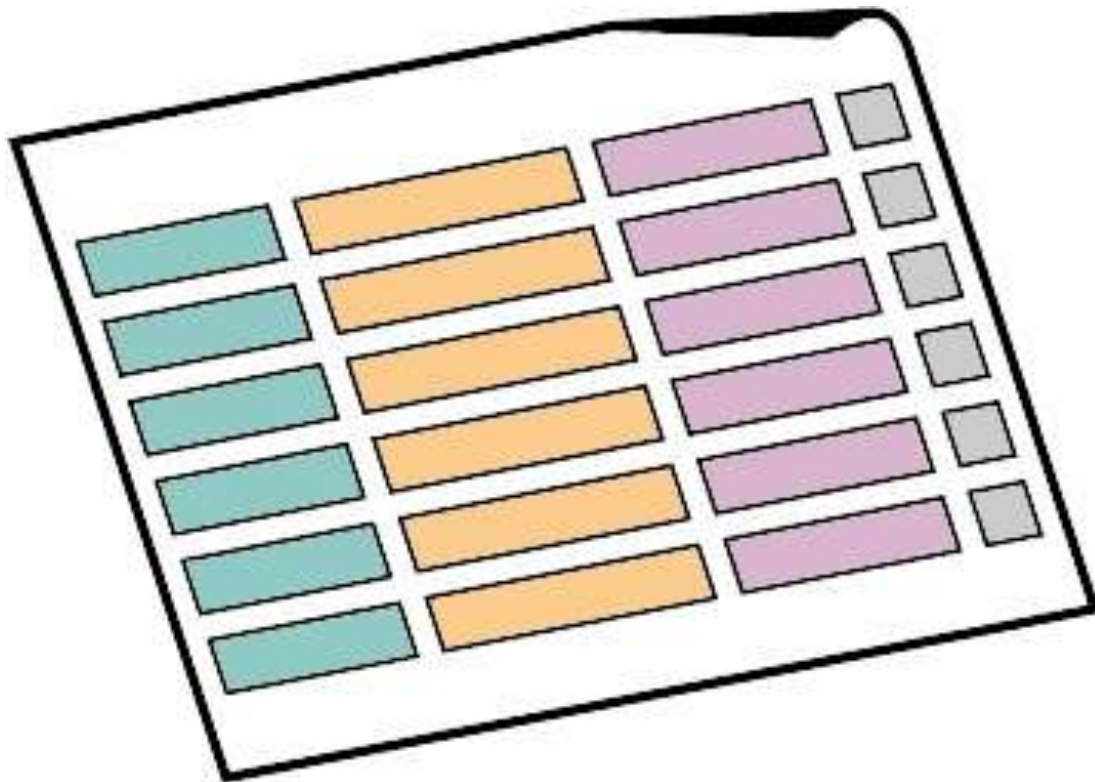
Visit www.PEDIAA.com

What is a Logical Data Model

After obtaining the business requirements, the logical model describes the data to a great extent, but it does not take part in implementing the database. In other words, it is the basis for developing the physical model. It includes all the entities, relationships between them and the attributes of the entities. Generally, the logical data model can be designed and developed independently from the DBMS. Moreover, it is possible to apply normalization. Usually, it is up to the third normal form (3NF).

What is a Physical Data Model

A physical data model is a model that helps to implement the database. In other words, it represents the way of building the database. Moreover, the physical data model gives an abstraction of the database and helps to generate the schema.



It helps to model the database column keys, constraints, indexes, triggers and other RDBMS features. It also converts the entities into tables, relationships into foreign keys and attributes into columns. Furthermore, it is possible to modify the physical data model based on physical constraints or requirements.

The Context Diagram

The Context Diagram shows the system under consideration as a single high-level process and then shows the relationship that the system has with other external entities (systems, organizational groups, external data stores, etc.).

Another name for a Context Diagram is a **Context-Level Data-Flow Diagram** or a Level-0 Data Flow Diagram. Since a Context Diagram is a specialized version of **Data-Flow Diagram**, understanding a bit about Data-Flow Diagrams can be helpful.

A **Data-Flow Diagram (DFD)** is a graphical visualization of the movement of data through an information system. DFDs are one of the three essential components of the structured-systems analysis and design method (SSADM). A DFD is process centric and depicts 4 main components.

