FIT1013 - Week 9 Resources

Modularisation, Structure Charts and Connecting to External Data

Week 9 Resources

Reference:

Satzinger, Jackson, Burd, Systems Analysis and Design in a Changing World, 7th Edition
Microsoft Excel 2016, New Perspectives Series, Parsons, Oja, Carey, Desjardins, Comprehensive Edn., Cengage Learning, **Modules 10,11,12**

1. Objectives

- Discuss program design approaches
- Design modules using structure charts
- Import data from text files

2. Two Approaches to Software Construction and Modelling

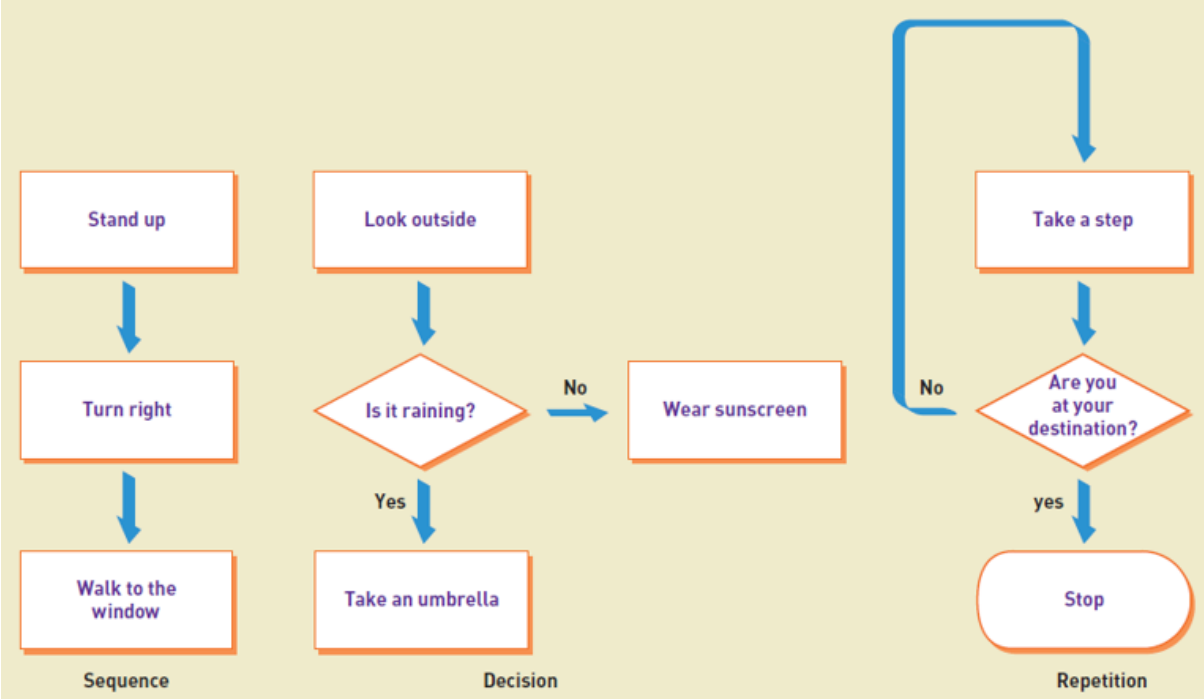
There are two main approaches to software development: structured and object-oriented. The structured approach is the earlier approach where the system is constructed in a modular manner. It assumes that a system is collection of processes (separable into a number of modules) that interact with data. Based on this idea, systems were constructed in a structured way from analysis, through to design and programming. The object-oriented approach on the other hand, assumes that the world consists of objects that interact to complete tasks. These objects can be classified into groups enabling concepts like inheritance to be applied in all stages of software development, i.e. analysis, design and programming.

3. The Structured Approach

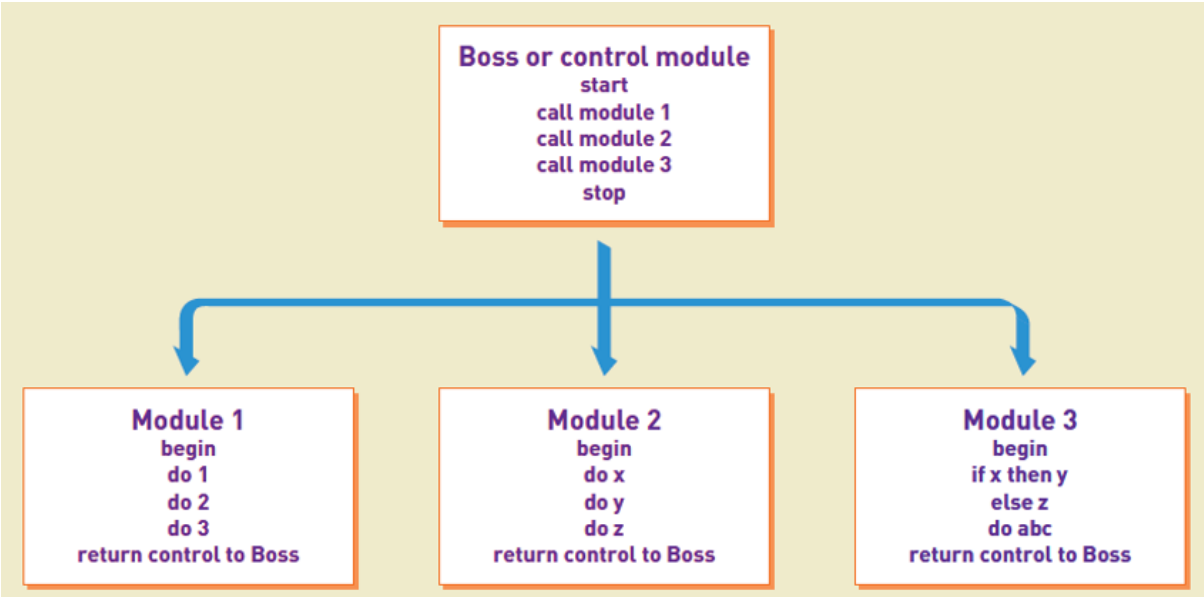
In structured programming, there are 3 basic program structures:

- Sequence
- Selection/decision
- Repetition

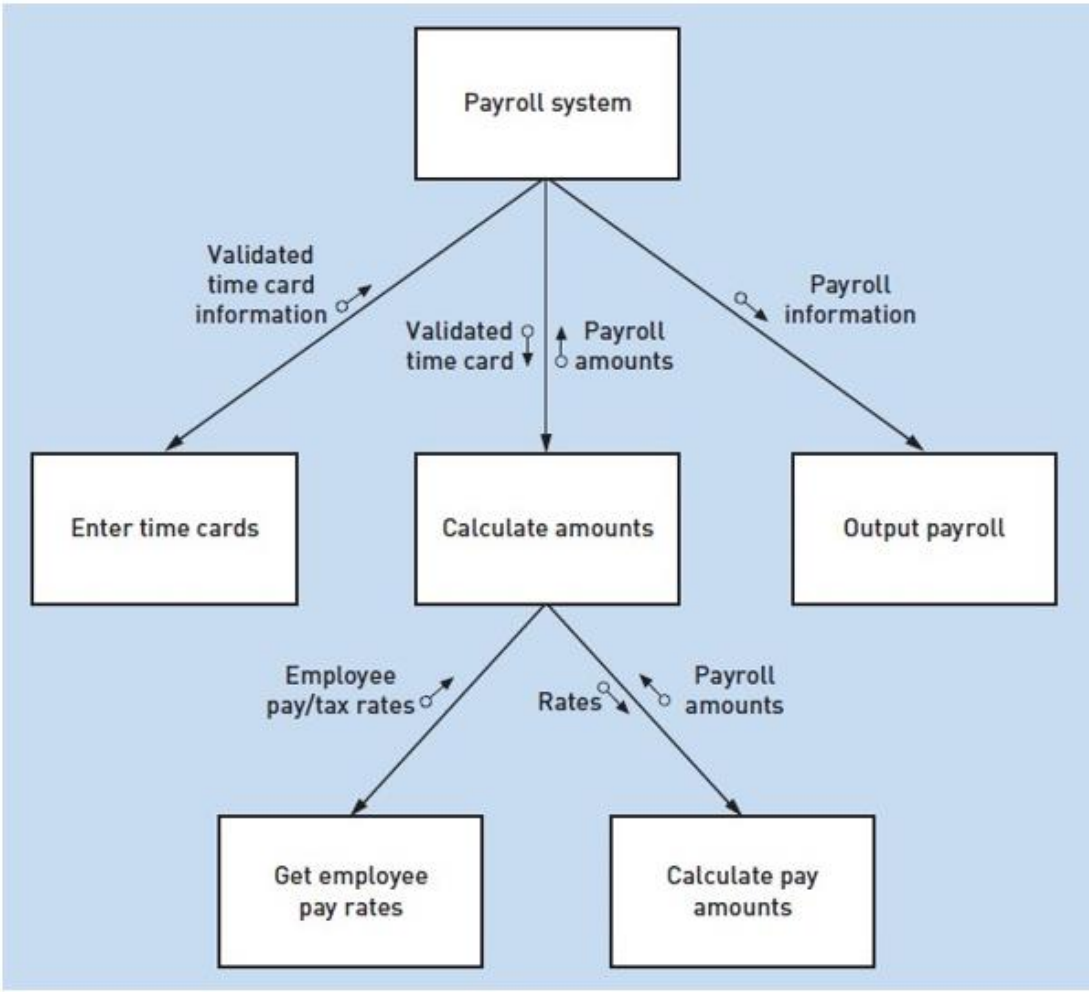
The following diagram shows the flow-chart of these structures.



