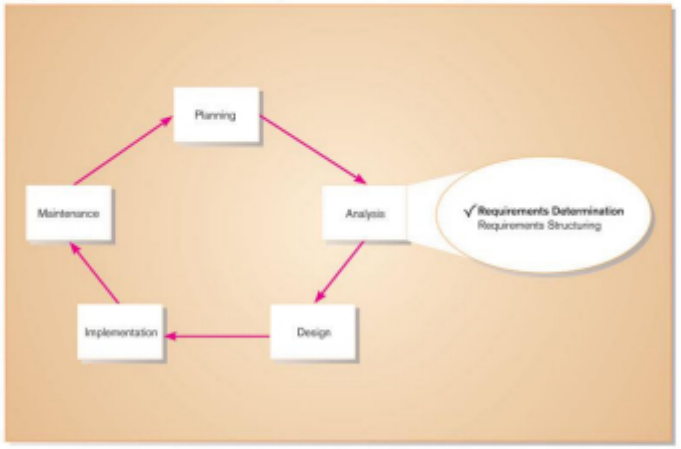
Analysis

3.1. Determining System Requirements

Performing Requirements Determination

Figure 6-1 Systems development life cycle with analysis phase highlighted



Performing Requirements Determination

- Information on what system should do may be collected from:
  - Users
  - Reports
  - Forms
  - Procedures

Performing Requirements Determination

- Characteristics for gathering requirements
  - Impertinence
    - Question everything
  - Impartiality
    - Find the best organizational solution
  - Relaxation of constraints, assume anything is possible
  - Attention to detail, every fact must fit with every other fact
  - Reframing
    - View the organization in new ways,

Deliverables and Outcomes

- Types of deliverables:
  - From interviews and observations - interview transcript observation notes, meeting minutes
  - From existing written documents - mission and strategy statements, business forms, procedure manuals, job descriptions, training manuals, system documentation, flowcharts
  - From computerized sources – Joint Application Design session results, CASE repositories, reports from existing systems, displays and reports from system prototype.

Deliverables and Outcomes

- **Types of deliverables:**
  - Information collected from users
  - Existing documents and files
  - Computer-based information
  - Understanding of organizational components
    - Business objective
    - Information needs
    - Rules of data processing
    - Key events

Traditional Methods for Determining Requirements

- **Interviewing and Listening**
  - Gather facts, opinions and speculations
  - Observe body language and emotions
  - Guidelines
    - Plan
      - Checklist
      - Appointment
    - Be neutral
    - Listen
    - Seek a diverse view

Traditional Methods for Determining Requirements

- Interviewing (Continued)
  - Interview Questions
    - Open-Ended
      - No pre-specified answers
    - Close-Ended
      - Respondent is asked to choose from a set of specified responses
  - Additional Guidelines
    - Do not phrase questions in ways that imply a wrong or right answer
    - Listen very carefully to what is being said
    - Type up notes within 48 hours
    - Do not set expectations about the new system

Traditional Methods for Determining Requirements

- Administering Questionnaires
  - More cost-effective than interviews
  - Choosing respondents
    - Should be representative of all users
    - Types of samples
      - Convenient, local site.
      - Random sample
      - Purposeful sample, people who satisfy certain criteria.
      - Stratified sample, random set of people from many hierarchical levels.

Traditional Methods for Determining Requirements

- Questionnaires
  - Design
    - Mostly closed-ended questions
    - Can be administered over the phone or in person
  - Vs. Interviews
    - Interviews cost more but yield more information
    - Questionnaires are more cost-effective
    - See table 7-4 for a complete comparison

Traditional Methods for Determining Requirements

- Interviewing Groups
- Advantages
  - More effective use of time
  - Enables people to hear opinions of others and to agree or disagree
- Disadvantages
  - Difficulty in scheduling
- Nominal Group Technique (NGT)
- A facilitated process that supports idea generation by groups.
  - Process
    - Members come together as a group, but initially work separately.
    - Each person writes ideas.
    - Facilitator reads ideas out loud, and they are written on a blackboard or flipchart.
    - Group openly discusses the ideas for clarification.
    - Ideas are prioritized, combined, selected, reduced.

Traditional Methods for Determining Requirements

- **Directly Observing Users**
  - Watching users do their jobs
  - Obtaining more firsthand and objective measures of employee interaction with information systems.
  - Can cause people to change their normal operating behavior.
  - Time-consuming and limited time to observe.