- Processes (circle)
- External Entities (rectangle)
- Data Stores (two horizontal, parallel lines or sometimes an ellipse)
- Data Flows (curved or straight line with arrowhead indicating flow direction)

Each DFD may show a number of processes with data flowing into and out of each process. If there is a need to show more detail within a particular process, the process is decomposed into a number of smaller processes in a lower level DFD. In this way, the Context Diagram or Context-Level DFD is labeled a “Level-0 DFD” while the next level of decomposition is labeled a “Level-1 DFD”, the next is labeled a “Level-2 DFD”, and so on.

Context Diagrams and Data-Flow Diagrams were created for systems analysis and design. But like many analysis tools they have been leveraged for other purposes. For example, they can also be leveraged to capture and communicate the interactions and flow of data between business processes. So, they don't have to be restricted to systems analysis.

A sample Context Diagram is shown here.



A Context Diagram (and a DFD for that matter) provides no information about the timing, sequencing, or synchronization of processes such as which processes occur in sequence or in parallel. Therefore it should not be confused with a flowchart or process flow which can show these things.

Some of the benefits of a Context Diagram are:

- Shows the scope and boundaries of a system at a glance including the other systems that interface with it
- No technical knowledge is assumed or required to understand the diagram
- Easy to draw and amend due to its limited notation
- Easy to expand by adding different levels of DFDs
- Can benefit a wide audience including stakeholders, business analyst, data analysts, developers

Data Flow Diagram

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD** - This type of DFD concentrates on the system process, and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

DFD Components

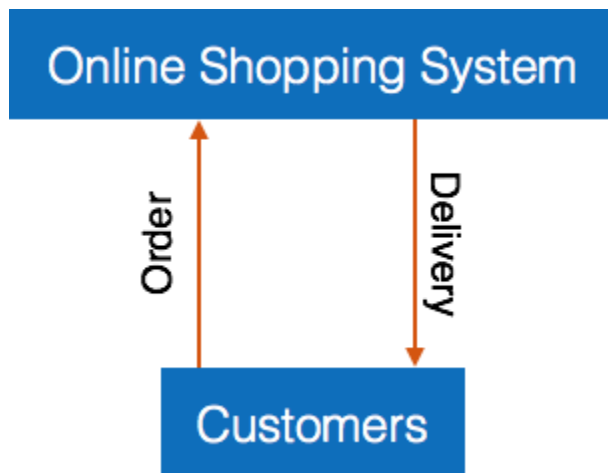
DFD can represent Source, destination, storage and flow of data using the following set of components -



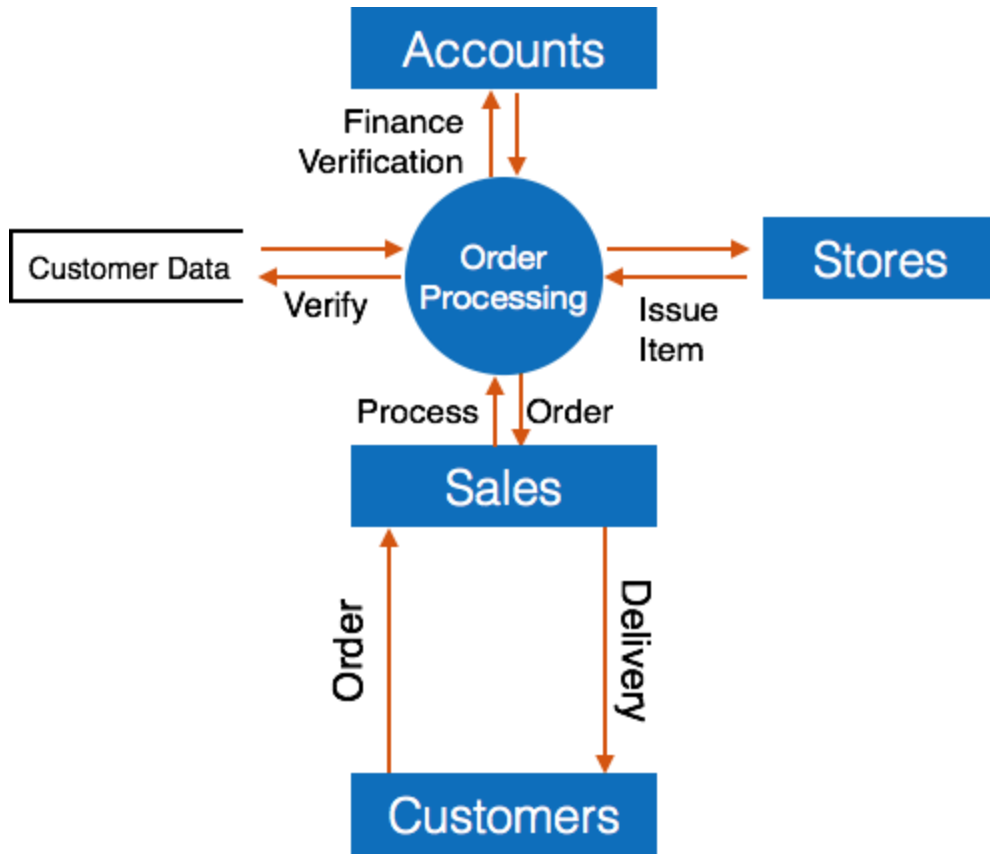
- **Entities** - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.
- **Process** - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- **Data Storage** - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- **Data Flow** - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD

