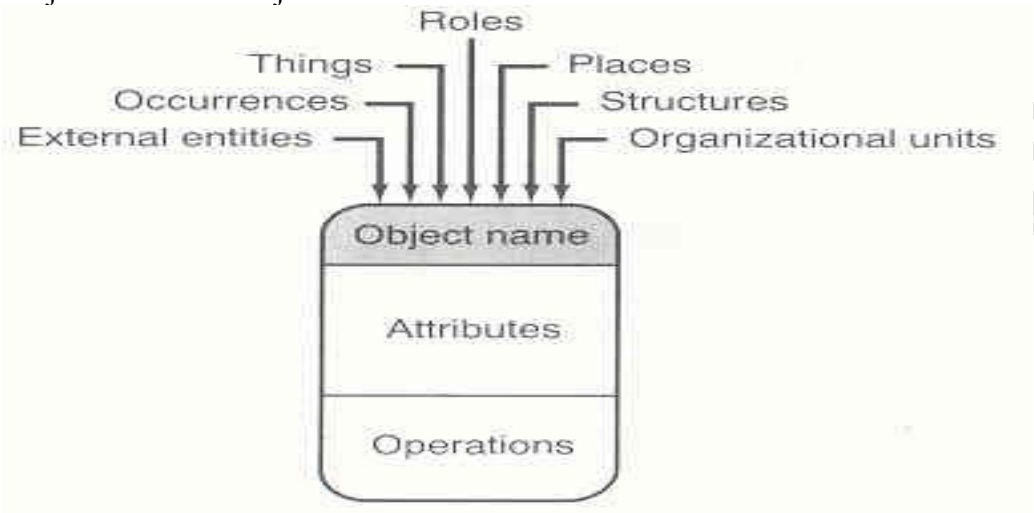
OBJECT ORIENTED CONCEPTS

To understand the object-oriented point of view, consider an example of real world object-the thing you are sitting in right now – a chair. Chair is a member of a much larger class of objects that we call furniture. A set of generic attributes can be associated with every object in the class furniture.

Formal definition of "object-oriented "will prove more meaningful.

Object-oriented = object + classification + inheritance + communication



IDENTIFYING OBJECTS

Identifying objects by examining the problem statement or performing a "grammatical parse" on the processing narrative for the system to be built. Objects are determined by underlining each noun or noun clause and entering it in a simple table. Synonyms should be noted.

- It is part of a problem space. Objects manifest themselves in one of the ways.
- External entities that produce or consume information to be used by a computer based system.
- Things that are the part of the information domain for the problem.
- Occurrence or events that occur within the context of system operation.
- Roles played by people who interact with the system.
- Organizational unit those are relevant to an application.
- Places that establish the context of the problem and the overall function of the system
- Structures that define a class of object or, in the extreme, related classes of objects.

Extracting the nouns, we can propose a number of potential objects:

Potential object/class	General classification
Homeowner	Role or external entity
Sensor	External entity
Control panel	External entity
Installation	Occurrence
System	Thing
Number, type	Not objects, attributes of sensor

Master password	Thing
Telephone number	Thing
Sensor event	Occurrence
Audible alarm	External entity
Monitoring service	Organization unit or external entity

Coad & Yourdon suggest six-selection characteristic that should be used as an analyst Considers each potential object for inclusion in the analysis model:

1. **Retained information**. The potential object will be useful during analysis only if information about it must be remembered so that the system can function.
2. **Needed service**. The potential object must have a set of identifiable operation that can change the value of its attributes in some way.
3. **Multiple attributes**. During requirement analysis, the focus should be on "major" information; an object with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of an object during the analysis activity.
4. **Common attributes**. A set of attributes can be defined for the potential object and these attributes apply to all occurrences of the object.
5. **Common operations**. A set of operations can be defined for the potential object And these operations apply to all occurrence of the object.
6. **Essential requirement**. External entities that appear in the problem space and the produce or consume information that is essential to the operation of any solution for the system will almost always be defined as objects in the requirement model.

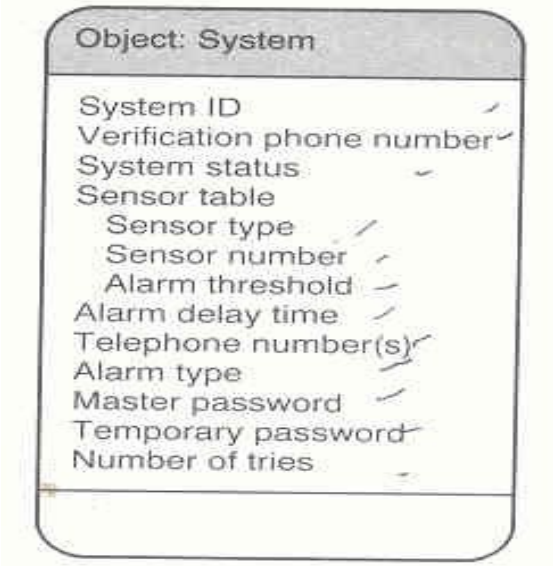
Potential object/class	Characteristic Number that Applies
Homeowner	Reject: 1,2 fail even though 6 applies
Sensor	Accepted: all apply
Control panel	Accepted: all apply
Installation	Rejected
System	Accepted: all apply
Number, type	Rejected: 3 fails, attributes of sensor
Master password	Rejected: 3 fails
Telephone number	Rejected: 3 fails
Sensor event	Accepted: all apply
Audible alarm	Accepted: 2, 3, 4, 5, 6 apply
Monitoring service	Rejected: 1, 2 fail even though 6 applies

SPECIFYING ATTRIBUTES

Attributes describe an object that has been selected for inclusion in the analysis model. In essence, it is the attributes that define the object-that clarify what is meant by the object in the context of the problem space.

To develop a meaningful set of attributes for an object, the analyst can again study the processing narrative (or statement of scope) for the problem and select those things that reasonably "belong" to the object.

Consider the system object defined for safe Home. Homeowner can configure the security system to reflect sensor information, alarm response information, activation/ deactivation information, identification information, and so forth.
Sensor information = sensor type + sensor number + alarm threshold
Alarm response information = delay time + telephone number + alarm type
Activation/deactivation information = master password + number of allowable tries + temporary password
Identification = system ID + verification phone number + system status



DEFINING OPERATIONS

An operation changes an object in some way. More specifically, it changes one or more attribute values that are contained within the object. Therefore, an operation must have "knowledge" of the nature of the objects attributes and must be implemented in a manner that enables it to manipulate the data structures that have been derived from the attributes. Although many different types of operations exit, they can generally be divided into three broad categories:
(1) Operations that manipulate data in some way (e.g. adding, deleting, reformatting, selecting);
(2) Operations that perform a computation;
(3) Operations that monitor an object for the occurrence of a controlling event.
As a "first cut " at deriving a set of operations for the objects of the analysis model, the analyst can again study the processing narrative for the problem and select those operation that reasonably "belong" to the object. To accomplish this, the grammatical parse is again studied and verbs are isolated. Some of these verbs will be legitimate operations and can be easily connected to a specific object.

- That an assign operation is relevant for the sensor object.
- That a program operation will be applied to the system object.
- That arm and disarm are operations that apply to system (also that system status may ultimately be defined, using data dictionary notation, as system status = [armed/disarmed]).

Upon further investigation, it is likely that the operation program will be divided into a number of more specific sub operation required to configure the system.

INTEROBJECT COMMUNICATION

Defining objects within the context of an analysis model may be enough to lay a foundation for design. But something else must be added (either during analysis or during design) so that the system can be built – a mechanism for communication must be established between objects. This mechanism is called a message

The interaction between messages is illustrated schematically in fig. An operation within sender object generates a message of the form.

Message [destination, operation, and parameter]

Where destination defines the receiver object that is stimulated by the message, operation refers to the operation that is to receive the message, and parameter provides information that is required for the operation to be successful.