Analyzing Procedures and Other Documents

- Types of information to be discovered when analyzing a document:
  - Problems with existing system
  - Opportunity to meet new need
  - Organizational direction
  - Names of key individuals
  - Values of organization
  - Special information processing circumstances
  - Reasons for current system design
  - Rules for processing data

Analyzing Procedures and Other Documents

Four types of useful documents to SA:

- Written work procedures
  - For an individual or work group.
  - Describes how a particular job or task is performed.
  - Includes data and information used and created in the process
- Business form
  - Explicitly indicate data flow in or out of a system
- Report generated by current systems
  - Enables the analyst to work backwards from the report to the data that generated it
- Description of current information system, how they were designed and how they work

Forma and Informal system

- Formal Systems:
  - the official way a system works as described in organizational documentation (i.e. work procedure).
- Informal Systems:
  - the way a system actually works (i.e. interviews, observations).

Modern Methods for Determining Requirements

- Joint Application Design (JAD)
  - Brings together key users, managers and systems analysts
  - Purpose: collect system requirements simultaneously from key people
  - Conducted off-site
- Prototyping
  - Repetitive process
  - Basic version of system is built
  - Refine understanding of system requirements in concrete terms.
  - Goal: to develop concrete specifications for ultimate system

Joint Application Design (JAD)

- Intensive group-oriented requirements determination technique.
- Team members meet in isolation for an extended period of time.
- Highly focused.
- Resource intensive.
- Started by IBM in 1970s.

Prototyping

- Quickly converts requirements to working version of system.
- Once the user sees requirements converted to system, will ask for modifications or will generate additional requests.
- Most useful when:
  - User requests are not clear
  - Few users are involved in the system
  - Designs are complex and require concrete form
  - History of communication problems between analysts and users
  - Tools are readily available to build prototype

Business Process Reengineering (BPR)

- Search for and implementation of radical change in business processes to achieve breakthrough improvements in products and services
- Goals
  - Reorganize complete flow of data in major sections of an organization.
  - Eliminate unnecessary steps.
  - Become more responsive to future change.
  - Combine steps

Business Process Reengineering (BPR)

Identification of processes to Reengineer

- Key business processes
  - Set of activities designed to produce specific output for a particular customer or market
  - Focused on customers and outcome
  - Key business process includes all activities of design, build, deliver and support a product.

Business Process Reengineering (BPR)

- Identify specific activities that can be improved through BPR, once it have been identified, Information Technology must be applied to radically improve business process
- Disruptive technologies are technologies that enable the breaking of long-held business rules that inhibit organizations from making radical business changes

Requirements determining using Agile Methodologies

- **Continual user involvement**
  - Replace traditional SDLC waterfall with iterative analyze – design – code – test cycle
- **Agile usage-centered design**
  - Focuses on user goals, roles, and tasks
  - Gather a group of people all stakeholders in one room.
  - Give everyone a chance to talk about current and new system.
  - Determine user roles and goals
  - Determine task needs to be completed to achieve the goal.
  - Task cards will be grouped together based on similarity.
  - For each task, list steps that are necessary to complete the step.
  - Treat each set of tasks to be supported by a single aspect of user interface (partition task)
  - Prototype and refine the prototype
- **The Planning Game**
  - Based on eXtreme programming
  - Exploration, steering, commitment

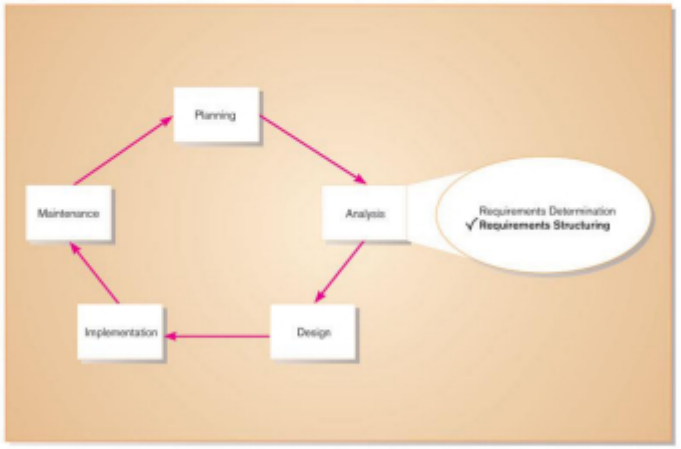
Agile Usage-Centered Design Steps

- Gather group of programmers, analysts, users, testers, facilitator.
- Document complaints of current system.
- Determine important user roles.
- Determine, prioritize, and describe tasks for each user role.
- Group similar tasks into interaction contexts.
- Associate each interaction context with a user interface for the system, and prototype the interaction context.
- Step through and modify the prototype.

