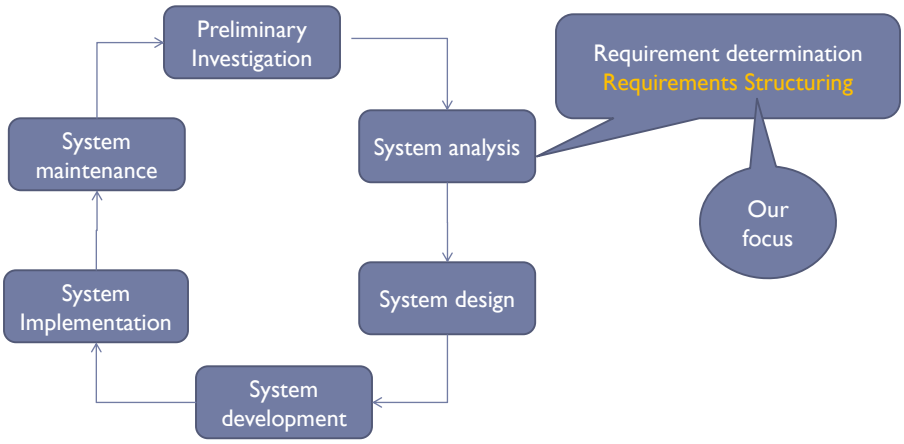
Structured System analysis and Design Tools

1

Scope



2

Requirement analysis

- Requirement analysis phase of the SDLC
  - Two sub phases
    - Requirements determination
    - Requirements structuring
  - Requirements structuring
    - Organize the information into a meaningful representation of the information system that currently exists and of the requirements desired in a replacement system
      - Process modelling
        - Data Flow Diagram
      - Logic modelling
        - Structured English, Decision tables…
      - Data modelling
        - ER diagram

3

Process Modeling

4

Process Modeling: Introduction

- Graphical representation of the functions or processes, that capture, manipulate, store and distribute data between a system and its environment and between components within a system [Dixit and Kumar]
- A common form of process model
  - Data Flow Diagrams (DFD)

5

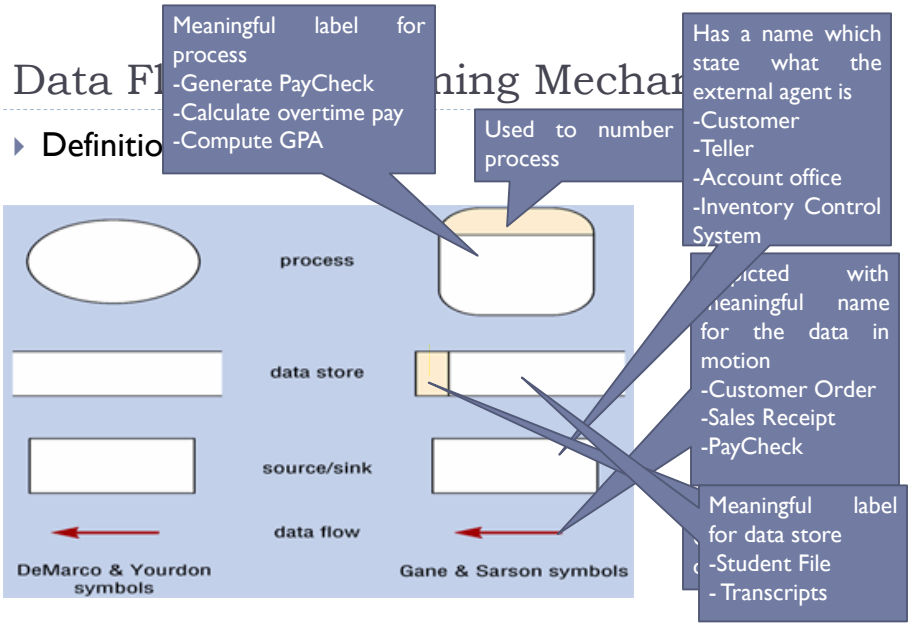
Process Modeling: Introduction…

- Primary deliverables
  - Set of coherent, interrelated data flow diagrams
    - Context Data Flow Diagram
    - DFDs of Current Physical System (adequate detail only)
    - DFDs of current logical system
    - DFDs of new logical system
    - Thorough description of each DFD component
    - Data Dictionary

6

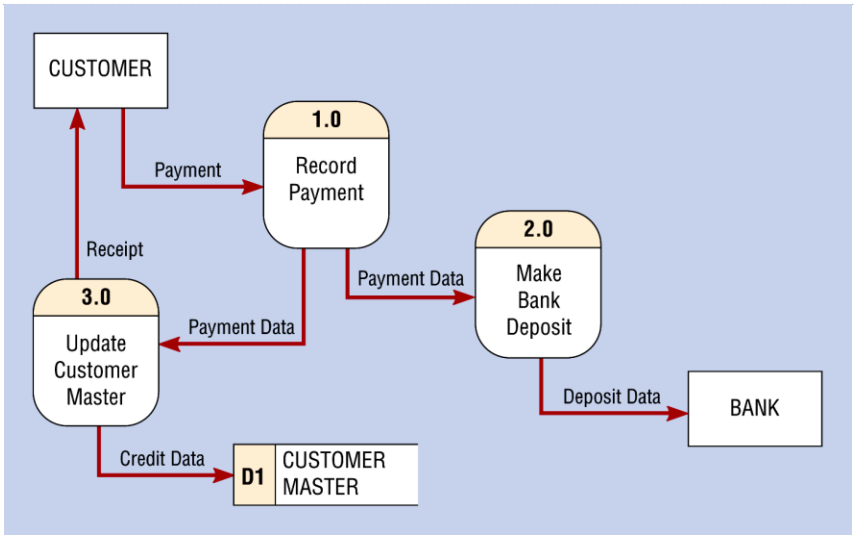
Data Fl ning Mechar

- Definitio



7

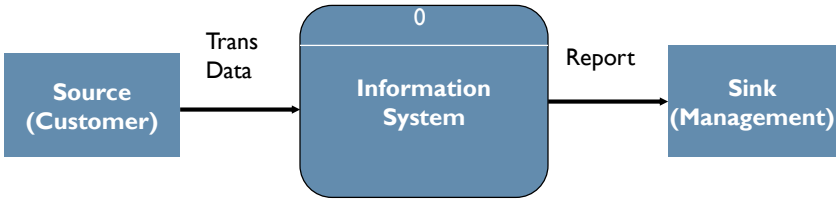
An Example - A Data Flow Diagram for a Banking System



8

An Information System : A Generic View

In general, a system could be viewed as a single Process



There can be multiple sources and sinks!