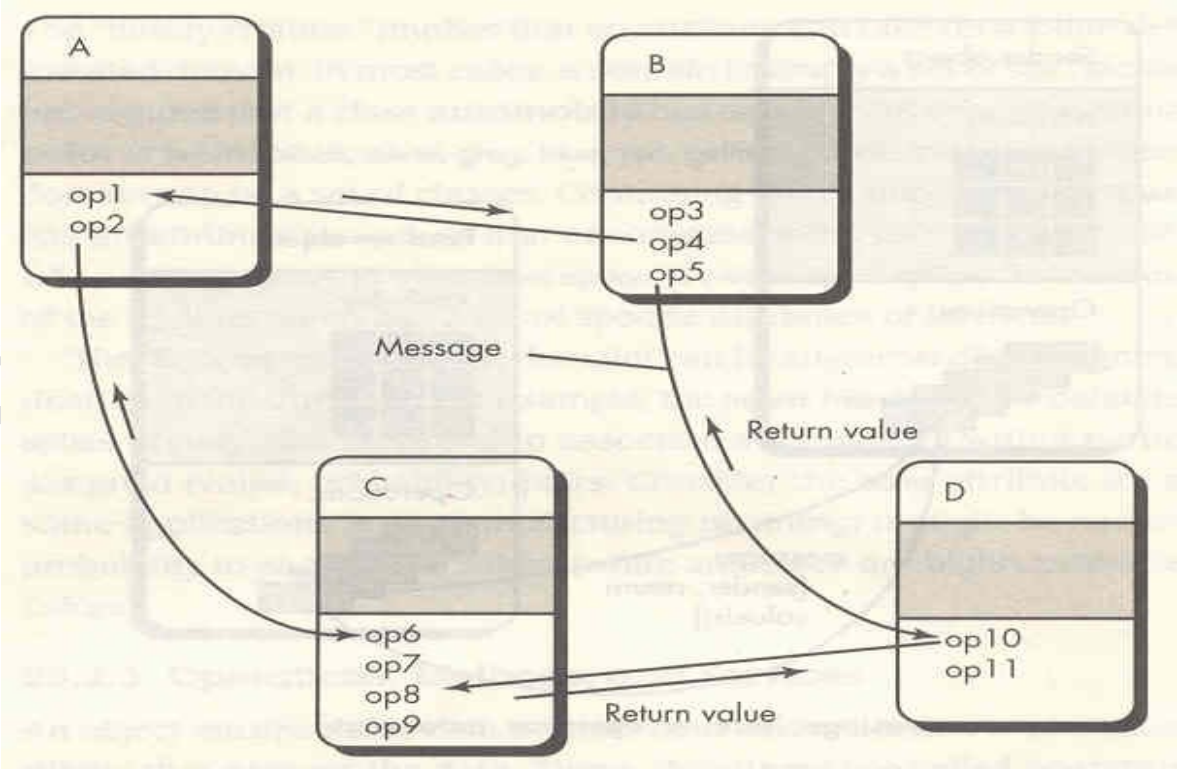
As an example of message passing within OO system , consider the object show in fig. four objects A,B,C,D communicate with one another by passing message. For example if object B requires processing associated with operation of object D, it would send d a message of the form

Message [D, op08, (data)]

As part of the execution of op10, object D may send a message to the object c of the form

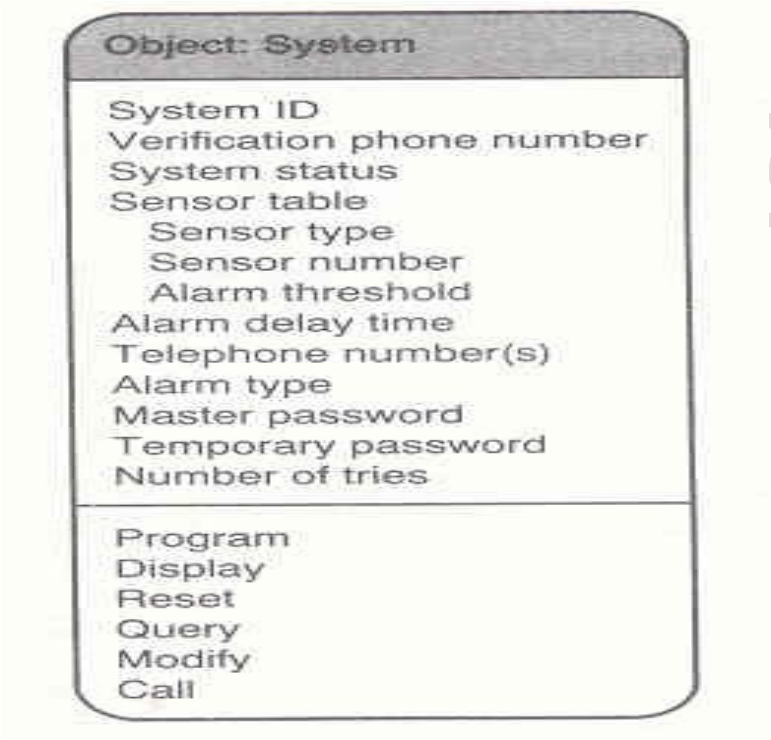
Message [C, op08, (data)]

Then c finds op08, performs it, & sends an appropriate return value to D. Operation op10 completes & sends a return value to B.



FINALIZING THE OBJECT DEFINITION

The definition of the operation is the last step in completing the specification of an object. The generic life history of an object can be defined by recognizing that the object must be created, modified, manipulated, or read in other ways, and possibly deleted. For the system object, this can be expended to reflect known activities that occur during its life. Some of the operation can be ascertained from likely communication between objects. For example, sensor event will send a message to system to display the event location and number; control panel will send a reset message to update system status; the audible alarm will send a query message; the control panel will send a modified message to change one or more attributes without reconfiguration the entire system object; sensor event will also send a message to call the phone no contained in the object. Other message can be considered and the operation derived. The resulting object definition is show in the fig.

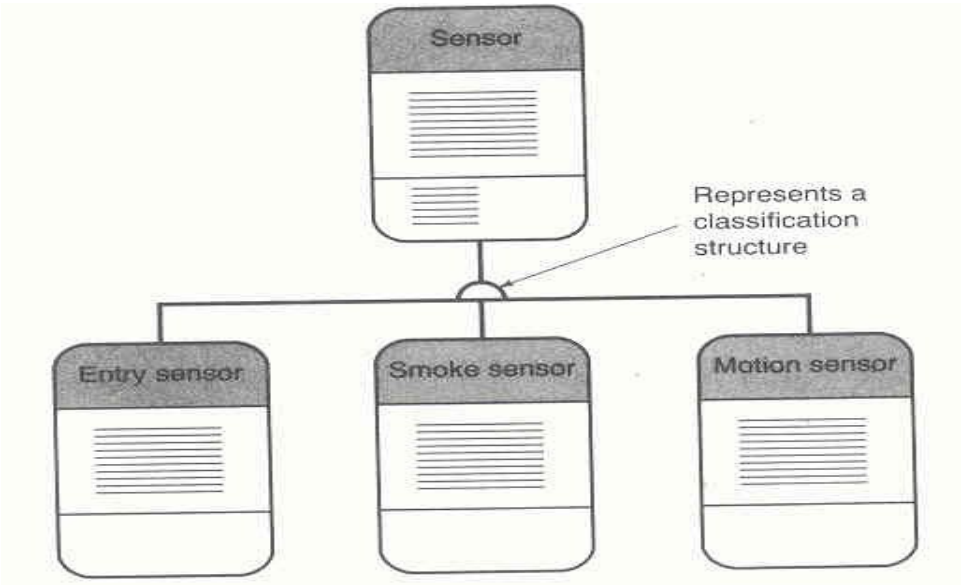


OBJECT ORIENTED ANALYSIS MODELLING: -

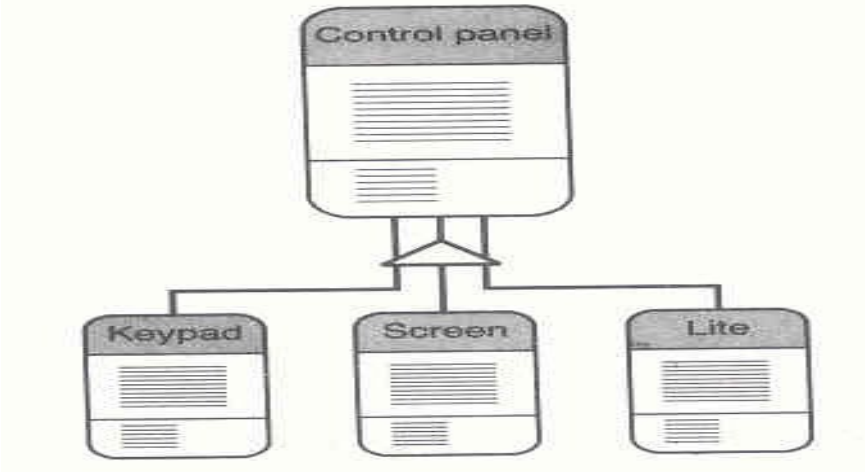
Classification and assemble structure: -