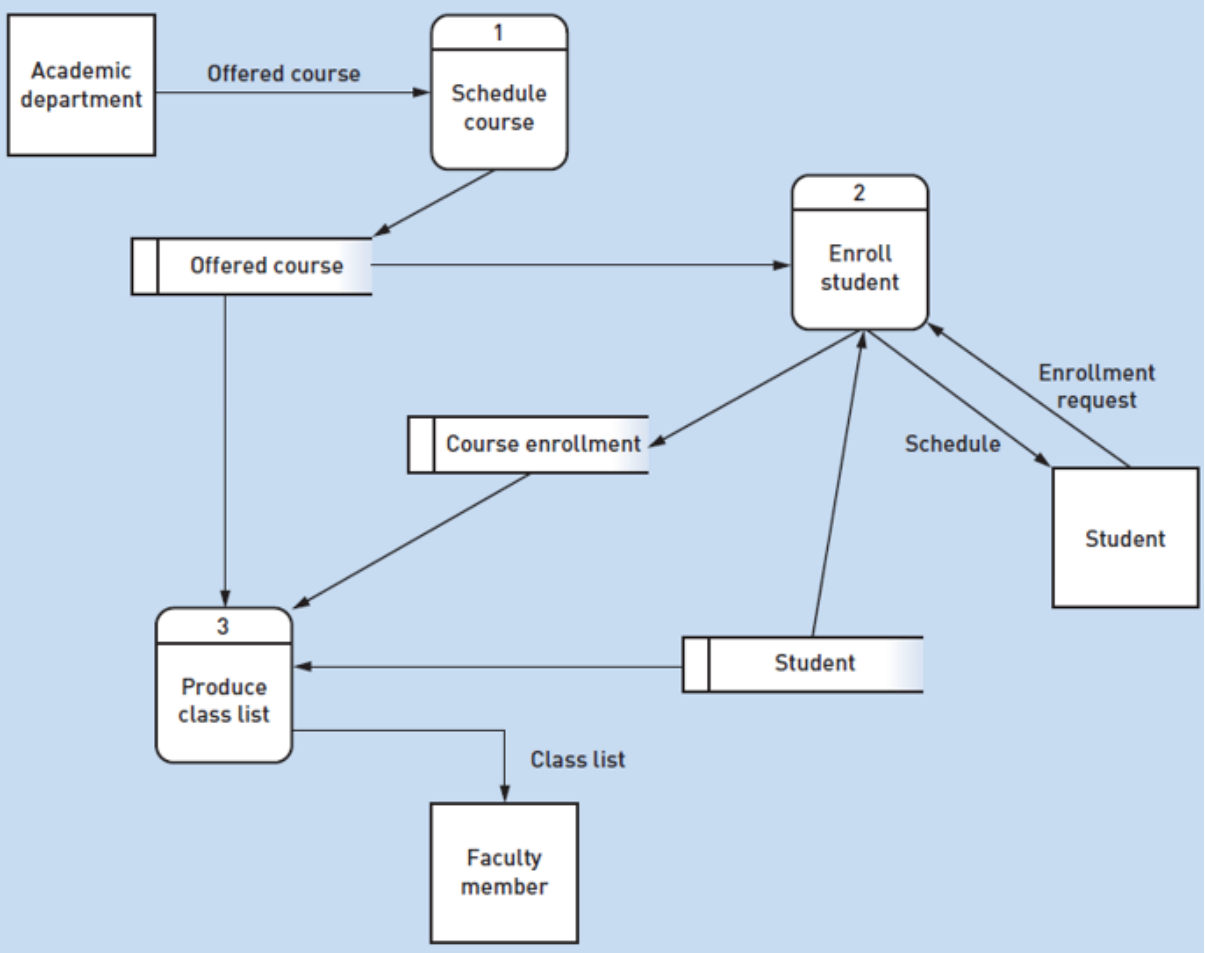
The next diagram shows a top down, modular programming approach whereby a top module (called the boss or control module) provides a pathway to 3 other modules. At the end of each of these modules, control is returned to the top module.



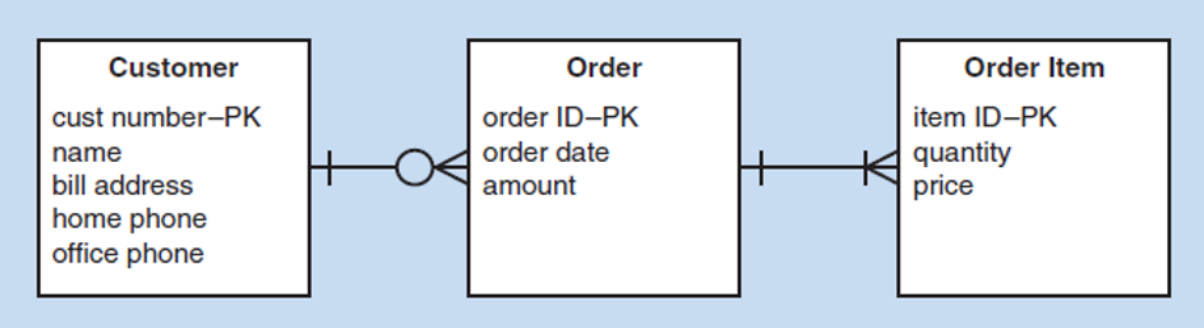
When designing the program, we use a structure chart as shown below where the data flow can be shown. These arrows are known as data couples as shown in the following diagram.



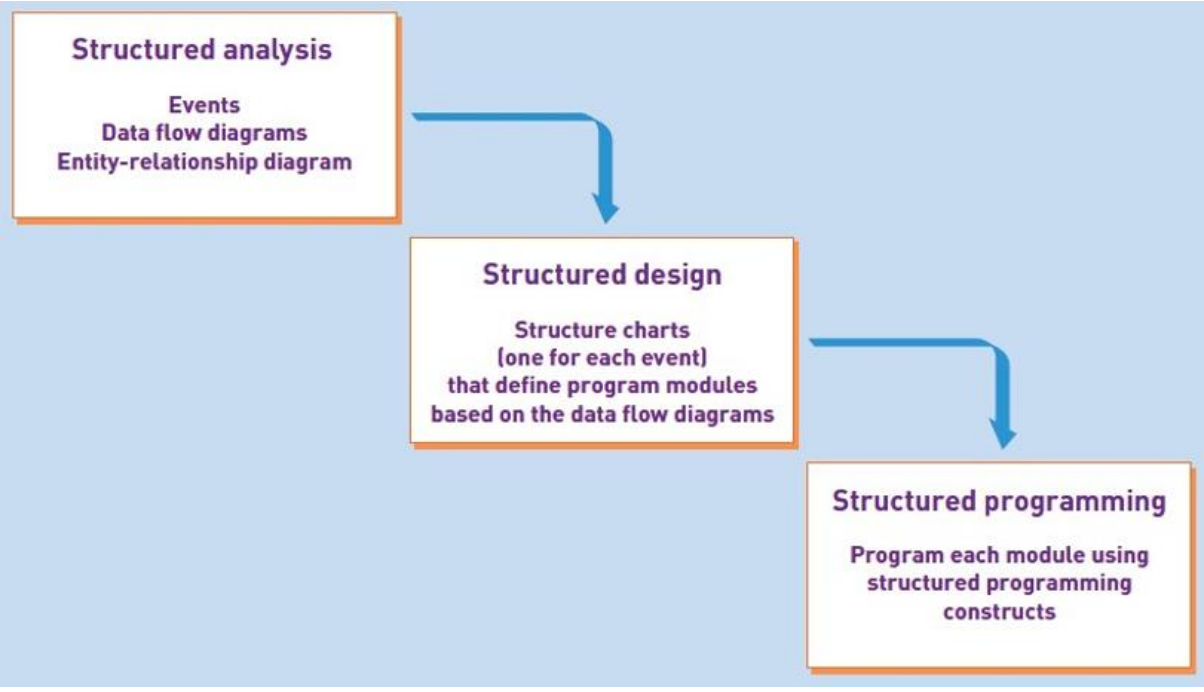
A data flow diagram (DFD) is sometimes used in the analysis phase of a software development project. The following diagram shows a DFD for student enrolment at an academic department.