- **Level 0** - Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.



- **Level 1** - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.



- **Level 2** - At this level, DFD shows how data flows inside the modules mentioned in Level 1.

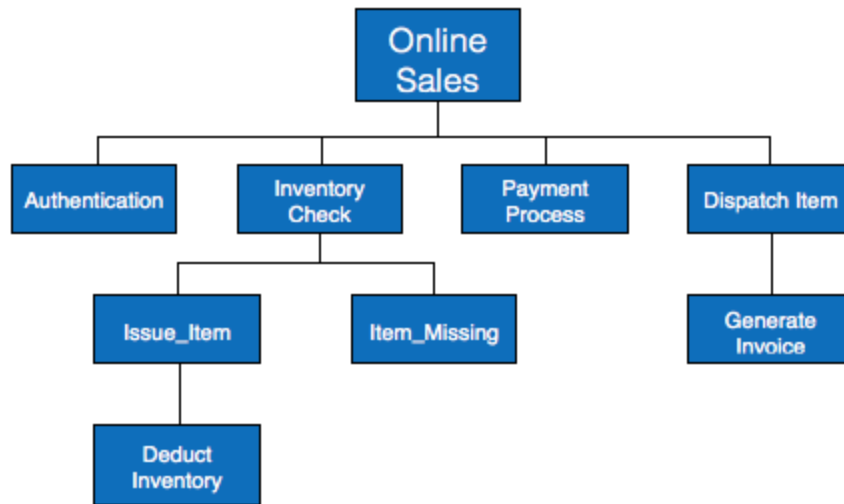
Higher level DFDs can be transformed into more specific lower level DFDs with deeper level of understanding unless the desired level of specification is achieved.

HIPO Diagram

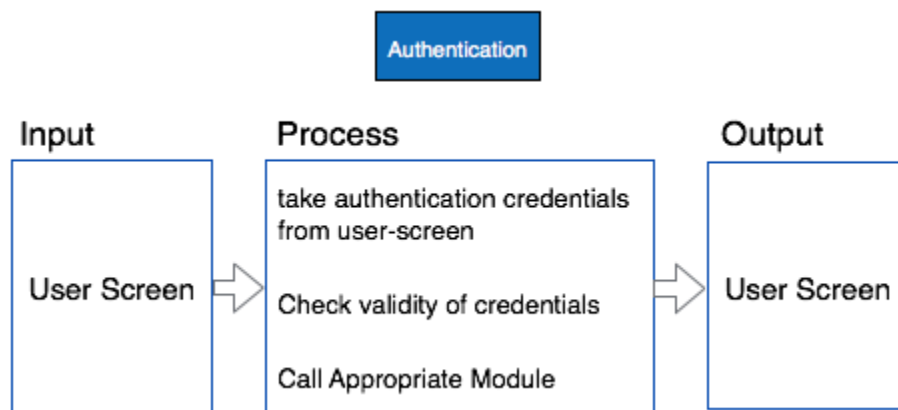
HIPO (Hierarchical Input Process Output) diagram is a combination of two organized method to analyze the system and provide the means of documentation. HIPO model was developed by IBM in year 1970.

HIPO diagram represents the hierarchy of modules in the software system. Analyst uses HIPO diagram in order to obtain high-level view of system functions. It decomposes functions into sub-functions in a hierarchical manner. It depicts the functions performed by system.

HIPO diagrams are good for documentation purpose. Their graphical representation makes it easier for designers and managers to get the pictorial idea of the system structure.



In contrast to IPO (Input Process Output) diagram, which depicts the flow of control and data in a module, HIPO does not provide any information about data flow or control flow.