This generic diagram is called "**Context Diagram**"

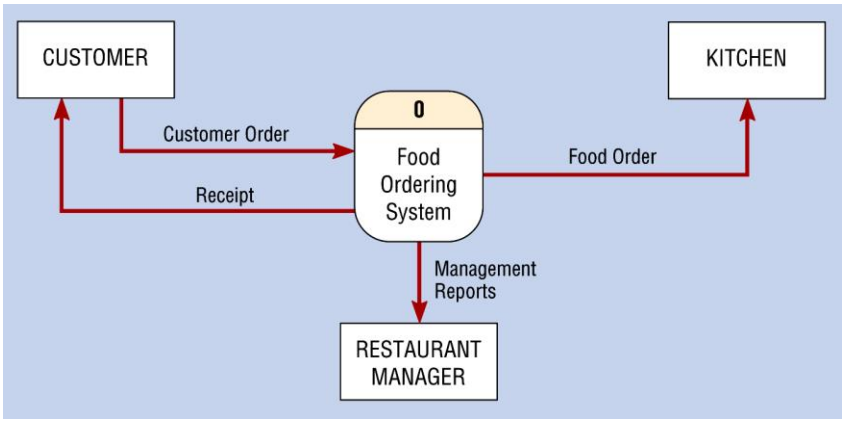
9

A Context Diagram

- An overview of an organizational system that shows the system boundary, sources / sinks that interact with the system, and the major information flows between the entities and the system
- A Context Diagram addresses only one process.
- An example ...

10

An Example - A Context Diagram for a Fast-Food IS

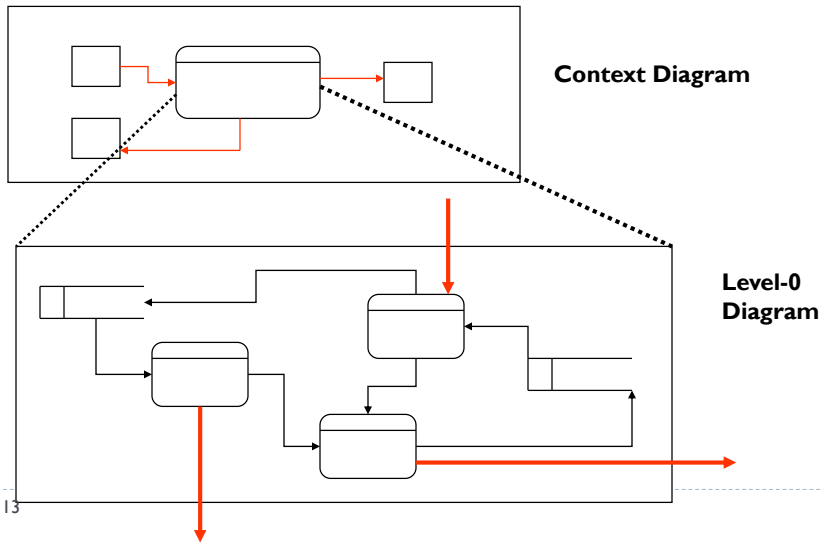


Process Decomposition

- In general, a system could be too complex to understand when viewed as a single Process
- We need a Process Decomposition scheme
  - i.e., to separate a system into its subsystems (sub-processes), which in turn could be further divided into smaller subsystems until the final subsystems become manageable units (i.e., primitive processes!)
- A divide and conquer strategy!!

Functional decomposition is an iterative process of breaking the description or perspective of a system down into finer and finer detail.

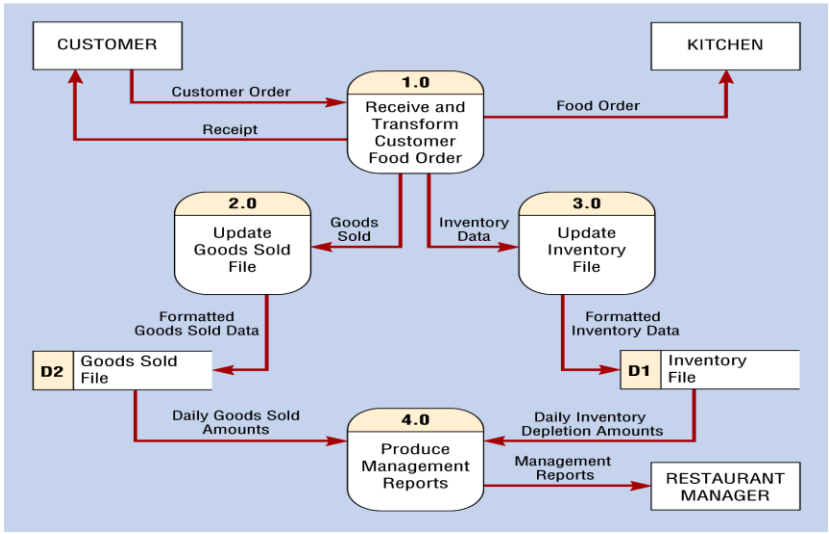
Decomposition Overview



Level-0 Diagram

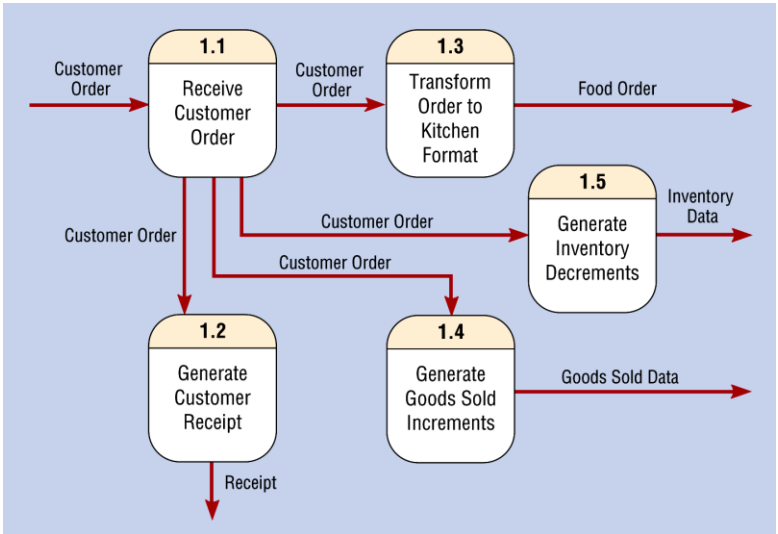
- A DFD that represents the primary functional processes in the system at the highest possible level
- An Example ...

An Example - A decomposed Context Diagram - Level 0 Diagram



15

An Example - A further decomposition A Level-1 Data Flow Diagram



16

Process Decomposition Rules

- Generic Decomposition Rules:
  - A process in a DFD could be either a parent process or a child process, or both.
  - A parent process must have two or more child processes.
  - A child process may further be decomposed into a set of child processes.