Once objects have been identified, the analyst begins to focus on classification structure. Each object is "Considered as a generation, then as a specialization" i.e., the instances of an object are defined and named.

Consider the sensor object defined for safe home and shown if fig (a)



In other cases an object represented in the initial model might actually be composed of a number of component parts that could themselves be defined as objects. Structures of this type are called assembly structures and are defined using a notation represented in fig (b).

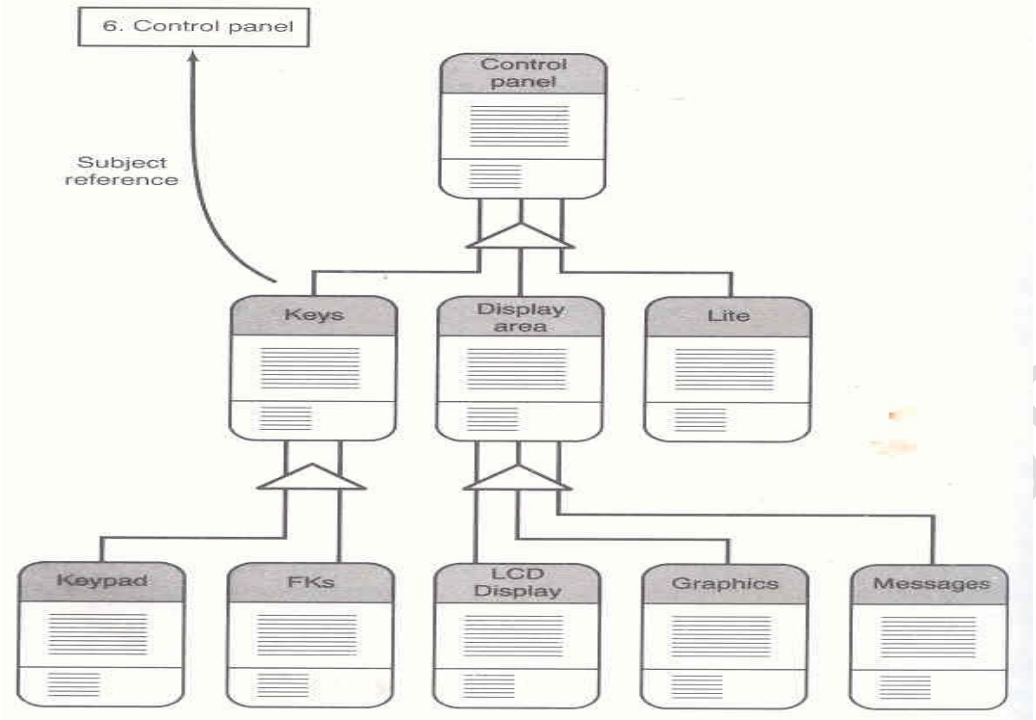


The triangle implies an assembly relationship. It should be noted that the connecting lines may be augmented with additional symbols that are adapted from the E-R modeling notation.

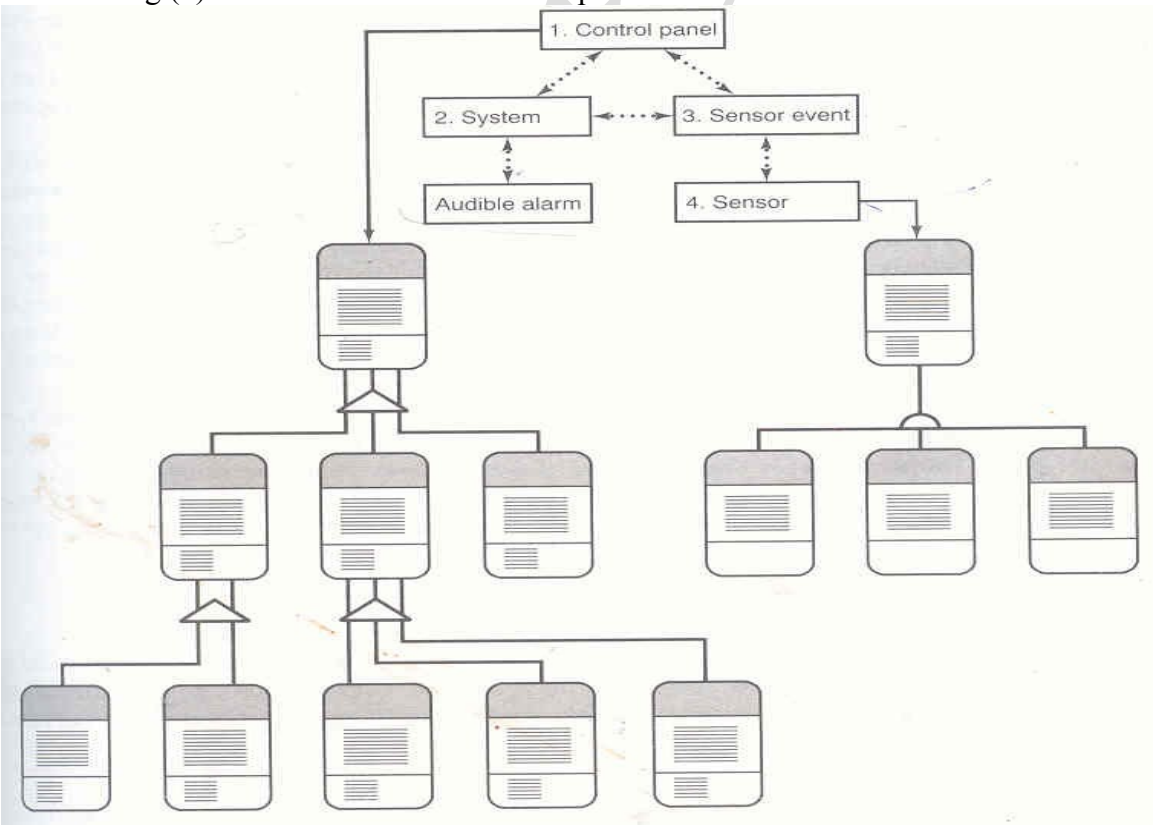
DEFINING SUBJECTS: -

A subject is nothing more than a reference or pointer to more detail in the analysis model. For example, assume that the control panel for safe home containing multiple display areas, a sophisticated key arrangement, and other features. It might be modified as an assemble structure.

If the overall requirements model contains dozens of these structures, it would be difficult to absorb the entire representation at one time. By defining a subject reference as shown in fig(c) the entire structure can be referenced by a single icon (the rectangle). Subject references are generally created for any structure that has more than five or six objects.



At the most abstract level, the OOA model would contain only subject references as shown in fig (d). Each reference would be expanded into a structure.