The next diagram is the entity-relationship diagram (ERD). This shows the relationship between objects/entities/classes. It doesn't capture the programming logic but gives information on how data is related and hence is used when designing database operations.



The following diagram shows the relationship between analysis, design and programming. Traditionally, this is the waterfall model where one stage flows into another. Today, most projects will allow a feedback loop to the previous stage. You will learn more about systems development methodologies in FIT2001.



Structured Analysis

- Define what system needs to do (processing requirements)
- Define data system needs to store and use (data requirements)
- Define inputs and outputs

- Define how functions work together to accomplish tasks
- Data flow diagrams (DFD) and entity relationship diagrams (ERD) show results of structured analysis

Structured Design

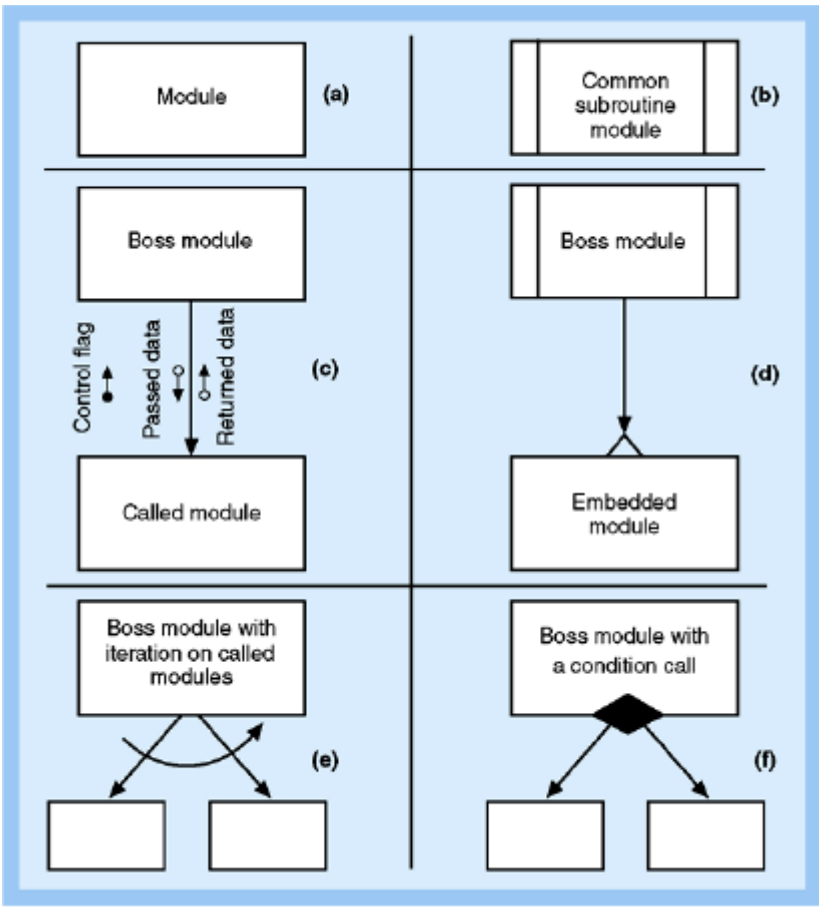
- Technique developed to provide design guidelines
    - What set of programs should be
    - What program should accomplish
    - How programs should be organized into a hierarchy
- Modules are shown with structure chart
- Main principle of program modules
    - Loosely coupled – module is independent of other modules
    - Highly cohesive – module has one clear task

Structured Programming

- Produces a program that has one beginning and one ending
- Each step in the program execution consists of one of the three programming constructs:
    - A sequence of program statements
    - A decision point at which one set or anohter set of statemets executes
    - A repetition of a set of statements

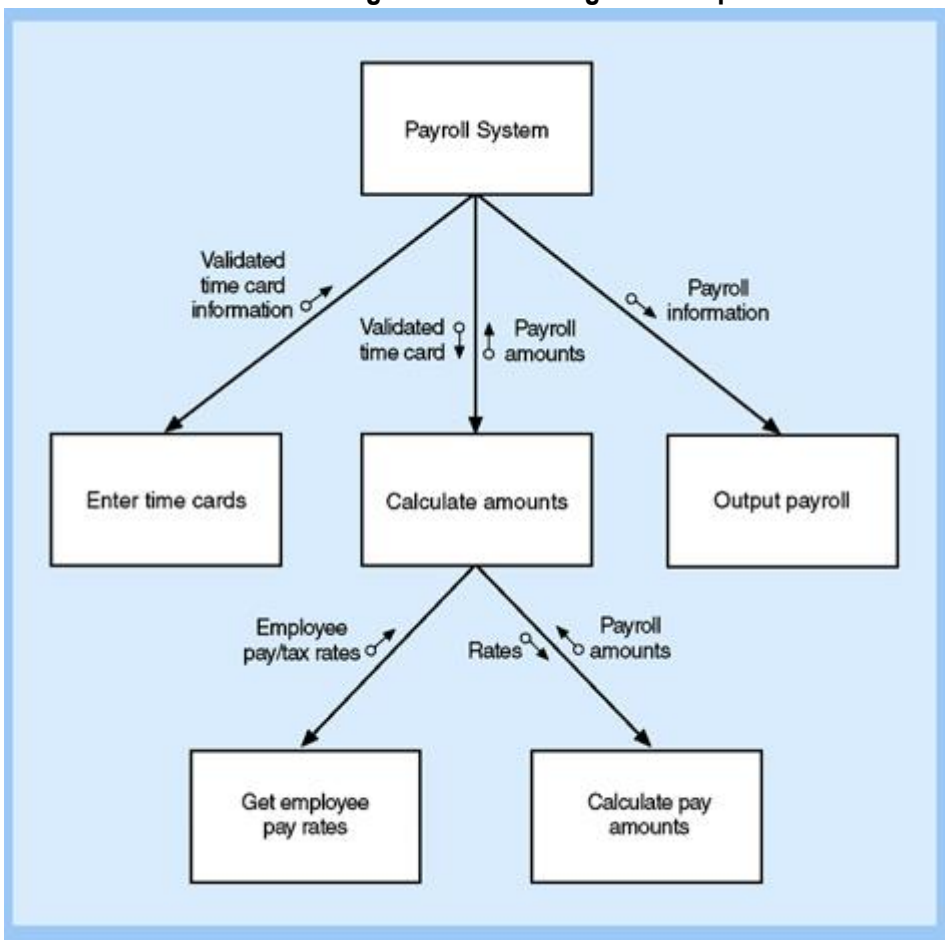
3.1 The Structure Chart

This describes functions and sub functions of each part of system. It also shows the relationships between modules of a computer program. Each module performs a specific function and each layer in a program performs specific activities. The following diagram shown the structure chart symbols.

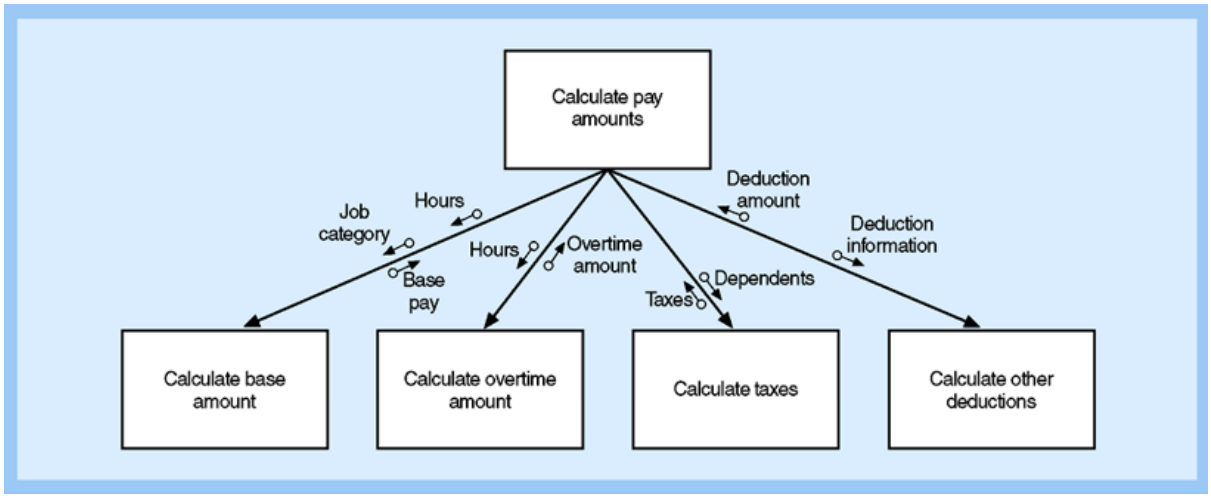


The structure chart is tree-like with root module and branches as shown in the following diagram.