17

Three Major Types of Process

- Function Process
  - A function is a set of related activities of the business (e.g., Marketing, Production, etc.)
- Event Process
  - An event process is a logical unit of work that must be completed as a whole. (e.g., Process customer credit verification)
- Primitive Process
  - a primitive process is a discrete, lowest-level activity/task required to complete an event. (e.g., Check the credit card balance)

18

Naming Rules for Processes

- Function Process - use a Noun
- Event Process - Use a general action verb
  - Process Student registration.
  - Respond to ...
  - Generate ...
- Primitive Process - use a strong action verb
  - Validate Student ID
  - Check ...
  - Calculate ...

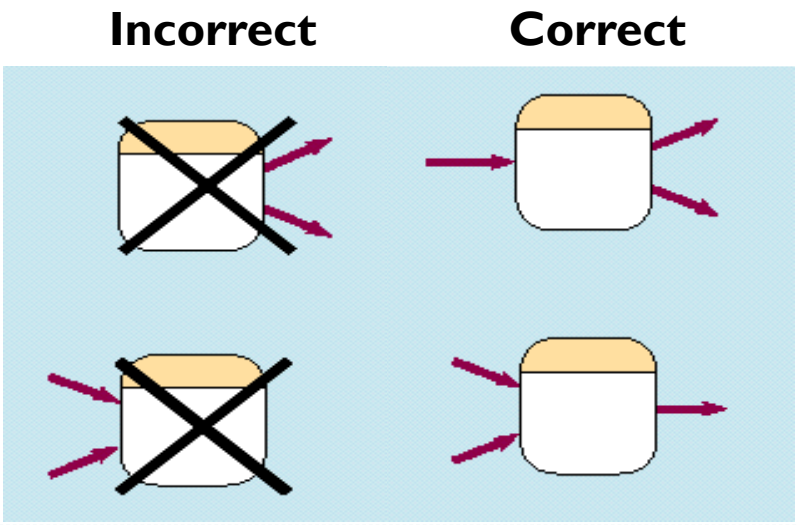
19

Rules for Processes

- No process can have only outputs (a miracle!)
- No process can have only inputs (a black hole!)
- No process can produce outputs with insufficient inputs (a gray hole!)

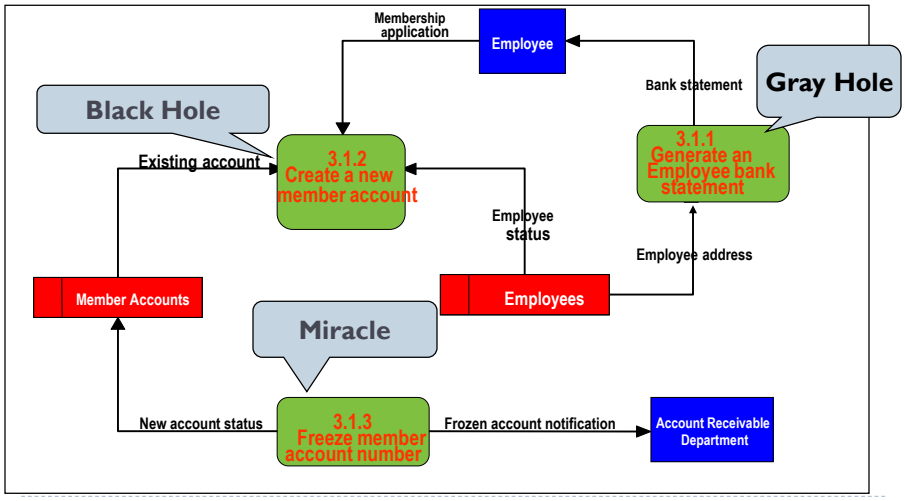
20

Processes in a DFD- Correct vs. Incorrect



21

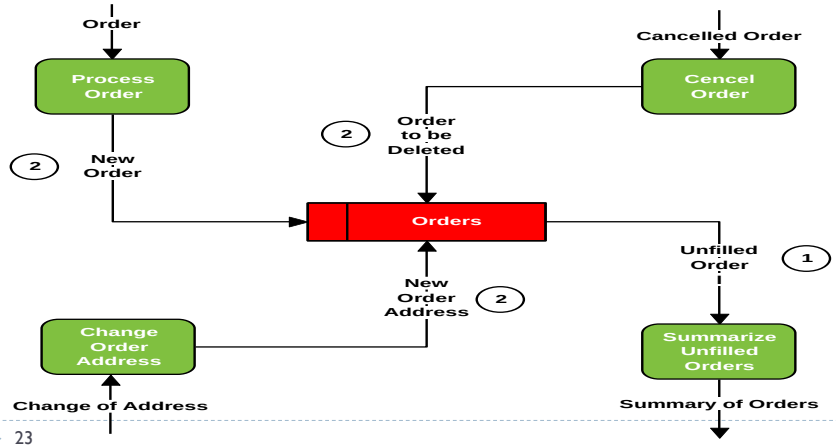
Can You Identify Errors in This Diagram?



22

Basic Concept About Data Flows …

- It has two kinds of flow: a) Inflow to a Data Store (Create/Modify/Delete) b) Outflow from a Data Store (Read)



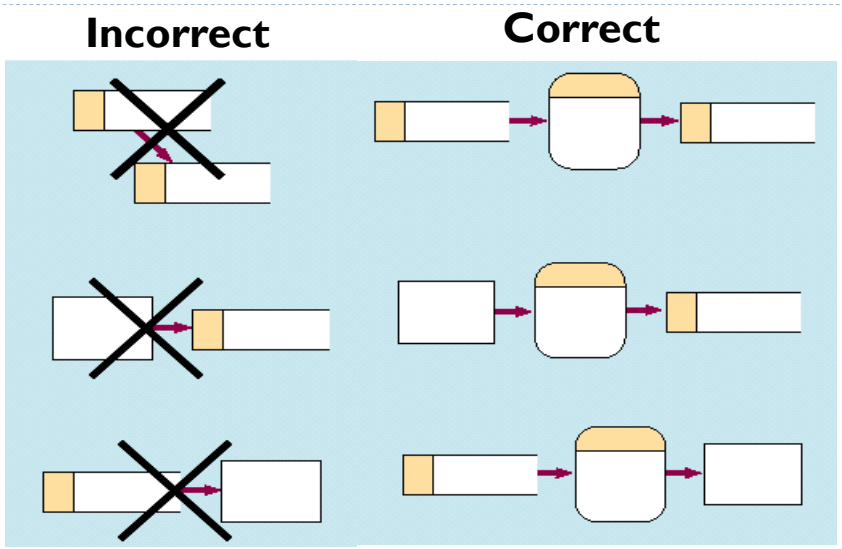
23

Rules for Data Stores

- Data cannot move directly from one data store to another data store
  - it must be moved by a process.
- Data cannot move directly from an outside source to a data store
  - it must be moved by a process.
- Data cannot move directly to an outside sink from a data store
  - it must be moved by a process.
- You need to use a Noun phrase to label each data flow

24

Data Flows in a DFD: Incorrect vs. Correct



25

Naming Rules Data Flow

- Use a *singular noun phrase* for each data flow
  - Ex: customer data, shipping report, …, etc.
- Carry logical meaning only,
  - i.e., *no implication on data form* or *data structure*
- Minimum flow (no data flooding!!)
- Should *never be "Unnamed!!"*
  - otherwise, there might be a modeling error.