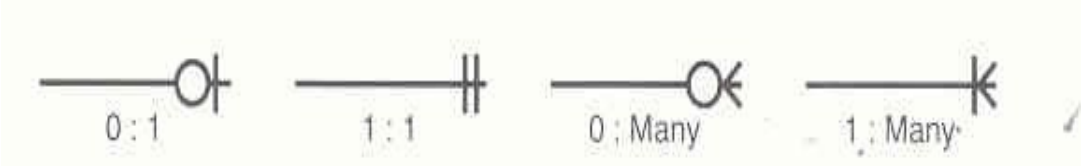
The dotted double-headed arrows represent communication path between objects.

INSTANCE CONNECTIONS AND MESSAGE PATHS: -

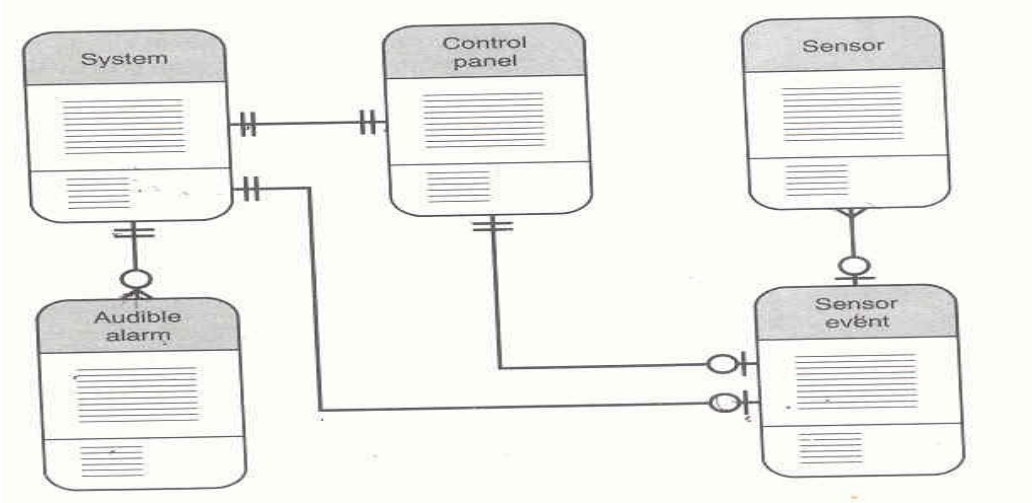
An instance connection is a modeling notation that defines a specific relationship between instances of an object.

For example, Fig (e) represents the first step in creation of instance connections for safe home objects. The simple lines between objects indicate that some important relationship does exist. For example, sensors can generate a sensor event and sensor events can be recognized by the system. Once the lines have been established, each end is evaluated to determine whether a single (1:1) connections exists or multiple connections (1: many) exists. For a 1:1 connection, the line is annotated with a single bar. A 1: Many connections are annotated with a three-pronged fork symbol as shown in fig (f). Finally, we add another vertical bar if the connection is mandatory & a circle to represent an optional connection .To illustrate at least one and possibly many sensors can lead to a sensor event for safe home yet in any particular situation, either 0 or 1 sensor events will occur. This type of evaluation continues for each instance connection.

Not only are the relationships between objects identified, but all important message paths are identified. In our discussion of figure {d}, we made reference to the dotted arrows that Connected subject symbols. These are message paths [sometimes referred to as message connections] each arrows implies the interchange of messages among objects in the model. As we have already noted, messages will move along each instance connection path; therefore message connections are derived directly from instance connections.



OOA model & provides an excellent foundation for the creation of a software requirement specification. Each object represented graphically must also be described using a text-oriented format.



OOA AND PROTOTYPING: -

The use of OOA can lead to extremely effective prototyping and "evaluating software engineering" techniques. The objects that are specified and ultimately implemented for a current project can be cataloged in a library. Objects are inherently reusable, and over time the library of reusable objects will grow. When OOA is applied to new projects, the analyst can work to specify the system using existing objects contained in the object library rather than inventing new ones(this practice is common place in hardware design, system engineers specify existing integrated circuits, rather than requiring custom design). For example, the sensor and sensor event objects described for safe home could be used without change in a variety of process control and monitoring applications.

By using existing objects during analysis specification time is reduced substantially and a rapid prototype of the specified system can be created and reviewed by the customer.

DATA MODELING:

Data modeling or information modeling focuses solely on data. Representing a "data network" that exists for a given system. Data modeling is useful for application in which data and the relationship that governs data is complex. Unlike the structured analysis approach, data modeling consider data independently of the processing that transforms the data.

Both used the term "object "but its definition is much more limited in a data modeling contest. Both describe relationship between objects, but data modeling does not censer it self with how these relationships are achieved.

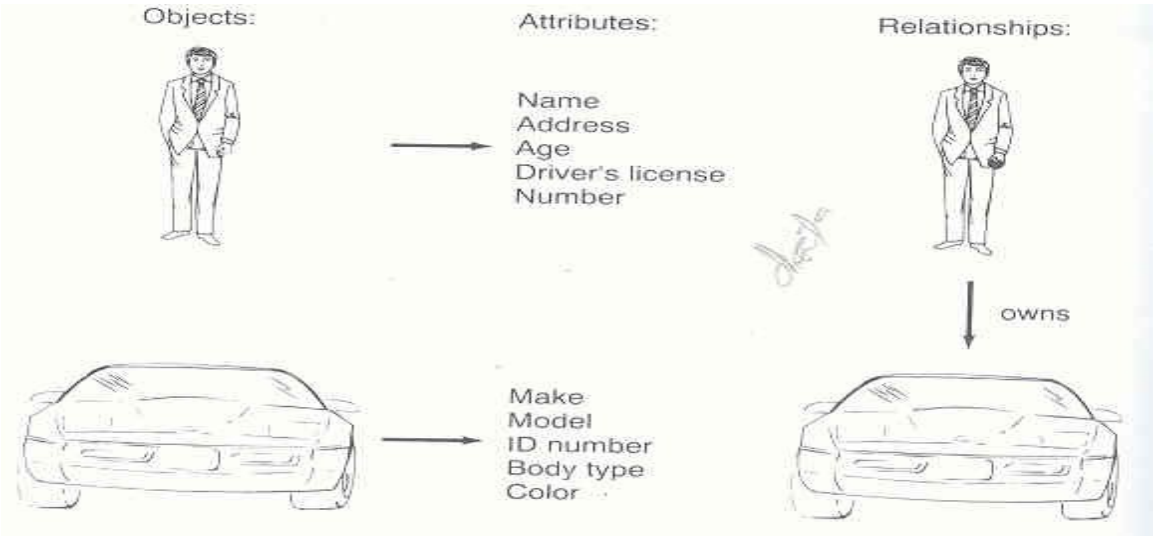
Data modeling is used extensively in data base application. It provides the data base analyst and designer with insight into the data and the relationship that govern data. In the contest of structure analysis data modeling can be used to represent the content of data stores and the relationship that exist among them. For example, a data item in one file structure might be a pointer into another file structure. Structure analysis notation provides no direct mechanism for presenting this relationship. Data modeling does

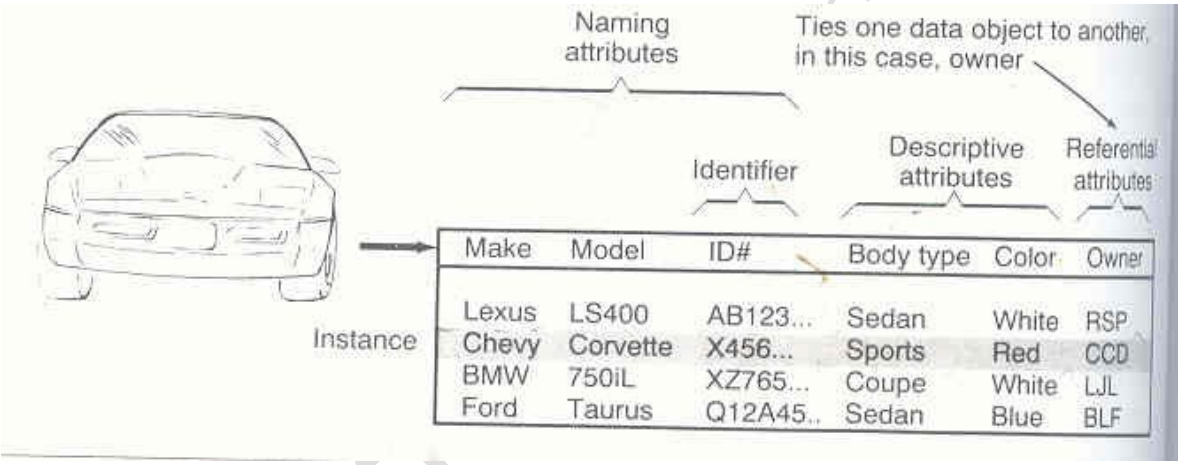
.