Example

Both parts of HIPO diagram, Hierarchical presentation and IPO Chart are used for structure design of software program as well as documentation of the same.

Data Dictionary

Data dictionary is the centralized collection of information about data. It stores meaning and origin of data, its relationship with other data, data format for usage etc. Data dictionary has rigorous definitions of all names in order to facilitate user and software designers.

Data dictionary is often referenced as meta-data (data about data) repository. It is created along with DFD (Data Flow Diagram) model of software program and is expected to be updated whenever DFD is changed or updated.

Requirement of Data Dictionary

The data is referenced via data dictionary while designing and implementing software. Data dictionary removes any chances of ambiguity. It helps keeping work of programmers and designers synchronized while using same object reference everywhere in the program.

Data dictionary provides a way of documentation for the complete database system in one place. Validation of DFD is carried out using data dictionary.

Contents

Data dictionary should contain information about the following

- Data Flow
- Data Structure
- Data Elements
- Data Stores
- Data Processing

Data Flow is described by means of DFDs as studied earlier and represented in algebraic form as described.

=	Composed of
{ }	Repetition
()	Optional
+	And
[/]	Or

Example

Address = House No + (Street / Area) + City + State

Course ID = Course Number + Course Name + Course Level + Course Grades

Data Elements

Data elements consist of Name and descriptions of Data and Control Items, Internal or External data stores etc. with the following details:

- Primary Name
- Secondary Name (Alias)

- Use-case (How and where to use)
- Content Description (Notation etc.)
- Supplementary Information (preset values, constraints etc.)

Data Store

It stores the information from where the data enters into the system and exists out of the system. The Data Store may include -

- **Files**
 - Internal to software.
 - External to software but on the same machine.
 - External to software and system, located on different machine.
- **Tables**
 - Naming convention
 - Indexing property

Data Processing

There are two types of Data Processing:

- **Logical:** As user sees it
- **Physical:** As software sees it

Pseudo-Code

Pseudo code is written more close to programming language. It may be considered as augmented programming language, full of comments and descriptions.

Pseudo code avoids variable declaration but they are written using some actual programming language's constructs, like C, Fortran, Pascal etc.

Pseudo code contains more programming details than Structured English. It provides a method to perform the task, as if a computer is executing the code.

Example

Program to print Fibonacci up to n numbers.

```
void function Fibonacci
Get value of n;
Set value of a to 1;
Set value of b to 1;
Initialize I to 0
for (i=0; i< n; i++)
{
```

```

if a greater than b
{
    Increase b by a;
    Print b;
}
else if b greater than a
{
    increase a by b;
    print a;
}
}

```

Decision Tables

A Decision table represents conditions and the respective actions to be taken to address them, in a structured tabular format.

It is a powerful tool to debug and prevent errors. It helps group similar information into a single table and then by combining tables it delivers easy and convenient decision-making.

Creating Decision Table

To create the decision table, the developer must follow basic four steps:

- Identify all possible conditions to be addressed
- Determine actions for all identified conditions
- Create Maximum possible rules
- Define action for each rule

Decision Tables should be verified by end-users and can lately be simplified by eliminating duplicate rules and actions.

Example

Let us take a simple example of day-to-day problem with our Internet connectivity. We begin by identifying all problems that can arise while starting the internet and their respective possible solutions.

We list all possible problems under column conditions and the prospective actions under column Actions.

	Conditions/Actions	Rules
Conditions	Shows Connected	N N N N Y Y Y Y

	Ping is Working	N	N	Y	Y	N	N	Y	Y
	Opens Website	Y	N	Y	N	Y	N	Y	N
Actions	Check network cable	X							
	Check internet router	X				X	X	X	
	Restart Web Browser								X
	Contact Service provider		X	X	X	X	X	X	
	Do no action								

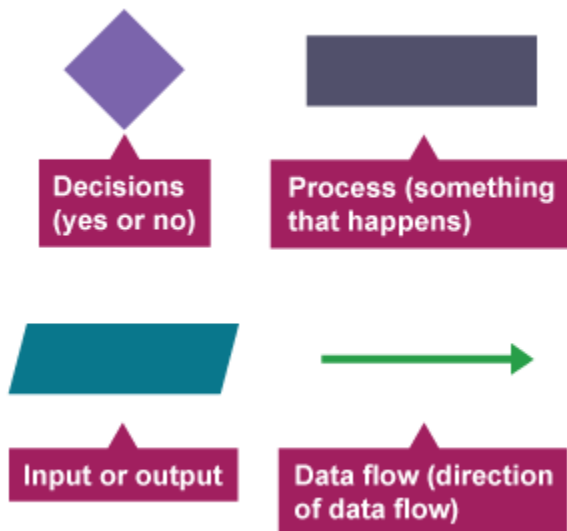
Table : Decision Table – In-house Internet Troubleshooting

What is a system flowchart?