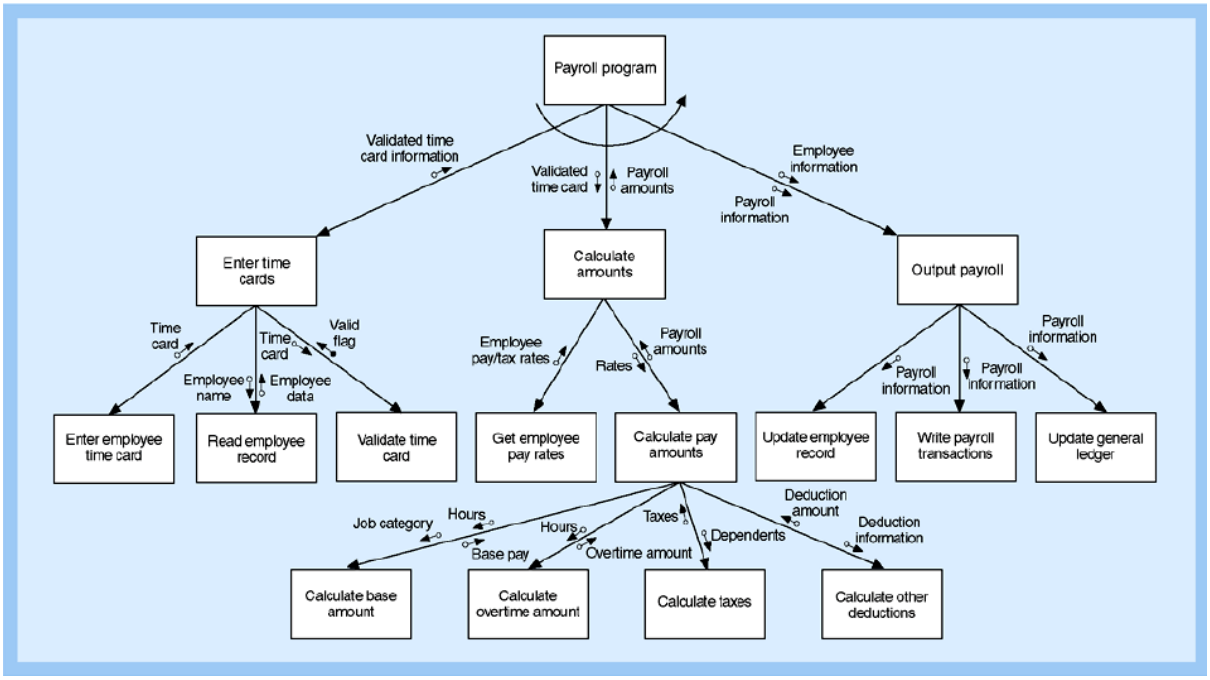
Structure Chart Created Using Structured Design Technique



The following diagram is a simple structure chart for the Calculate Pay Amounts Module



The next diagram shows the Structure Chart for Entire Payroll Program



3.2 Developing a Structure Chart

There are two ways to develop a structure chart. In transaction analysis, we start with the upper level modules and identify each transaction that is supported by the system. In this approach, the analyst will start by looking at the system flow chart and event table for the inputs to the system. (You will learn about system flow charts and event tables in FIT2090). Transform analysis on the other hand, uses DFD fragments for inputs. The computer program transforms the inputs into outputs.

Steps to Create a Structure Chart from a DFD Fragment

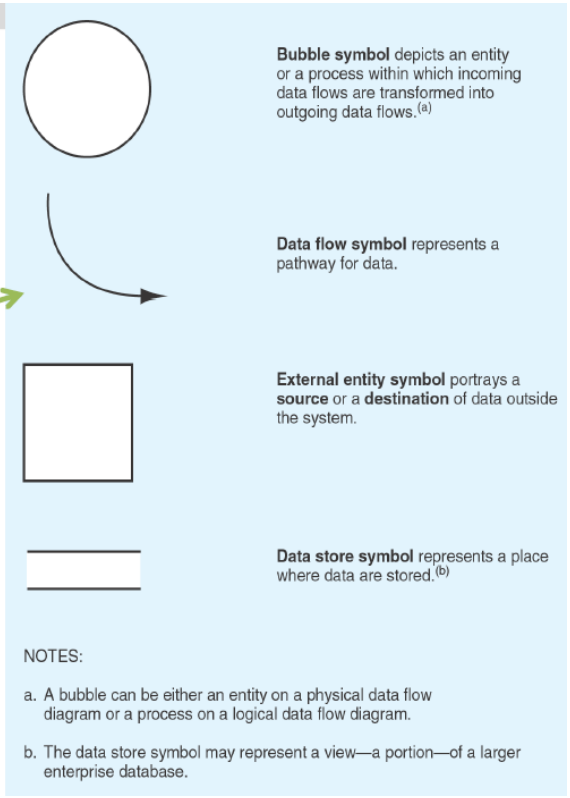
1. Determine primary information flow, i.e. the main stream of data that is transformed from some input form to an output form
2. Find process that represents most fundamental change from input to output
3. Redraw DFD with inputs to left and outputs to right and a central transform process usually goes in middle of the diagram
4. Generate first draft structure chart based on redrawn data flow

Some notes on Data Flow Diagrams

A DFD is a graphical representation of a system that depicts the systems components; the data flows among the components; and the sources, destinations, and storage of data. It uses a limited number of symbols and does not depict management or operational elements of a system.

Basic DFD Symbols

- Movement of data among:
  - Entities (sources or destinations)
  - Processes
  - Data stores.
- Label should describe the information moving



Data Flow Diagram Levels

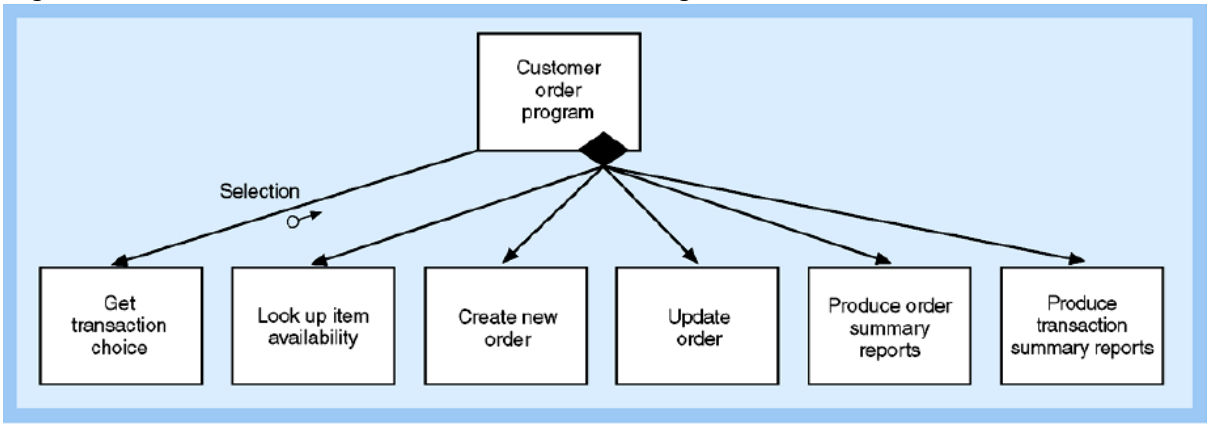
- Context
  - Highest level (most general)
  - Purpose: show inputs and outputs into system
  - Characteristics: one process symbol only, no data stores.
- Level-0
  - Purpose: show all major activity steps of a system
  - Characteristics: processes are labelled 1.0, 2.0 and so on.
- Level-1
  - Purpose: show one major activity divided into sub-activities
  - Characteristics: processes are labelled 1.1, 1.2 and so on.