DATA OBJECTS, ATTRIBUTES, AND RELATIONSHIP

When data modeling is used as requirement analysis technique, the analyst begins by creating models for data objects. A data object is defined in much the same way that an object is define for OOA.

The data object can be represented as a table. The heading in the table reflects attributes of the object. In this case a car is define in terms of make, model, ID#, body type, color, and owner the body of the table represent specific instance of the data object. For example, a Chevy corvette is an instance of the data object car.





Data object attributes take on one of three different characteristics. They can be used to

- Name an instance of the data object.
- Describe the instance.
- Make reference to another instance in another table.

In addition, one or more of the attributes must be define as an identifier- that is, the identifier attributes becomes a" key" when we want to find an instance of the data object. In some case, values for the identifier are unique; all though this is not are requirement.

Data object table can be "formalized" by applying a set of normalization rules that will result in a relational model for the data. Normalization rules, when applied to the data object tables, result in minimum redundancy – that is, the amount of information that we need to maintain to satisfy a particular problem is minimized.

1. A given instance of data object has one and only one value for each attributes.

For example this rules required that the car table have only one owner for each instance of car. If Chevy corvette had two owners, the data object would have to be redesigned. Additionally the table assumes to have an entry for every attributes. There should be no "holes" in the table. This rule helps us define a relation when we used the jargon of data base designer.

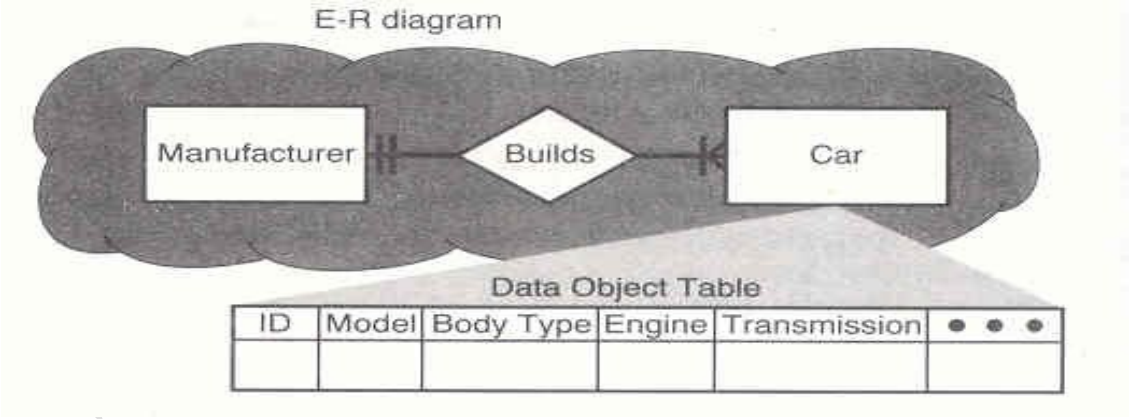
2. Attributed represent elementary data items, they should contain no internal structure.

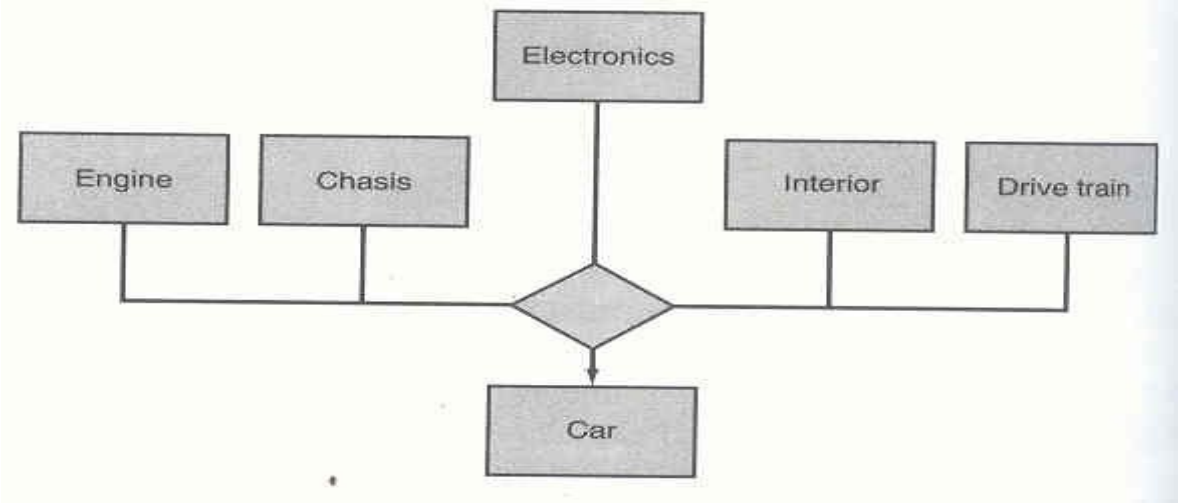
For example, if body type where define as
Body type = no of doors +height +width + name
Name = [coupe/ sedan /sports/luxury]
We would have a violation of thin rule.
To accommodate a situation in which an attributes is a composite data item, we must define the table so that each of the data component of body type are noted as separate attributes.

3. When more than one attributes is used to identify the data object, be sure that description and referential attributes be present " a characteristic of the entire object, and not a characteristic of some thing that would be identified by only part of the identifier ".
4. All non-identifier attributes must represent some characteristic instance named by the identifier of the data object and describe some other attributes that are not an identifier.

ENTITY – RELATIONSHIP DIAGRAM

The primary purpose of the ER diagram is to represent data object and the relationship. ER diagram notation is relatively simple. Data object are represented by labeled rectangle. Relationships are indicated with diamonds. Connection between data object and relationship are established using variety of special connection lines. Ex. One manufacturer builds one or many cars as shown in figure
ER notation provides a mechanism that represent the associatively between object.
The data object that model the individual sub system are each associated with the data object car. Data modeling and the entity relationship diagram provides the analysis with a concise notation for examine data within the context of a data processing application. In most cases, the data modeling approach is used as an adjunct to structure analysis, but it can also be used for data base design and to support any other requirements analysis method.





REQUIREMENT ANALYSIS METHOD

Requirement analysis methods enable an analyst to apply fundamental analysis principles in a systematic fashion.

Common characteristics

Requirement analysis methods have more in common than a cursory inspection might indicate. Each support the fundamental requirements analysis principle; each create a hierarchical representation of a system; each demand a careful consideration of external and internal interface; each provide a foundation for the design and implementation steps That follow; and non focuses serious attention on the representation of constraints or validation criteria.

Although each method introduces new notation and analysis heuristics, all methods can be evaluated in the context of the following common characteristics:

1. Mechanism for information domain analysis.
2. Approach for function and/or behavioral representations.
3. Definition of interfaces.
4. Mechanisms for problem partitioning.
5. Support for abstraction, and
6. Representation of essential and implementation views.

Differences in analysis methods

Each method for the analysis of computer-based systems has its own point of view, its notation, and its own approach to modeling.

The degree to which the method establishes a firm foundation for design differs greatly. In some cases, the analysis model can be mapped directly into a working program. In other cases, the analysis method establishes a starting point only and designer is left to derive the design with little help from the analysis model.