26

Naming Scheme for Other DFD Components

- Process (Event)
  - Use an Action Verb Phrase
  - Process member order, Generate bank statement, ...
- External Agent (Sink/Source)
  - Use a singular descriptive noun
  - Ex: Student, Customer, etc.
- Data Store
  - Use a plural descriptive noun (Members, Customers, etc.)
  - Or use a noun + file (Inventory file, Goods sold file)

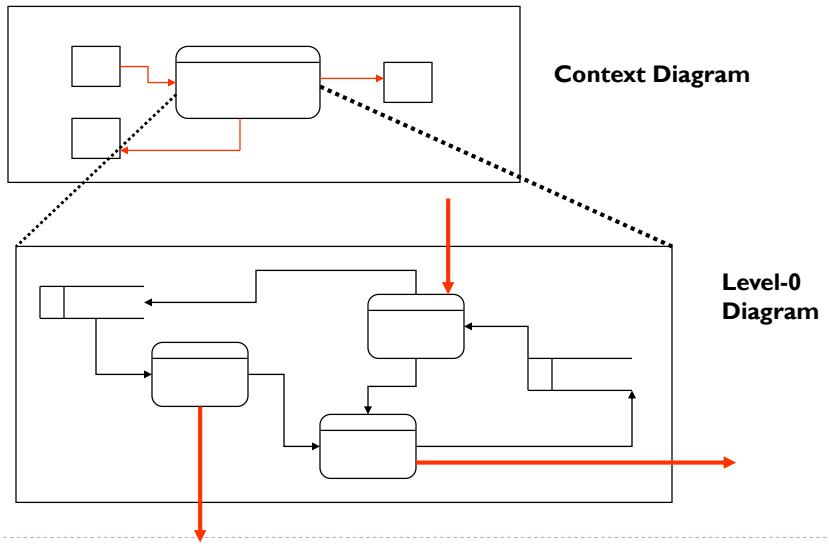
27

Basic Rule in DFD Decomposition

- Balancing Principle
  - the decomposed DFD (i.e., the next lower level DFD) should retain the same number of inputs and outputs from its previous higher level DFD (i.e., No new inputs or outputs when a DFD is decomposed)

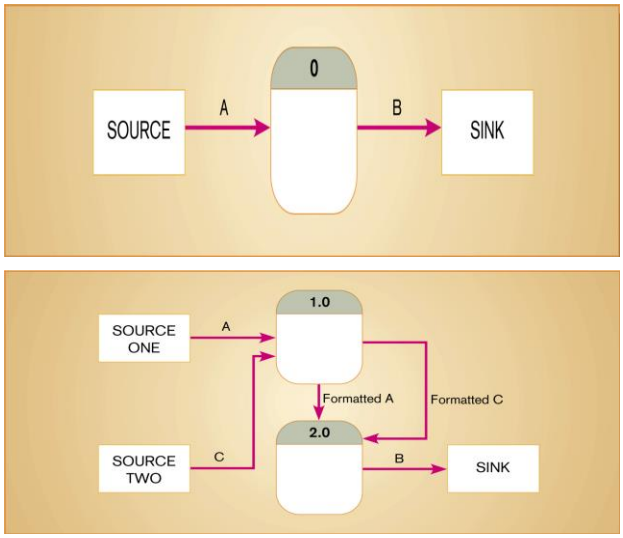
28

Basic Rule in DFD Decomposition…



29

Unbalanced DFD



30

Data Dictionary

- Collection of information of all data elements or contents of databases such as data types, text descriptions etc.
- It makes it easier for user and analyst to use data as well as understand
- Helps to have common knowledge about inputs, outputs, components of a database, and intermediate calculations

31

Data Dictionary…

- Example: Reservation system
  - In reservation system, "passenger" is an data item whose information is available on data dictionary as follows:

Passenger: Passenger_name + Passenger_address

Passenger_name: Passenger_lastname + Passenger_firstname + Passenger_middle_initial

Passenger_address: Local_address + Community_address + Zip_code

Local_address: House_number + street_name + Apartment_number

Community_Address: City_name + State_name

32

Data Dictionary…

- Example: Reservation system
  - In reservation system, "passenger" is an data item whose information is available on data dictionary as follows:

Passenger: Passenger_name + Passenger_address

Passenger_name: Passenger_lastname + Passenger_firstname + Passenger_middle_initial

Passenger_address: Local_address + Community_address + Zip_code

Local_address: House_number + street_name + Apartment_number

Community_Address: City_name + State_name

33

A few examples…

34

Joe's Yard

Joe's builders' suppliers has a shop and a yard. His system is entirely manual. He has a stock list on the wall of his shop, complete with prices. When a builder wants to buy supplies, he goes into the shop and picks the stock items from the list. He writes his order on a duplicate docket and pays Joe, who stamps the docket as paid. The builder takes the duplicate docket and he goes to the yard and hands it to the yard foreman. The yard foreman gets the ordered items from the yard and gives them to the builder. The builder signs the duplicate docket and leaves one copy with the foreman and takes one copy as a receipt. Every week, Joe looks around the yard to see if any of his stock is running low. He rings up the relevant suppliers and reorders stock. He records the order in his order book, which is kept in the yard. The yard foreman takes delivery of the new stock and checks it against what has been ordered. He pays for it on delivery and staples the receipt into the order book. At the end of every month, Joe goes through all the dockets and the order book and produces a financial report for the shareholders.

Let us draw a context level DFD and a level-1 DFD for this system.

35

Context Diagram

- Find the people who send data into the system
- Find the people who get data out of the system.
  - The only data you need is data that is transformed or sent completely out of the system – not data that is handled by an operator within the system.

36

Identify External Entities

- Joe's builders' suppliers has a shop and a yard. His system is entirely manual.
- He has a stock list on the wall of his shop, complete with prices.
- When a **builder** wants to buy supplies, he goes into the shop and picks the stock items from the list.
- He writes his order on a duplicate docket and pays ***Joe***, who stamps the docket as paid.
- The builder takes the duplicate docket and he goes to the yard and hands it to the yard ***foreman***.
- The yard foreman gets the ordered items from the yard and gives them to the builder.
- The builder signs the duplicate docket and leaves one copy with the foreman and takes one copy as a receipt.
- Every week, Joe looks around the yard to see if any of his stock is running low.
- He rings up the relevant **suppliers** and reorders stock.
- He records the order in his order book, which is kept in the yard.
- The yard foreman takes delivery of the new stock and checks it against what has been ordered.
- He pays for it on delivery and staples the receipt into the order book.
- At the end of every month, Joe goes through all the dockets and the order book and produces a financial report for the **shareholders**.