Context Diagram

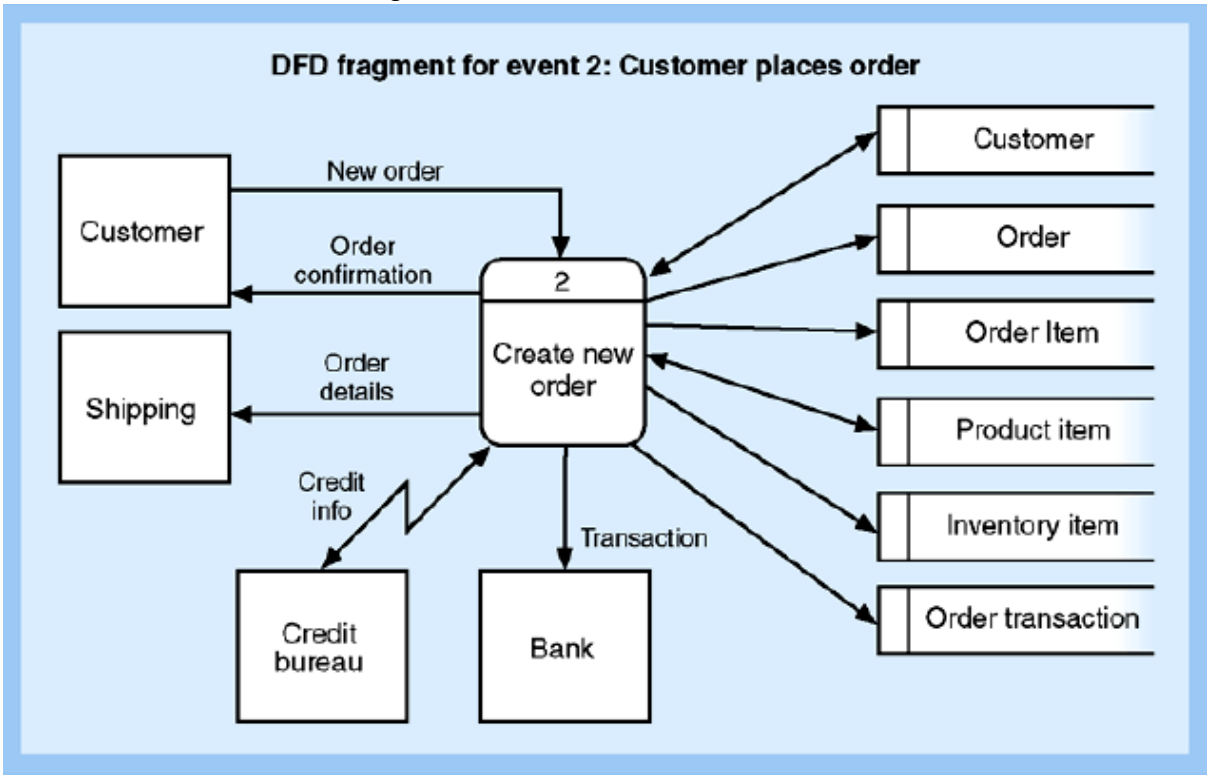
- Context diagram
  - top-level, and less detailed, diagram of a system depicting the system and all its activities as a single bubble and showing the data flows into and out of the system and into and out of the external entities.

- External entities
  - those entities (i.e., persons, places, or things) outside the system that send data to, or receive data from, the system.
- Internal entities
  - those entities within the system that transform data
  - Includes, for example, accounting clerks (persons), departments (places), and computers (things)

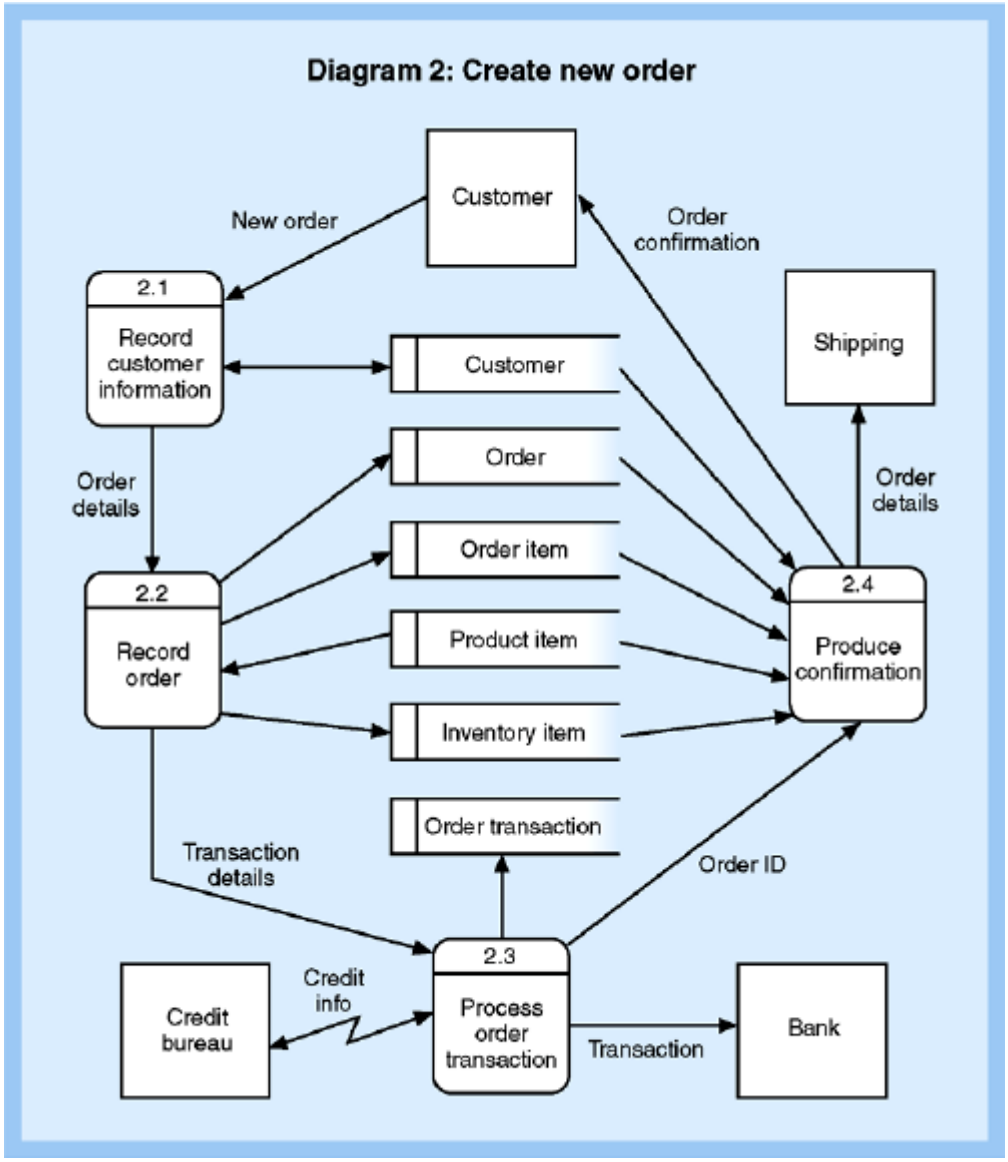
High-level Structure Chart for the Customer Order Program