The level of CASE tool support varies greatly among the methods. Structured analysis-the most widely used method is supported by dozens of high-quality CASE tool. But other more obscure methods may have only one rudimentary tool available.

DATA STRUCTURE ORIENTED METHOD

We have already noted that the information domain for a software problem encompasses flow, content, and structure. Data structure – oriented analysis method represent software requirement by focusing on data structure rather than data flow. Although each data structure-oriented method has a distinct approach and notation, all have some characteristic in common:

1. Each assists the analyst in identifying key information objects and operations.
2. Each assumes that the structure of information is hierarchical.
3. Each requires that the data structure be represented using the sequence, selection, repletion constructs for composite data.
4. Each provides a set of steps for mapping a hierarchical data structure into a program structure.

Data structure-oriented analysis methods lay the foundation for software design. In every case, an analysis method may be extended to encompass architectural and procedural design for software.

Data structured system developments

Data structure system development (DSSD), also called warnier-Orr methodology, evolved from pioneering work on information domain analysis conducted by J.D.warnier. warnier developed a notation for representing an information hierarchy using the three construct for sequence, selection, and repetition and demonstrated that the software structure could be derived directly from the data structure.

Ken Orr has extended warnier works to in compass some what broader view of the information domain that has evolved into data structured system development. DSSD considers information flow and functional characteristic as well as data hierarchy

WARNIER DIAGRAMS

The warnier diagram enables the analyst to represent information hierarchy in a compact manner. The information domain in analyzed and the hierarchical nature of the output is represented. The general organization of the paper takes the following form:

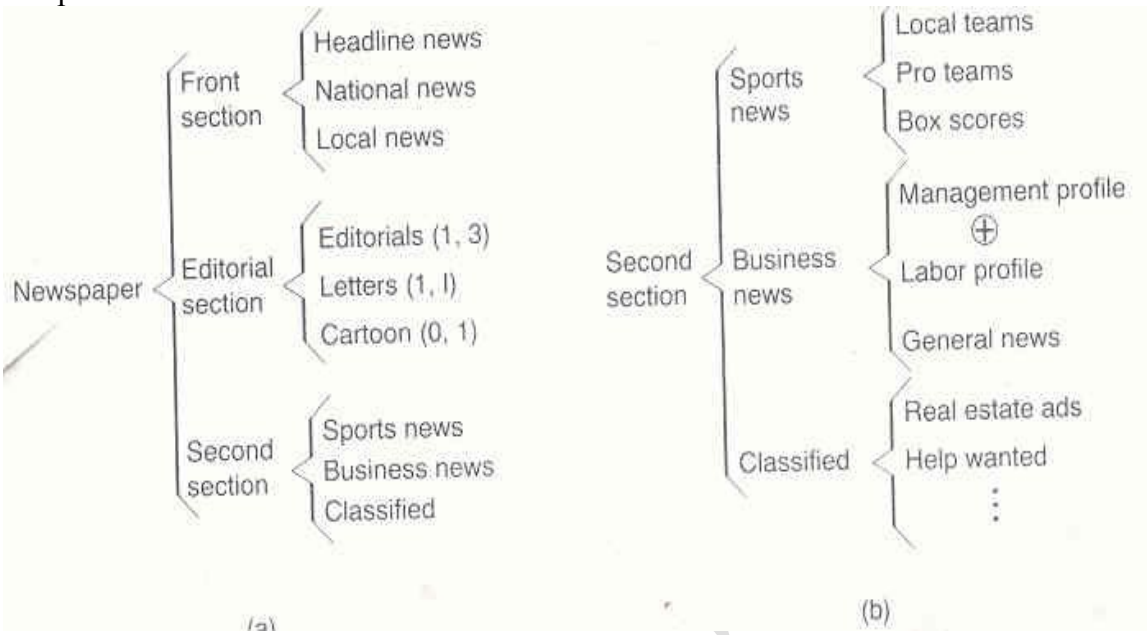
Front section
- Headline news
- National news
- Local news

Editorial section
- Editorial columns
- Letters to the editors
- Satirical cartoon

Second section
- Sports news
- Business news
- Classified

The newspaper outline shown above is an information hierarchy. The warnier diagram may be used to represent the hierarchy at any level of detail. The newspaper information hierarchy is represented using warnier notation. The brace ({) is used to differentiate levels of the information hierarchy. All names contained within a brace represent a sequence of information items. The notation next to some names represents repetition, that is, the number of times the particular item appear in the hierarchy. For example, 1 to

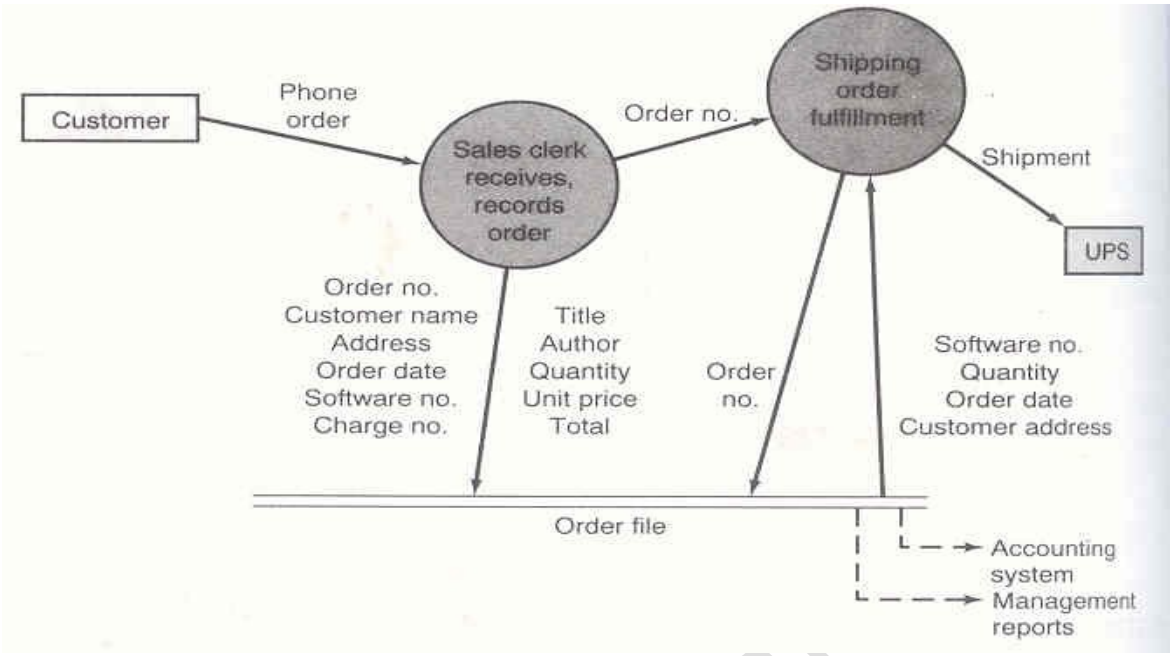
3 editorials will appear in the editorial section, while a cartoon may or may not be present (appears 0 to 1 times).
The warnier diagram may be used to further partition the information domain by refining composite data items.



THE DSSD APPROACH

Rather than beginning analysis by examining the information hierarchy, DSSD first examines the application context, that is, how data moves between producers and consumers of information from the perspective of one of the producers or consumers. Next, application functions are assessed with a wormier-like representation that depicts information items and the processing that must be performed on them. Finally application results are modeled using the wormier diagram. Using this approach, DSSD encompasses all the attributes of the information domain: data flow, content, and structure.

To illustrate DSSD notation and to provide an overview of the analysis method, we present a simple example. A mail/ phone order business, called the software store, sell personal computer software. A computer-based order processing system is to be specified for the business.