37

Level-1 DFD processes: Identify the verbs

Joe's builders' suppliers has a shop and a yard. His system is entirely manual. He has a stock list on the wall of his shop, complete with prices. When a builder wants to buy supplies, he goes into the shop and picks the stock items from the list. He writes his order on a duplicate docket and pays Joe, who stamps the docket as paid. The builder takes the duplicate docket and he goes to the yard and hands it to the yard foreman. The yard foreman gets the ordered items from the yard and gives them to the builder. The builder signs the duplicate docket and leaves one copy with the foreman and takes one copy as a receipt. Every week, Joe looks around the yard to see if any of his stock is running low. He rings up the relevant suppliers and reorders stock. He records the order in his order book, which is kept in the yard. The yard foreman takes delivery of the new stock and checks it against what has been ordered. He pays for it on delivery and staples the receipt into the order book. At the end of every month, Joe goes through all the dockets and the order book and produces a financial report for the shareholders.

38

Verbs from script

- Buy supplies
- Picks stock items
- Writes order
- Pays joe
- Stamps docket
- Takes docket to yard
- Hands it to foreman
- Gets items
- Gives them to builder
- Builder signs docket
- Takes copy as receipt
- Looks around yard and reorders
- Records order in order book
- Foreman takes delivery – checks
- Foreman pays supplier
- Staples receipt to order book
- Produces financial report

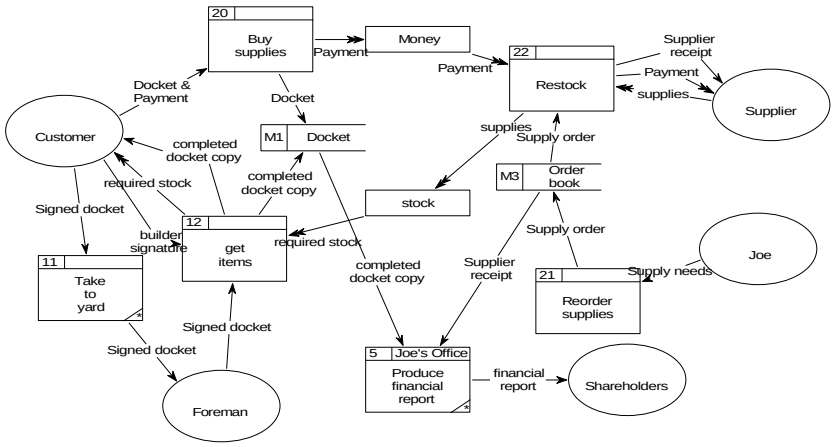
39

Group the processes

- Buy supplies
  - Picks stock items (views list)
  - Writes orders
  - Pays joe
  - Stamps docket
- Customer then
  - Takes docket to yard
  - Hands it to foreman
- Gets items
  - Gives them to builder
  - Builder signs docket
  - Takes copy as receipt
- Joe then
  - Looks around yard and reorders
  - Records order in order book
- Foreman
  - takes delivery – checks
  - Foreman pays supplier
  - Staples receipt to order book
- Joe
  - Produces financial report

40

Level-1 DFD



Lemonade Stand Example

Creating Data Flow Diagrams

Example

The operations of a simple lemonade stand will be used to demonstrate the creation of dataflow diagrams.`

Steps:

1. Create a list of activities
2. Construct Context Level DFD (identifies sources and sink)
3. Construct Level 0 DFD (identifies manageable sub processes)
4. Construct Level 1- n DFD (identifies actual data flows and data stores)

43

Creating Data Flow Diagrams

Example

Think through the activities that take place at a lemonade stand.

1. Create a list of activities

Customer Order
Serve Product
Collect Payment
Produce Product
Store Product

44

Creating Data Flow Diagrams

Example

Also think of the additional activities needed to support the basic activities.

1. Create a list of activities

Customer Order
Serve Product
Collect Payment
Produce Product
Store Product
Order Raw Materials
Pay for Raw Materials
Pay for Labor

45

Creating Data Flow Diagrams

Example

Group these activities in some logical fashion, possibly functional areas.

1. Create a list of activities

Customer Order
Serve Product
Collect Payment

Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

46

Creating Data Flow Diagrams

Example

Create a context level diagram identifying the sources and sinks (users).

Customer Order
Serve Product
Collect Payment

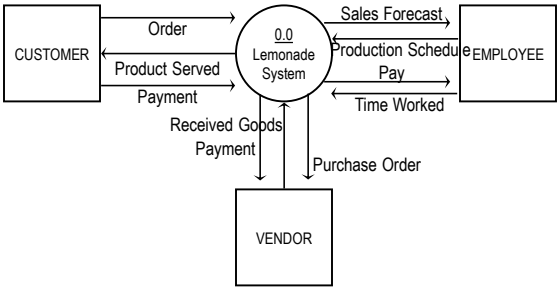
Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

2. Construct Context Level DFD (identifies sources and sink)

Context Level DFD



47

Creating Data Flow Diagrams

Example

Create a level 0 diagram identifying the logical subsystems that may exist.

Customer Order
Serve Product
Collect Payment

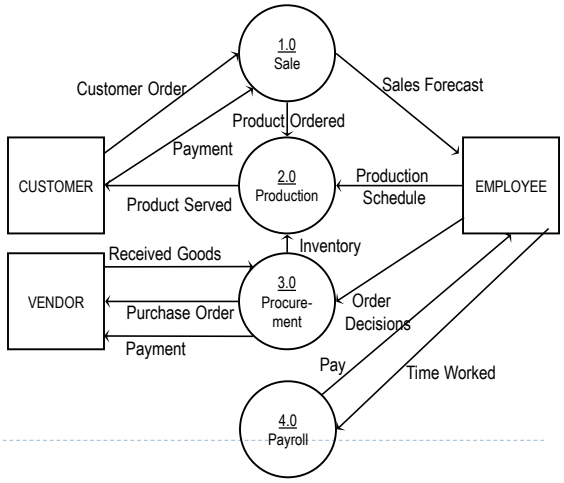
Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

3. Construct Level 0 DFD (identifies manageable sub processes)

Level 0 DFD



48

Creating Data Flow Diagrams

Example

Create a level 1 decomposing the processes in level 0 and identifying data stores.

Customer Order
Serve Product
Collect Payment

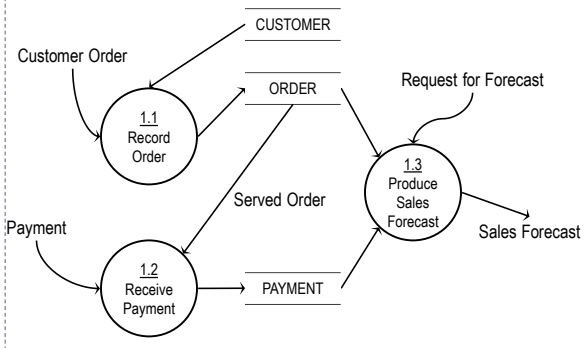
Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

4. Construct Level 1- n DFD
(identifies actual data flows and data stores)

Level 1 DFD



49

Creating Data Flow Diagrams

Example

Create a level 1 decomposing the processes in level 0 and identifying data stores.