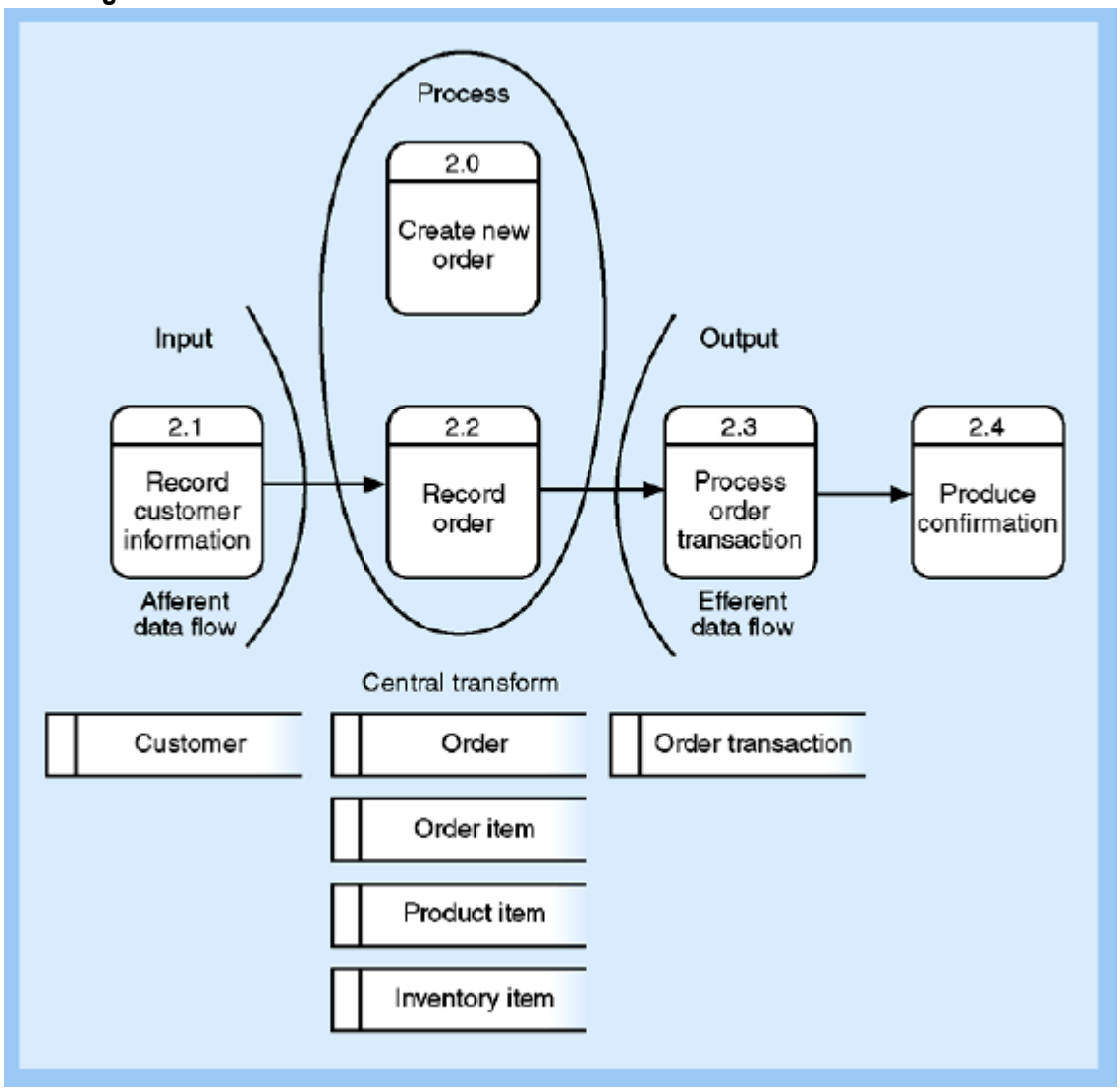
The Create New Order DFD Fragment



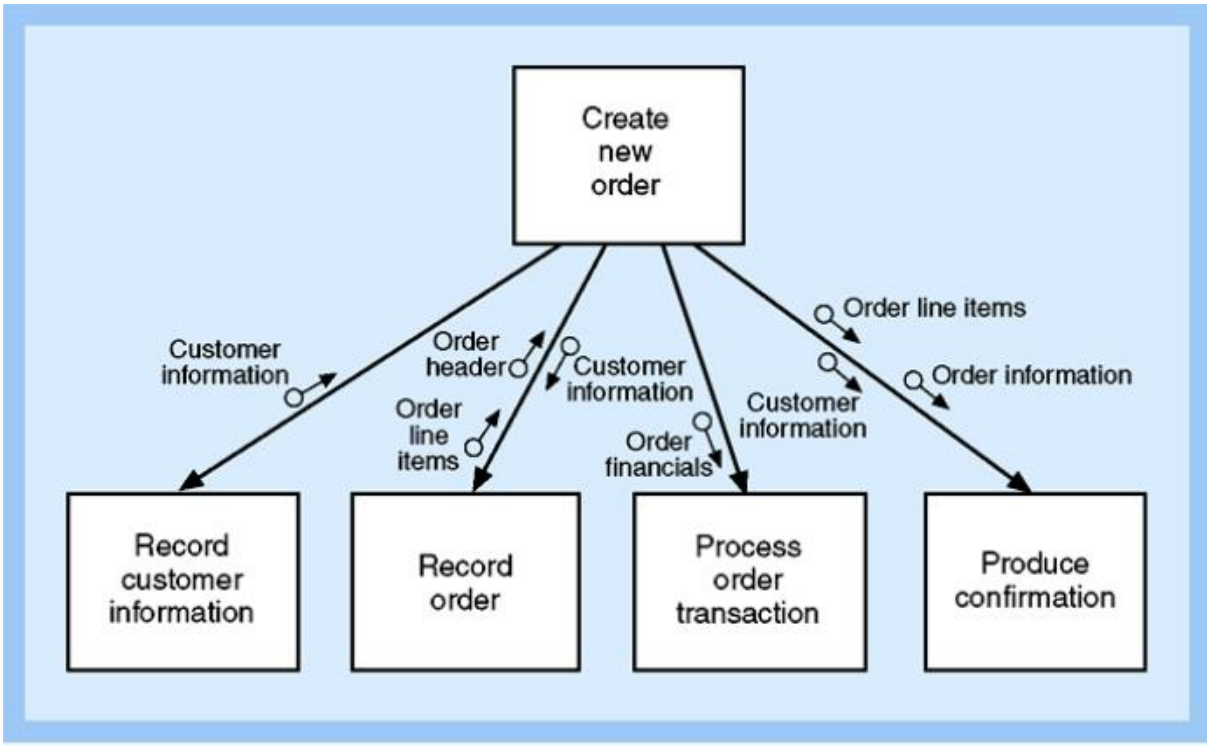
Exploded View of Create New Order DFD



Rearranged Create New Order DFD



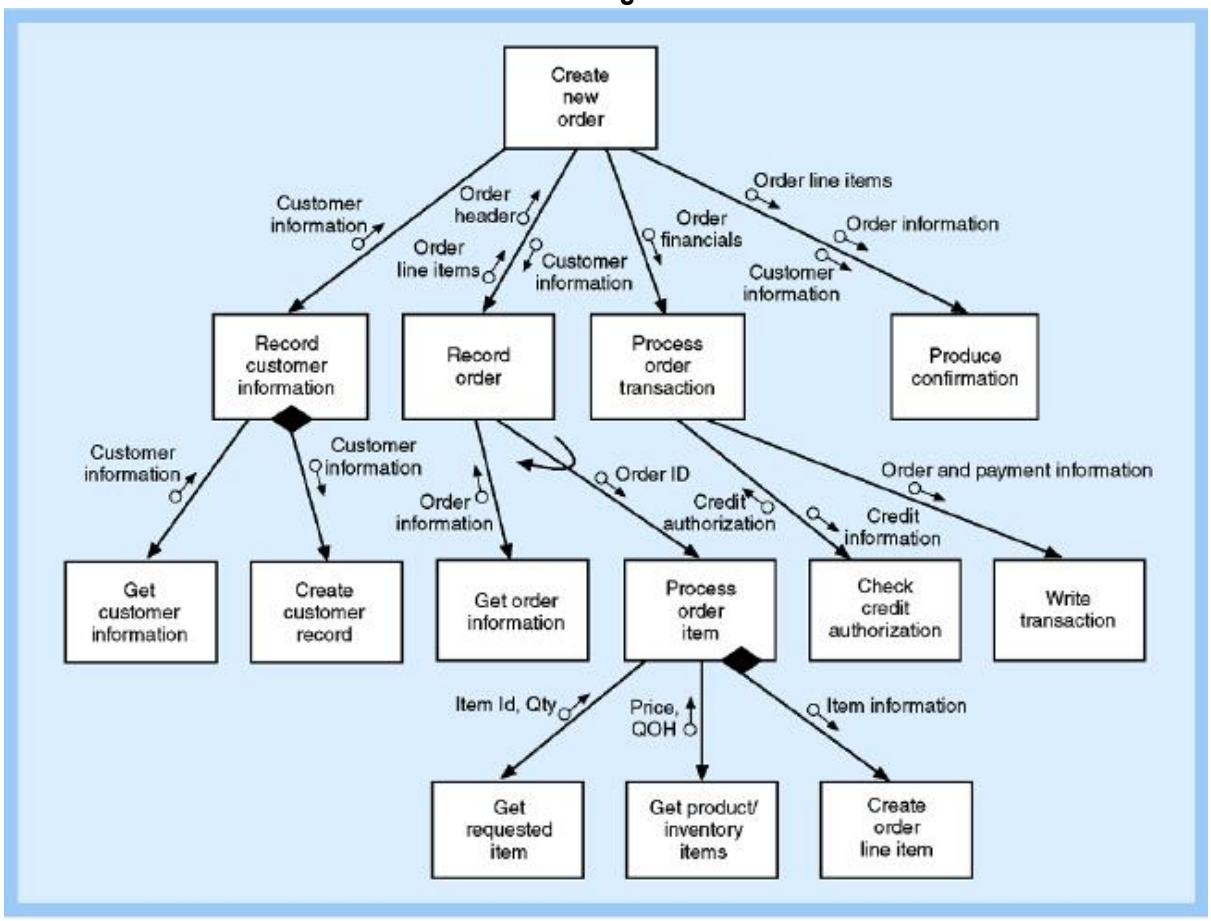
First Draft of the Structure Chart



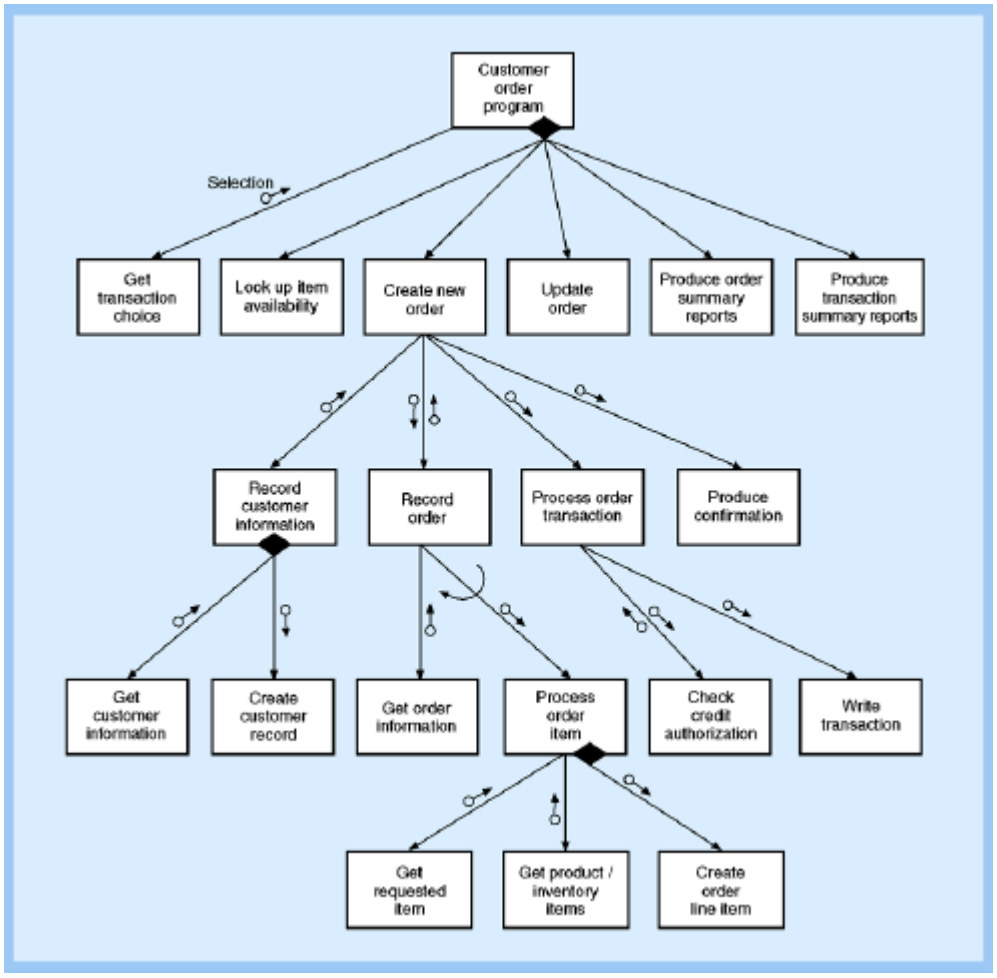
Steps to Create a Structure Chart from a DFD Fragment (continued)

- Add other modules
  - Get input data via user-interface screens
  - Read from and write to data storage
  - Write output data or reports
- Add logic from structured English or decision tables
- Make final refinements to structure chart based on quality control concepts

The Structure Chart for the Create New Order Program



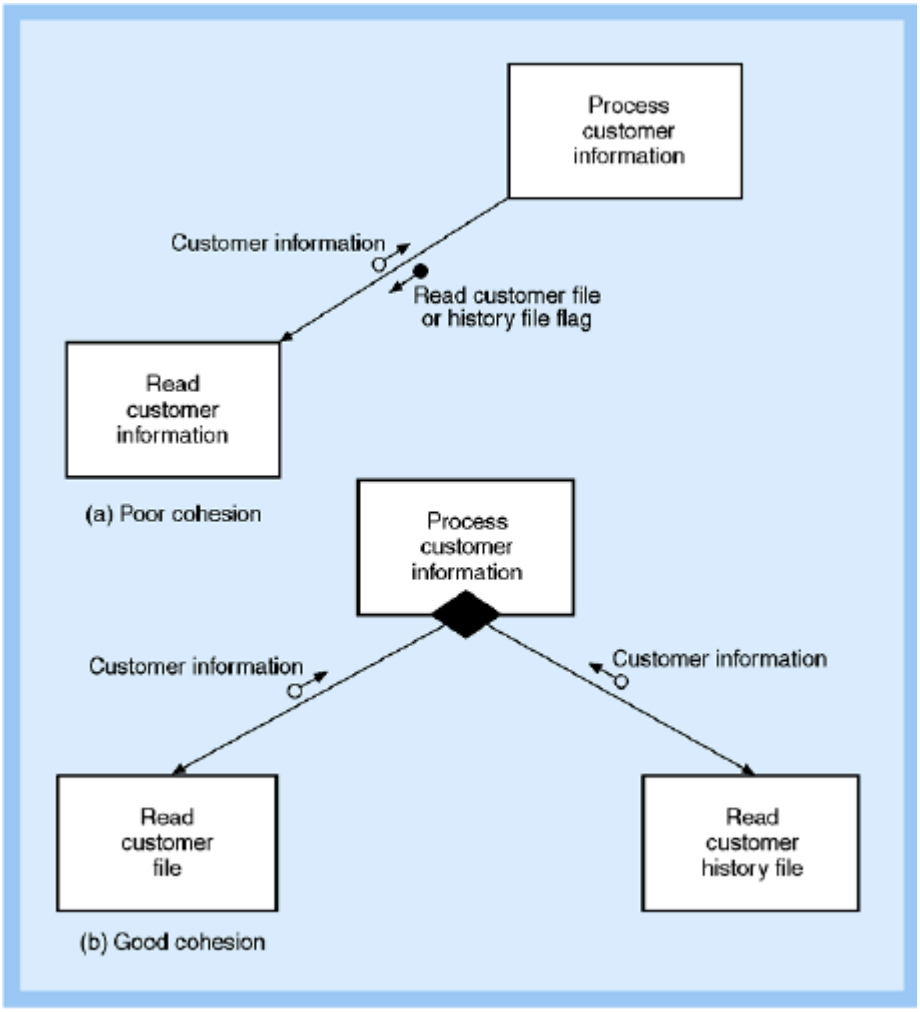
Combination of Structure Charts



3.3 Evaluating the Quality of a Structure Chart

There are two main measures for evaluating the quality of a structure chart. Module coupling is a measure of how module is connected to other modules in the program. The goal is to be loosely coupled so that modules are independent. Module cohesion is a measure of the internal strength of a module. Each module performs one defined task and the goal is to be highly cohesive so that all activities relating to the task are contained in one module.