APPLICATION CONTEXT

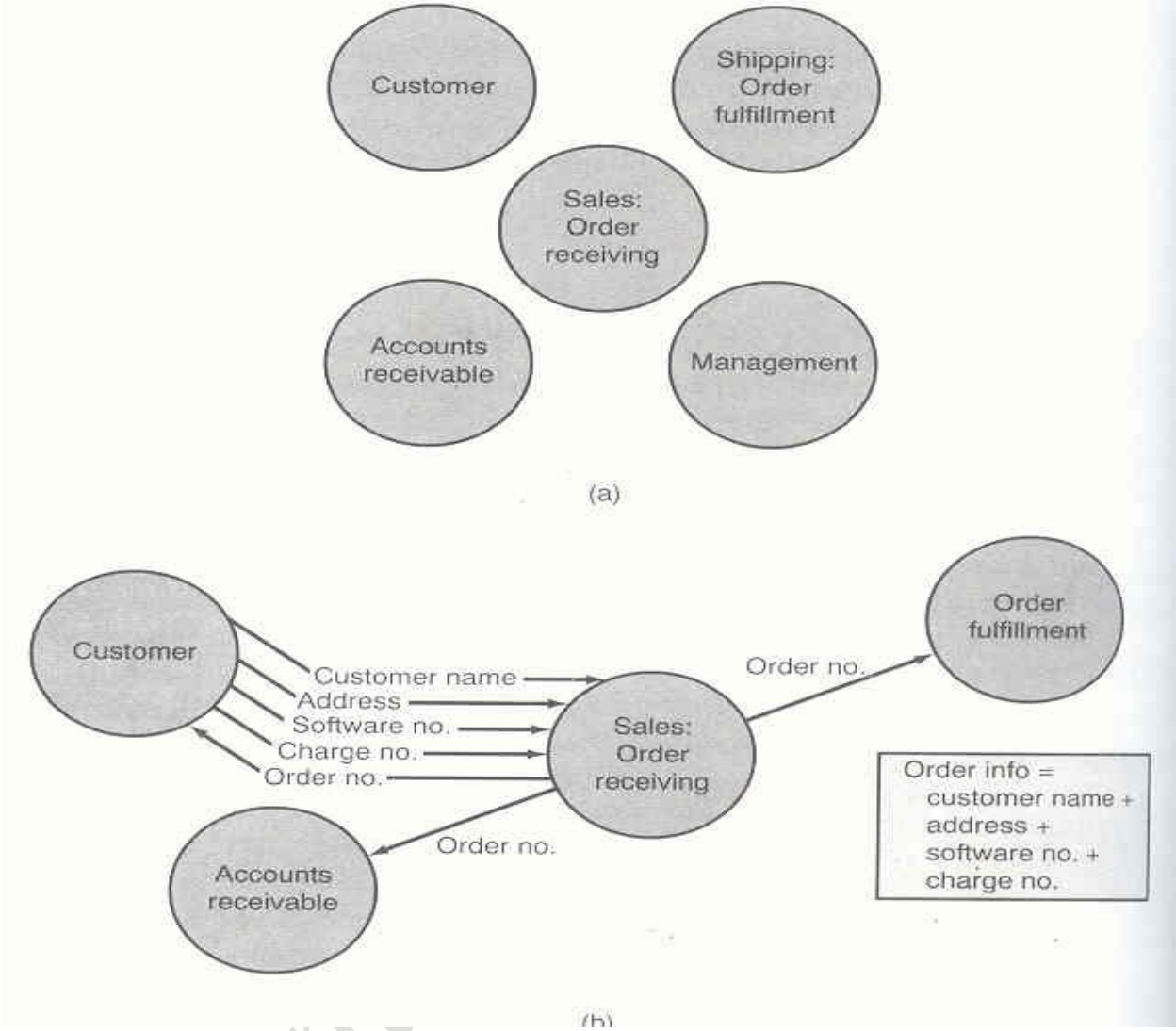
To determine the DSSD application context, the problem must be stated in a manner that enables us to answer three questions:

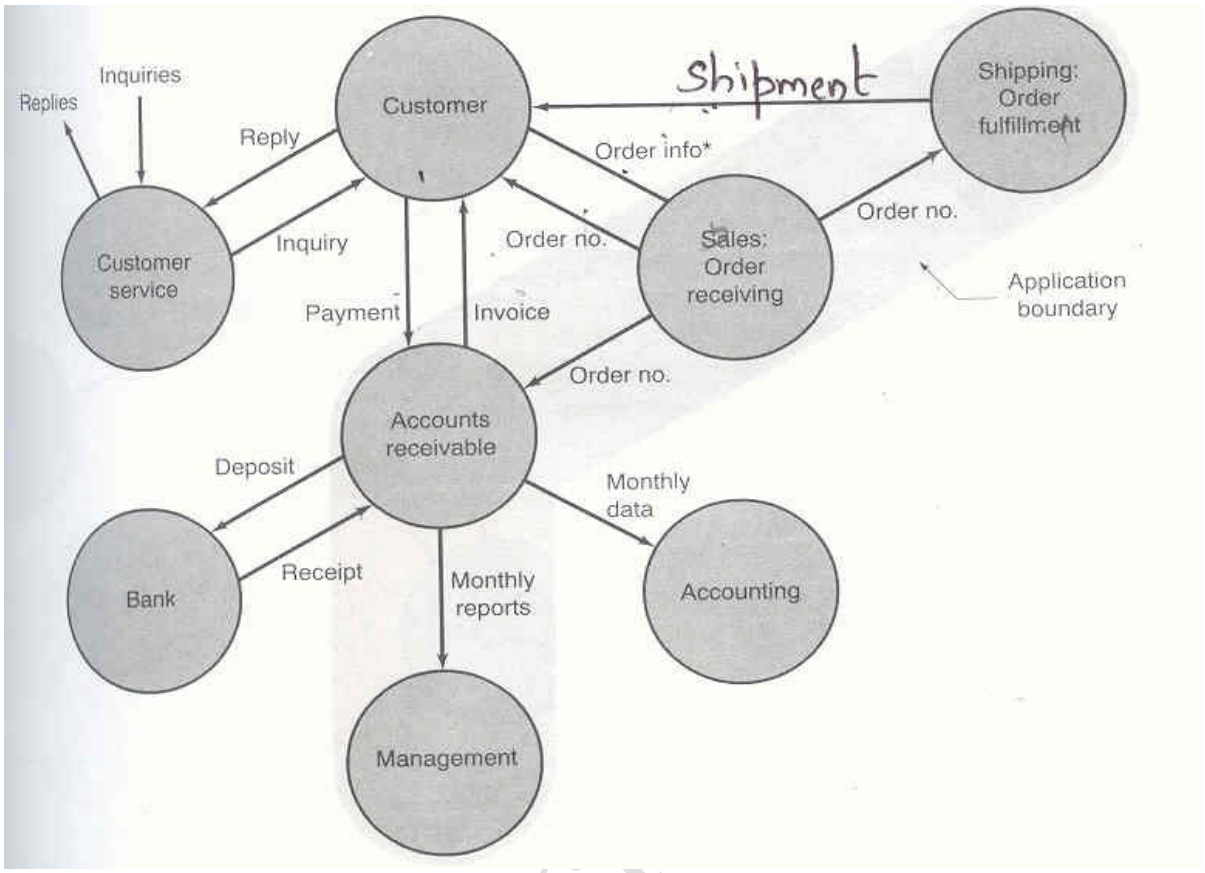
1. What are the information items that must be processed?
2. Who/ what are the producers and consumers of information?
3. How does each producer/ consumer view information in the context of other constituencies?