Customer Order
Serve Product
Collect Payment

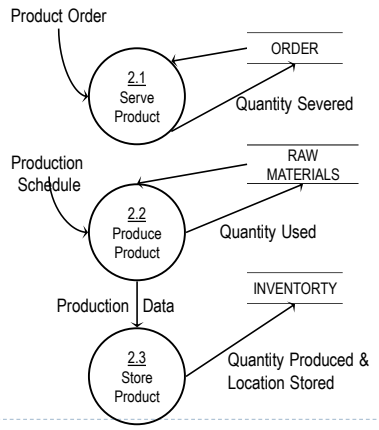
Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

4. Construct Level 1 (continued)

Level 1 DFD



50

Creating Data Flow Diagrams

Example

Create a level 1 decomposing the processes in level 0 and identifying data stores.

Customer Order
Serve Product
Collect Payment

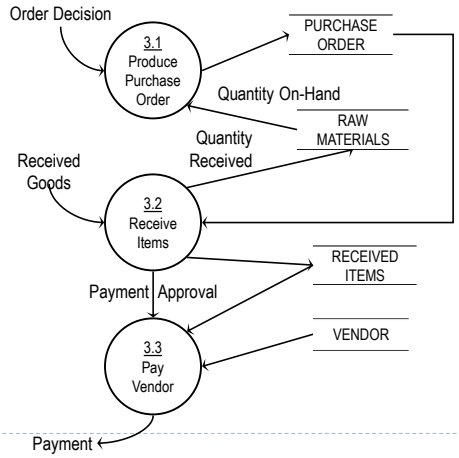
Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

4. Construct Level 1 (continued)

Level 1 DFD



51

Creating Data Flow Diagrams

Example

Create a level 1 decomposing the processes in level 0 and identifying data stores.

Customer Order
Serve Product
Collect Payment

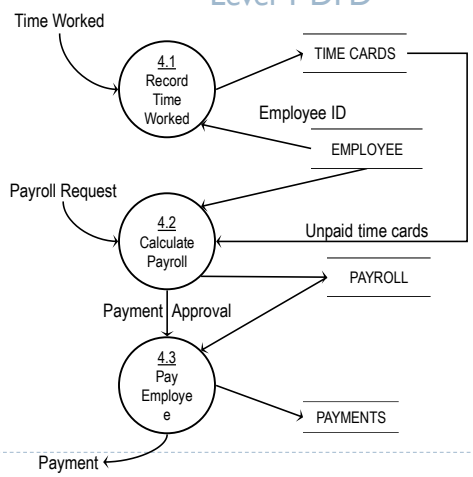
Produce Product
Store Product

Order Raw Materials
Pay for Raw Materials

Pay for Labor

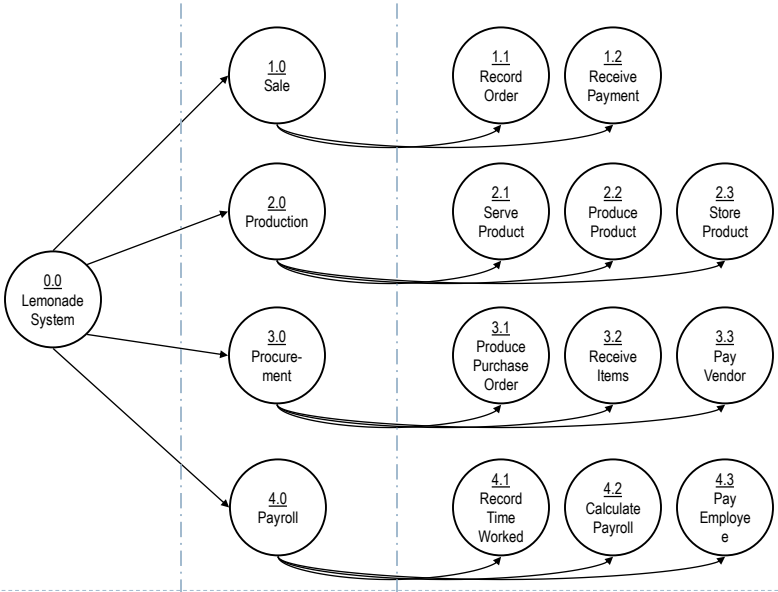
4. Construct Level 1 (continued)

Level 1 DFD



52

Process Decomposition



Context Level | Level 0 | Level 1

Logic modeling

Logic Modeling

- Data flow diagrams do not show the **logic** inside the processes
  - what occurs within a process?
  - How input data is converted into output information
- Logic modeling involves representing internal **structure** and **functionality** of **processes** depicted on a DFD.
- Processes must be **clearly described** before translating them into programming language.
- Logic modeling can also be used to show **when processes** on a DFD occur.
- Logic modeling will be **generic** without taking syntax of a particular programming language

55

Logic Modeling: Deliverables and Outcomes

Each process on the lowest level DFD will be represented by one or more of the following: primitive

- Structured English
- Decision Tables
- Decision Trees
- State-transition diagrams
- Sequence diagrams
- Activity diagrams

56

Modeling Logic with Structured English

Modeling with Structured English

- **Structured English** is a modified form of English used to specify the **logic** of information processes
- Uses a subset of English vocabulary to express process procedures
  - *Action verbs* – read, write, print, move, merge, add, sort
  - *Noun phrases* – name, address
    - No adjectives or adverbs
- No specific standards – each analyst will have his own way
- File and variable names are CAPITALIZED
- Logical comparisons are *spelled out* and not used symbols

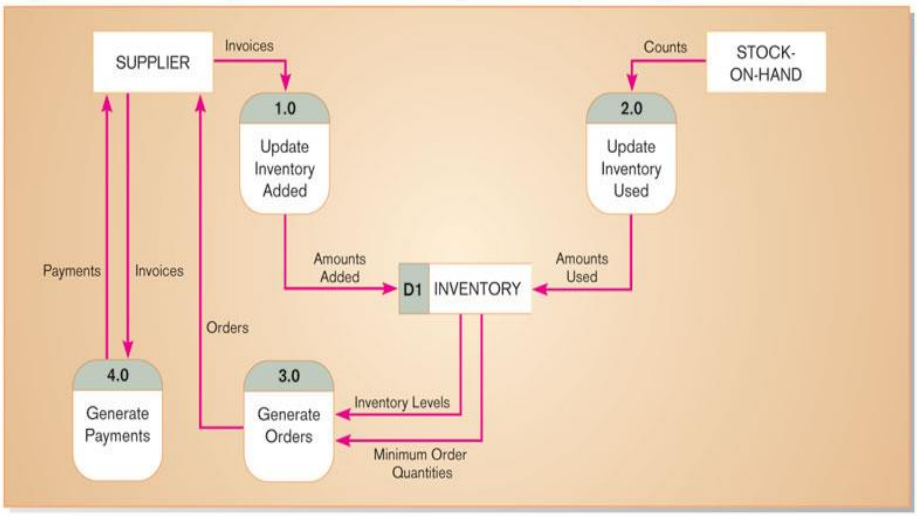
Structured English is used to represent processes in a shorthand manner that is relatively **easy** for users and programmers to read and understand