Examples of Module Cohesion



In (a), it can be seen that the read function of the process can perform either one of the two tasks, i.e. read customer file or customer history file. In this case, there is poor cohesion because the process is concerned with more than one specific task. The diagram in (b) on the other hand is a better design because the activities are divided into two separate processes.

3.4 Module Algorithm Design: Pseudocode

This describes the internal logic of software modules. It consists of a variation of structured English that is closer to the programming code. However, the syntax should be close to the development language. There are three types of control statements used in structured programming:

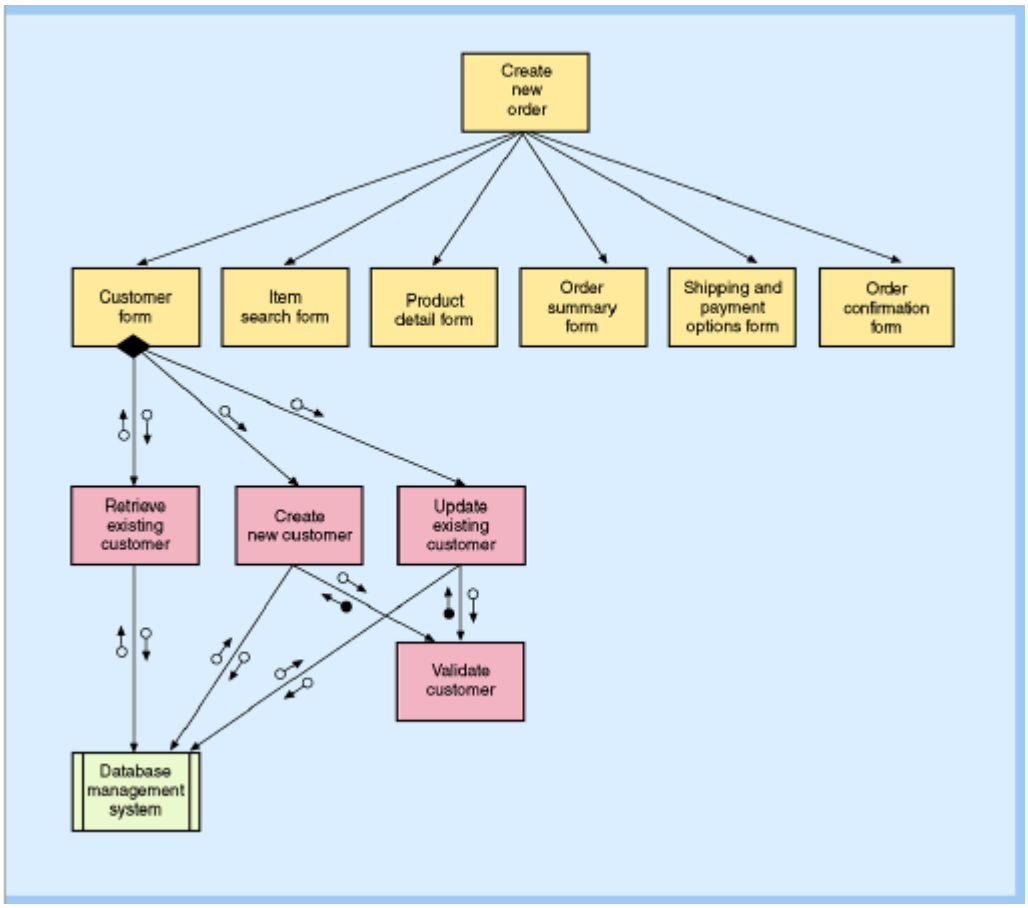
- Sequence: sequence of executable statements
- Decision: if-then-else logic
- Iteration: do-until or do-while

3.5 Integrating Structured Application Design with Other Design Tasks

The structure chart must be modified or enhanced to integrate with the design of the user interface and database. For example, the following are questions that need to be addressed:

- Are additional modules needed?
- Does pseudocode in modules need modification?
- Are additional data couples needed to pass data?