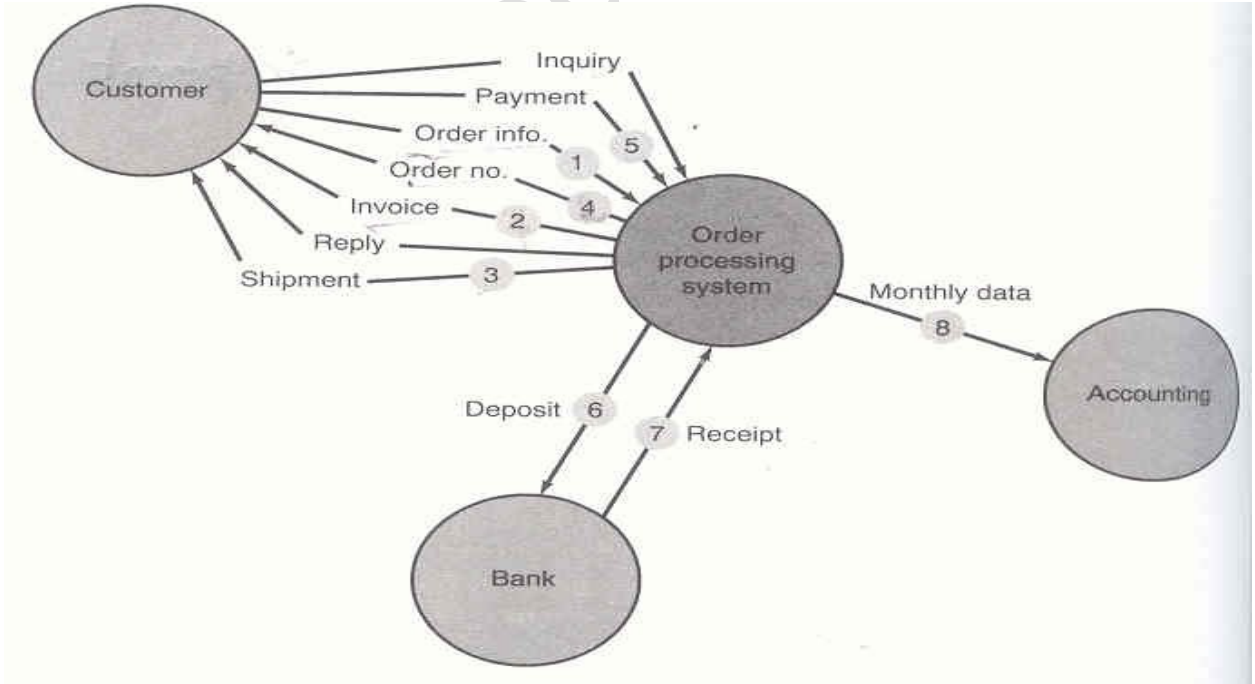
DSSD proposes an entity diagram as a mechanism for answering these questions.

The entity diagram uses a notation that is, regrettably, very similar to the data flow diagram. However, similar symbols have different meanings. The circle in an entity diagram depicts a producer or consumer of information. The five producers and consumers of information for the software store .an entity diagram for the sales: order receiving department. All interfaces between sales: order receiving and other constituencies are shown from the point of view of sales: order receiving. Entity diagrams for other producers and consumers of information could also be developed.

After each entity diagram is reviewed for correctness, a combined entity diagram is created for all producers and consumers of information. Those entities that fall within the bounds of the proposed system are indicated by identifying an application boundary. The detail within the application boundary may be hidden. Information moving across the application boundary must be processed by the order processing system to be analyzed.


Customer
Shipping: Order fulfillment
Sales: Order receiving
Accounts receivable
Management
(a)
Customer
Customer name
Address
Software no.
Charge no.
Order no.
Sales: Order receiving
Order no.
Order fulfillment
Order info = customer name + address + software no. + charge no.
Order no.
Accounts receivable
(b)

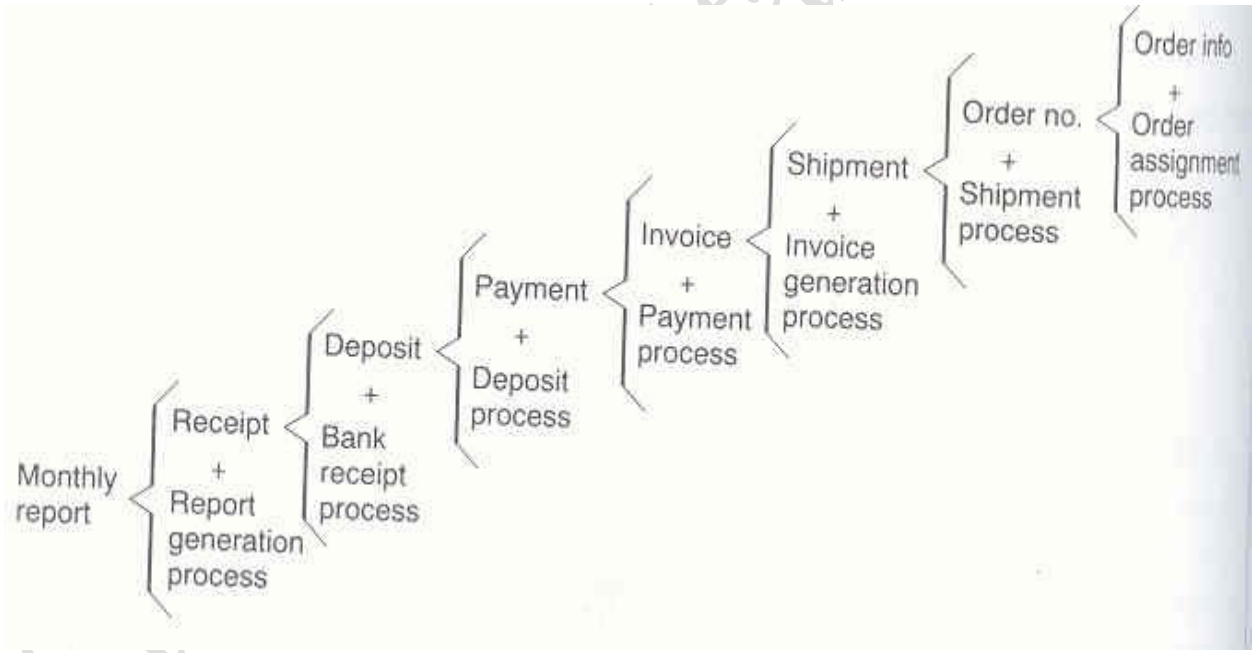

Customer
Shipment
Shipping: Order fulfillment
Inquiries
Replies
Reply
Order info*
Customer service
Inquiry
Order no.
Sales: Order receiving
Order no.
Application boundary
Payment
Invoice
Order no.
Accounts receivable
Deposit
Monthly data
Bank
Receipt
Monthly reports
Accounting
Management


Customer
Inquiry
Payment
Order info.
Order no.
Invoice
Reply
Shipment
1
2
3
4
5
Order processing system
Monthly data
8
Accounting
Deposit
6
7
Receipt
Bank

APLICATION FUNCTIONS

The functions that must be implemented to accomplish the automated order processing system can be discerned by examining information flow across the application boundary. Using earlier figure as a guide, the sequence in which data items move across the boundary is noted as shown in the figure. Using a warnier-like notation called an assembly line diagram; DSSD provides a mechanism for coupling information and the processes that are applied to it. Conceptually, the ALD plays the same role as the data flow diagram.

Beginning with the last numbered information flow develops an assembly line diagram and working backward showing the processing that derives the preceding numbered information item. An ALD for the order processing system is shown in figure. Reading left to right, taking bank receipt information derives the monthly report and applying a report is derived by taking bank receipt information and applying a report generation process. The plus sign indicates the coupling between process and information. Receipts information is derived from a bank deposit and the associated function that processes the deposit, producing a receipt. A similar progression occurs until we reach order info, the first input in the sequence.



APPLICATION RESULTS

DSSD requires the analyst to build a paper prototype of desired output for the system. The prototype identifies primary system output and the organization of information items that comprise the output. Once a prototype has been created, the information hierarchy may be modeled using a warnier-Orr diagram essentially a warnier diagram with small variations in notation and format.

To illustrate the use of paper prototyping and the Warnier-Orr diagram in the derivation of application results, we consider the monthly report generated as output from the automated order processing system for the software store. Figure (a) shows a paper prototype for the report and figure (b) illustrates a warnier-Orr diagram of the corresponding information hierarchy.