Modeling with Structured English…

- It is possible to represent all three processes used in structured programming
  - sequence, conditional, repetition
- Sequence
  - no special structure but one statement following another
- Conditional
  - IF THEN ELSE statement; CASE statement
- Repetition
  - DO-UNTIL loops or DO-WHILE loops
- Format of Structured English uses indentation used in programming languages
- Structured English does not initialize variables, open and close files, or find related records in separate files – all are done in later design process

59

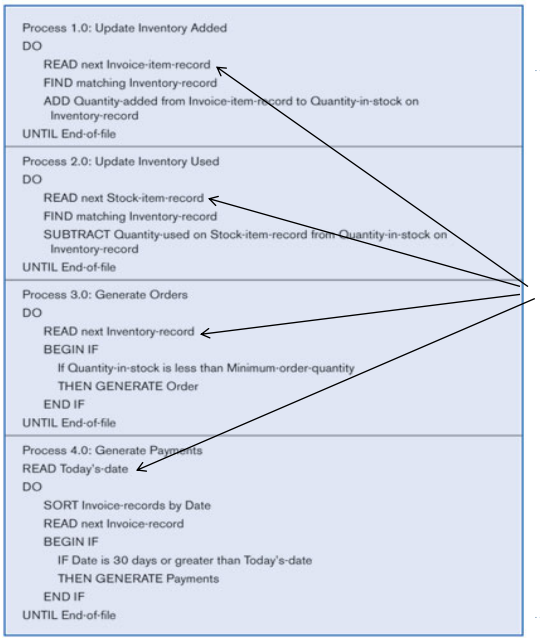
Figure 8-2 Current logical DFD for Hoosier Burger's inventory control system



Courtesy: Modern Systems Analysis and Design, Prentice hall

60

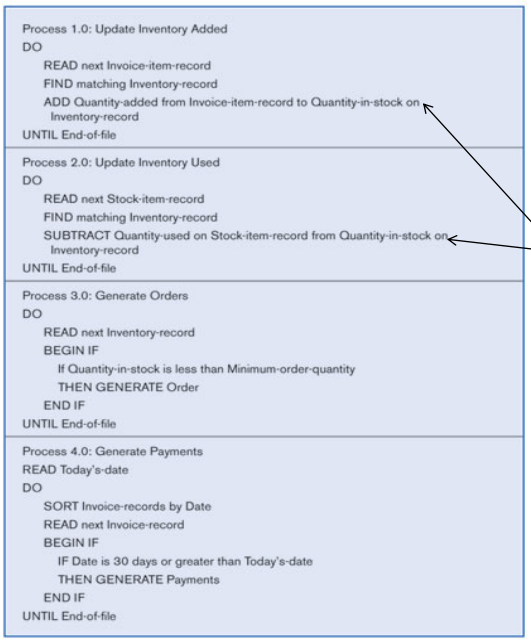
Figure 8-3
Structured English representations of the four processes depicted in Figure 8-2



Structured English is used here to describe input and output.

61

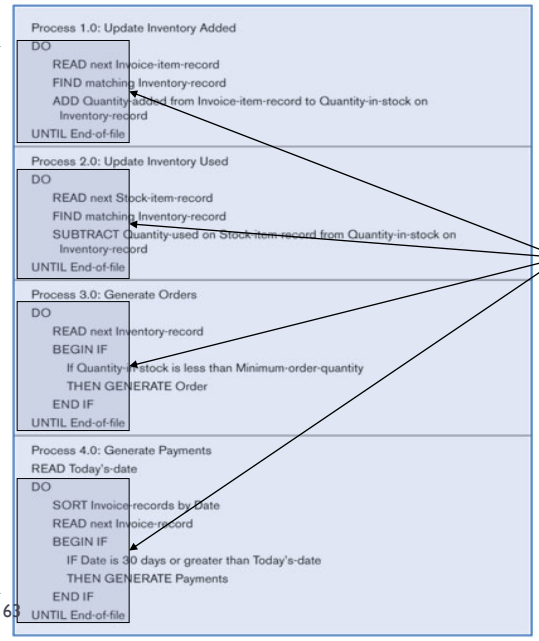
Figure 8-3
Structured English representations of the four processes depicted in Figure 8-2



Structured English is used here to describe arithmetic operations.

62

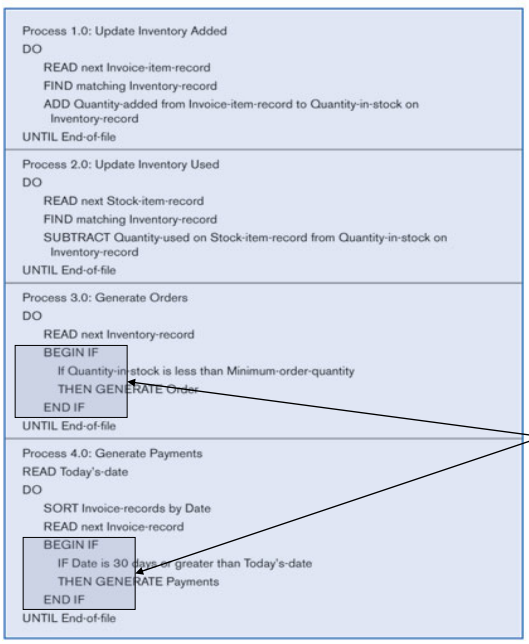
Figure 8-3
Structured English representations of the four processes depicted in Figure 8-2



Structured English is used here to describe repetition.

63

Figure 8-3
Structured English representations of the four processes depicted in Figure 8-2



Structured English is used here to describe decisions.

64

Figure 8-3
Structured English representations of the four processes depicted in Figure 8-2

```
Process 1.0: Update Inventory Added
DO
    READ next Invoice-item-record
    FIND matching Inventory-record
    ADD Quantity-added from Invoice-item-record to Quantity-in-stock on
      Inventory-record
UNTIL End-of-file

Process 2.0: Update Inventory Used
DO
    READ next Stock-item-record
    FIND matching Inventory-record
    SUBTRACT Quantity-used on Stock-item-record from Quantity-in-stock on
      Inventory-record
UNTIL End-of-file

Process 3.0: Generate Orders
DO
    READ next Inventory-record
    BEGIN IF
      If Quantity-in-stock is less than Minimum-order-quantity
      THEN GENERATE Order
    END IF
UNTIL End-of-file

Process 4.0: Generate Payments
READ Today's-date
DO
    SORT Invoice-records by Date
    READ next Invoice-record
    BEGIN IF
      IF Date is 30 days or greater than Today's-date
      THEN GENERATE Payments
    END IF
UNTIL End-of-file
```

Structured English is used here to describe invoking other processes.