Similarly, structure charts must correspond to the planned network architecture, i.e. the required protocols, capacity, and security features of the system. The following is an example of a structure chart showing the Create New Order program.

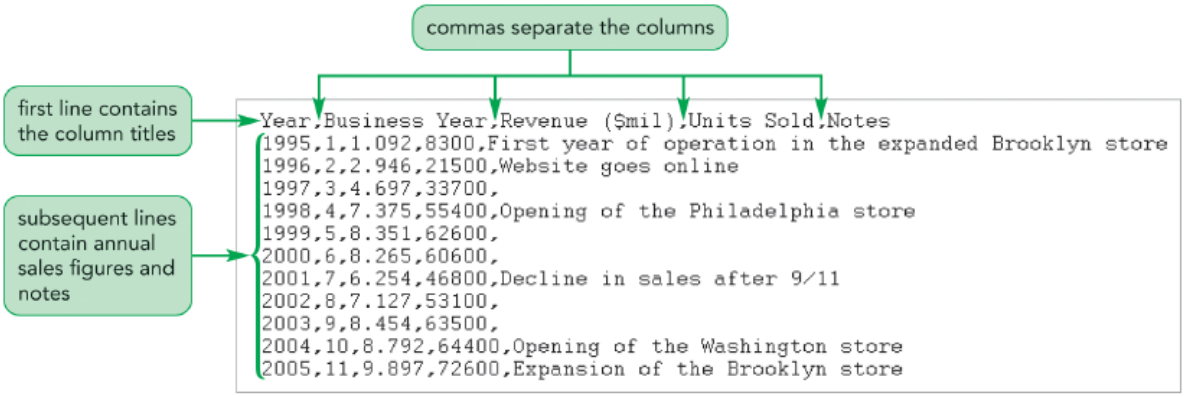


4. Object-Oriented Approach

This approach views the information system as a collection of interacting objects that work together to accomplish tasks. Objects are things in the computer system that can respond to messages. No processes, programs, data entities, or files are defined here, just the objects in the system. Like structured programming, there are three areas: analysis, design and programming. Object-oriented analysis (OOA) defines the types of objects that do the work of the system and shows how the objects interact with users to complete the tasks. Object-oriented analysis (OOA) is the process of identifying and defining the use cases and sets of objects (classes) in the new system. In object-oriented design (OOD), all of the types of objects necessary to communicate with people and devices are defined and their interactions to complete tasks are also shown. Object-oriented programming (OOP) involves writing statements that define the actual classes and what each object of the class does in the system.

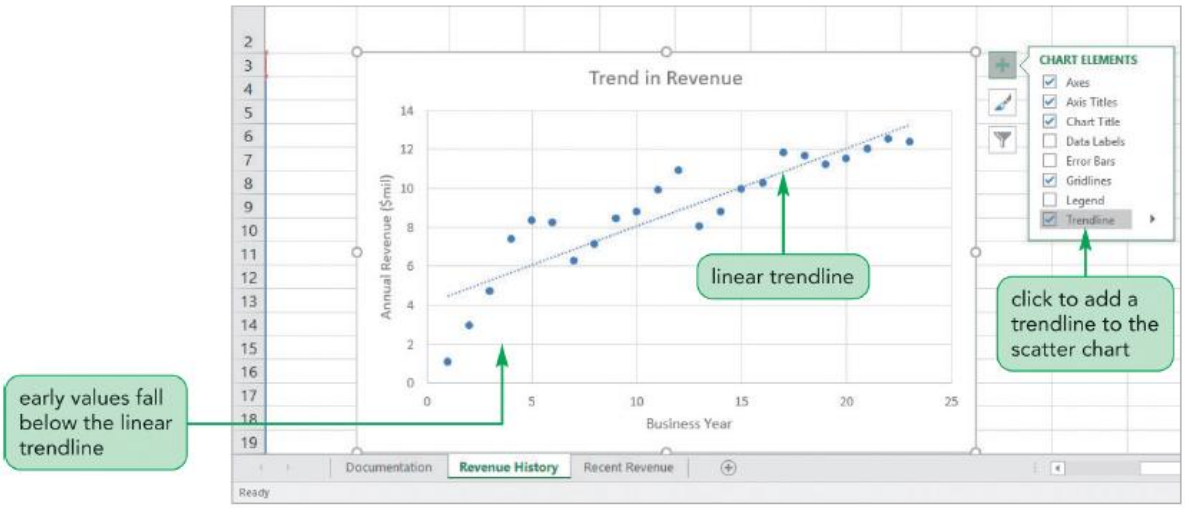
5. Writing a Data Query

- Text files
  - Simple, widely used format for storing raw, unformatted data (text and numbers)
  - Useful for sharing data across software programs and computer systems
- Understanding Text File Formats
  - Use a delimiter (space, comma, or tab) to separate columns of data
  - Use fixed-width text file to start each column at the same location

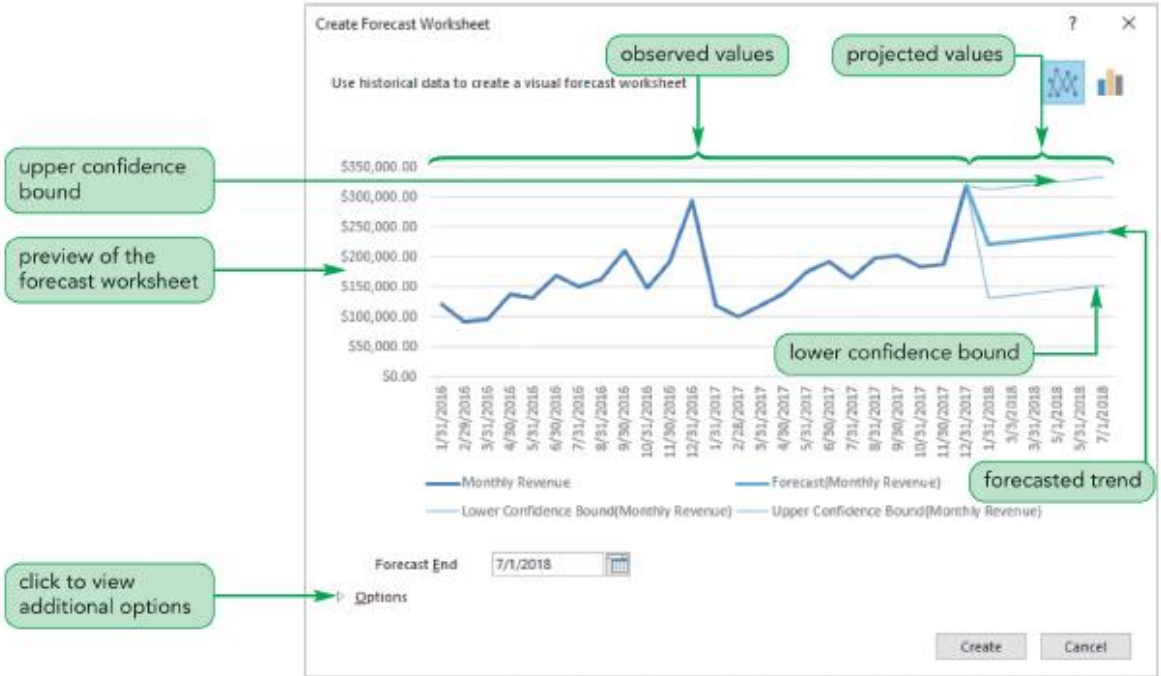


5.1 Charting Trends

- When examining the relationship between two data fields, such as monthly revenue and date of purchase, analysts often want to find trends in the data.
- One way to identify a trend is by adding a trendline to a scatter chart of the data.



5.2 Creating a Forecast Sheet



Summary

- Program design with Structure Charts
- Analysing Data with Excel – Work through Module 11 on your own

6. Practice and Apply

- Discuss program design approaches
- Design modules using structure charts
- Import data from external files
- Complete Tutorial 9 Exercises