System flowcharts are a way of displaying how data flows in a system and how decisions are made to control events.

To illustrate this, symbols are used. They are connected together to show what happens to data and where it goes.

Note that system flow charts are very similar to data flow charts. Data flow charts do not include decisions, they just show the path that data takes, where it is held, processed, and then output.



Input Design

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well designed input forms and screens have following properties –

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.
- All these objectives are obtained using the knowledge of basic design principles regarding –
 - What are the inputs needed for the system?
 - How end users respond to different elements of forms and screens.

Objectives for Input Design

The objectives of input design are –

- To design data entry and input procedures
- To reduce input volume
- To design source documents for data capture or devise other data capture methods
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

Data Input Methods

It is important to design appropriate data input methods to prevent errors while entering data. These methods depend on whether the data is entered by customers in forms manually and later entered by data entry operators, or data is directly entered by users on the PCs.

A system should prevent user from making mistakes by –

- Clear form design by leaving enough space for writing legibly.
- Clear instructions to fill form.
- Clear form design.
- Reducing key strokes.
- Immediate error feedback.

Some of the popular data input methods are –

- Batch input method (Offline data input method)

- Online data input method
- Computer readable forms
- Interactive data input

Input Integrity Controls

Input integrity controls include a number of methods to eliminate common input errors by end-users. They also include checks on the value of individual fields; both for format and the completeness of all inputs.

Audit trails for data entry and other system operations are created using transaction logs which gives a record of all changes introduced in the database to provide security and means of recovery in case of any failure.

Output Design

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

Objectives of Output Design

The objectives of input design are –

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end users requirements.
- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

Let us now go through various types of outputs –

External Outputs

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

Internal outputs

Internal outputs are present inside the system, and used by end-users and managers. They support the management in decision making and reporting.

There are three types of reports produced by management information –

- **Detailed Reports** – They contain present information which has almost no filtering or restriction generated to assist management planning and control.

- **Summary Reports** – They contain trends and potential problems which are categorized and summarized that are generated for managers who do not want details.
- **Exception Reports** – They contain exceptions, filtered data to some condition or standard before presenting it to the manager, as information.

Output Integrity Controls

Output integrity controls include routing codes to identify the receiving system, and verification messages to confirm successful receipt of messages that are handled by network protocol.

Printed or screen-format reports should include a date/time for report printing and the data. Multipage reports contain report title or description, and pagination. Pre-printed forms usually include a version number and effective date.

Forms Design

Both forms and reports are the product of input and output design and are business document consisting of specified data. The main difference is that forms provide fields for data input but reports are purely used for reading. For example, order forms, employment and credit application, etc.

- During form designing, the designers should know –
 - who will use them
 - where would they be delivered
 - the purpose of the form or report
- During form design, automated design tools enhance the developer's ability to prototype forms and reports and present them to end users for evaluation.

Objectives of Good Form Design

A good form design is necessary to ensure the following –

- To keep the screen simple by giving proper sequence, information, and clear captions.
- To meet the intended purpose by using appropriate forms.
- To ensure the completion of form with accuracy.
- To keep the forms attractive by using icons, inverse video, or blinking cursors etc.
- To facilitate navigation.

Types of Forms

Flat Forms

- It is a single copy form prepared manually or by a machine and printed on a paper. For additional copies of the original, carbon papers are inserted between copies.
- It is a simplest and inexpensive form to design, print, and reproduce, which uses less volume.

Unit Set/Snap out Forms

- These are papers with one-time carbons interleaved into unit sets for either handwritten or machine use.
- Carbons may be either blue or black, standard grade medium intensity. Generally, blue carbons are best for handwritten forms while black carbons are best for machine use.

Continuous strip/Fanfold Forms

- These are multiple unit forms joined in a continuous strip with perforations between each pair of forms.
- It is a less expensive method for large volume use.

No Carbon Required (NCR) Paper

- They use carbonless papers which have two chemical coatings (capsules), one on the face and the other on the back of a sheet of paper.
- When pressure is applied, the two capsules interact and create an image.