65

Modeling Logic with Decision Tables

66

Modeling Logic with Decision Tables

- Structured English is not good to represent *complicated logic* (having several different conditions) as it becomes difficult to understand
- Decision table
  - A matrix representation of the logic of a decision
  - Specifies all the possible conditions and the resulting actions in a tabular form
  - Best used for complicated decision logic
- Parts of a Decision Table

1. Condition stubs
   - Lists condition relevant to decision
2. Action stubs
   - Actions that result from a given set of conditions
3. Rules
   - Specify which actions are to be followed for a given set of conditions

- Indifferent Condition
  - Condition whose value *does not affect* which action is taken for two or more rules

67

Procedure for Creating Decision Tables

- Name the conditions and values each condition can assume
  - some conditions values will be just "yes" or "no" and some may have many values (called an extended entry)
- Name all possible actions that can occur
- List all possible rules
  - Create exhaustive set of rules – every possible combination of conditions must be represented
  - Some rules may be *redundant* or make *no sense* that can be altered later
  - **Number of rules = number of values for *condition 1* X number of values for *condition 2* X …..X number of values for *condition n***
- Define the actions for each rule
  - If an action doesn't make sense create an "**impossible**" row for that action
  - If the action is not known place a ? for that rule
- Simplify the table
  - Remove any rules with **impossible** actions

68

Decision Table

Figure 8-4 Complete decision table for payroll system example

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Note: for salaried employees the action stub chosen will always be the same…therefore hours worked is an *indifferent condition*

69

Reduced Decision Table

Figure 8-5 Reduced decision table for payroll system example

Conditions/ Courses of Action	Rules			
	1	2	3	4
Employee type	S	H	H	H
Hours worked	–	<40	40	>40
Pay base salary	X			
Calculate hourly wage		X	X	X
Calculate overtime				X
Produce Absence Report		X		

Because of indifferent condition, the complete decision table can be reduced to one with fewer rules

70

Procedure for Creating Decision Tables

- Decision tables can also be used to specify additional decision-related information:
  - If actions for a rule are more complicated and can't be conveyed in one or two lines of text (or)
  - If some conditions depend on other conditions (nested conditions)
    - use separate, **linked decision** table by writing "Perform Table B" as action in the action stub
    - Table B could contain an action stub that returns to the original table
- Use **numbers** to indicate *sequence* rather than just Xs where rules and action stub intersect
- Decision tables are **compact**
  - pack a lot of information into a small table
- Decision tables allow you to check for
  - **completeness**, **consistency**, and **redundancy** of *logic*

71

Modeling Logic with Decision Trees

72

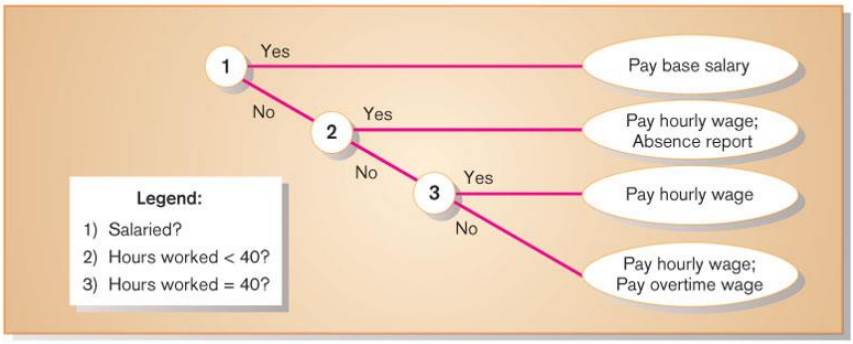
Modeling Logic with Decision Trees

- A **decision tree** is a graphical representation of a decision situation
- Decision situation points (**nodes**) are connected together by **arcs** and terminate in **ovals**
- Main components
  - **Decision points** represented by **nodes**
  - **Actions** represented by **ovals**
  - Particular choices from a decision point represented by **arcs**
- To read a decision tree – begin at root node on far **left**
- Each **node** is numbered and each number corresponds to a **choice**
- Choices are spelled out in a **legend**
- From each node there are *at least two paths* leading to next step – another *decision point* or an *action*
- All possible **actions** are listed on the far **right** in **leaf nodes**
- Each **rule** is represented by tracing a series of *paths* from **root** node to the next node and so on until an **action** *oval* is reached

73

Decision tree representation of salary decision

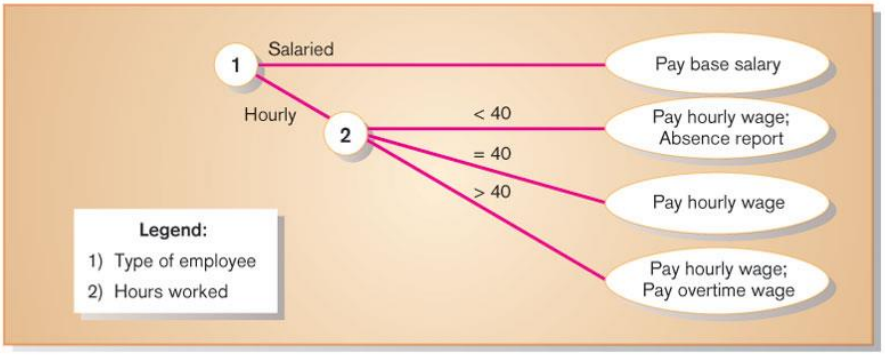
Figure 8-9 Decision tree representation of the decision logic in the decision tables in Figures 8-4 and 8-5, with only two choices per decision point



74

Alternative decision tree representation of salary decision

Figure 8-10 Decision tree representation of the decision logic in the decision tables in Figure 8-4 and 8-5, with multiple choices per decision point



75

Model the logic using Decision table and decision tree for the following problem

Consider the following excerpt from an actual requirements document:

If the customer account is billed using a fixed rate method, a minimum monthly charge is assessed for consumption of less than 100 kwh. Otherwise, apply a schedule A rate structure. However, if the account is billed using a variable rate method, a schedule A rate structure will apply to consumption below 100 kwh, with additional consumption billed according to schedule B.

[taken from *Software Engineering: A Practitioner's Approach* by Roger Pressman]

77

Model the logic using Decision table and decision tree for the following problem

Consider the following excerpt from an actual requirements document:

At Christmas, a company pays a gift of money to some of its employees. To be eligible for the gift, an employee must have worked for the company for at least six months. Managers get $500 and other employees get @300 for their first Christmas with the company and $500 thereafter.

References

- Books
  - Structured System Analysis and Design, University Science Press
  - Modern Systems Analysis and Design, Prentice Hall