3.2. Structuring System Process Requirements

Process Modeling

Figure 7-1 Systems development life cycle with the analysis phase highlighted



Process Modeling

- Graphically represent the processes that capture, manipulate, store and distribute data between a system and its environment and among system components
- Data flow diagrams (DFD)
  - Graphically illustrate movement of data between external entities and the processes and data stores within a system

Process Modeling

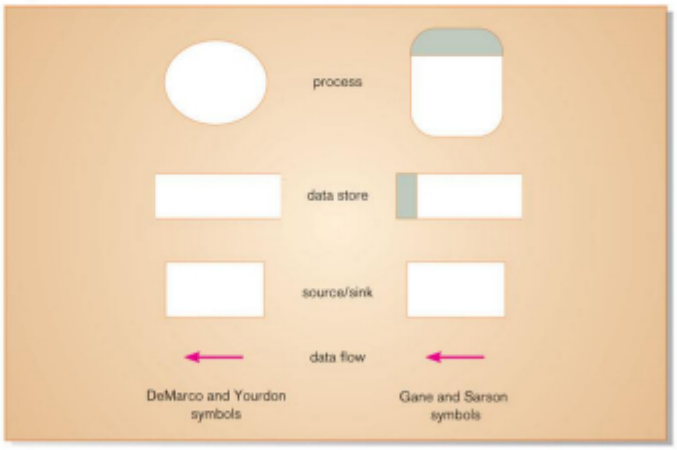
- Modeling a system's process
  - Utilize information gathered during requirements determination and organized it into meaningful representation.
  - In addition you must also model the process logic and timing
  - And structure of data within the system

Deliverables for Process Modeling

- Deliverables and Outcomes
  - Set of coherent, interrelated data flow diagrams, such diagrams are:
    - Context data flow diagram (DFD)
      - Scope of system
    - DFDs of current system
      - Enables analysts to understand current system
      - it shows what data processing functions performed by the current system
    - DFDs of new logical system
      - Show data flows, structure and functional requirements of new system
    - Description of each DFD component, entries for all of the objects included in all diagrams (in data dictionary or CASE repository)

Definitions and Symbols

Figure 7-2 Comparison of DeMarco and Yourdon and Gane and Sarson DFD symbol sets



Data Flow Diagramming Mechanics

- Four symbols are used in DFD
  - process as an oval.
  - data store as a rectangle
  - source/sink as s square
  - data flow as an arrow
- Two different standard sets can be used
- DeMarco and Yourdan
- Gane and Sarson

Data Flow Diagramming Mechanics

- Data Flow
  - Depicts data that are in motion and moving as a unit from one place to another in the system.
  - Drawn as an arrow
  - Select a meaningful name to represent the data

Data Flow Diagramming Mechanics

- Data Store
  - Depicts data at rest
  - May represent data in
    - File folder
    - Computer-based file
    - Notebook
  - The name of the store as well as the number are recorded in between lines

Data Flow Diagramming Mechanics

- Process
  - Depicts work or action performed on data so that they are transformed, stored or distributed
  - Number of process as well as name are recorded

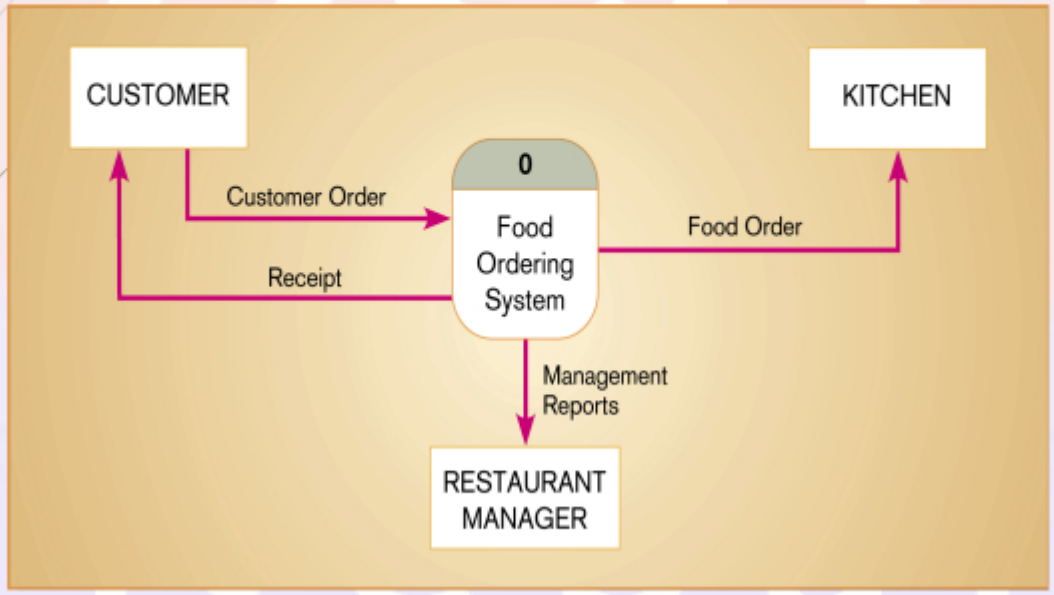
Data Flow Diagramming Mechanics

- Source/Sink
  - Depicts the origin and/or destination of the data
  - Sometimes referred to as an external entity
  - Drawn as a square symbols

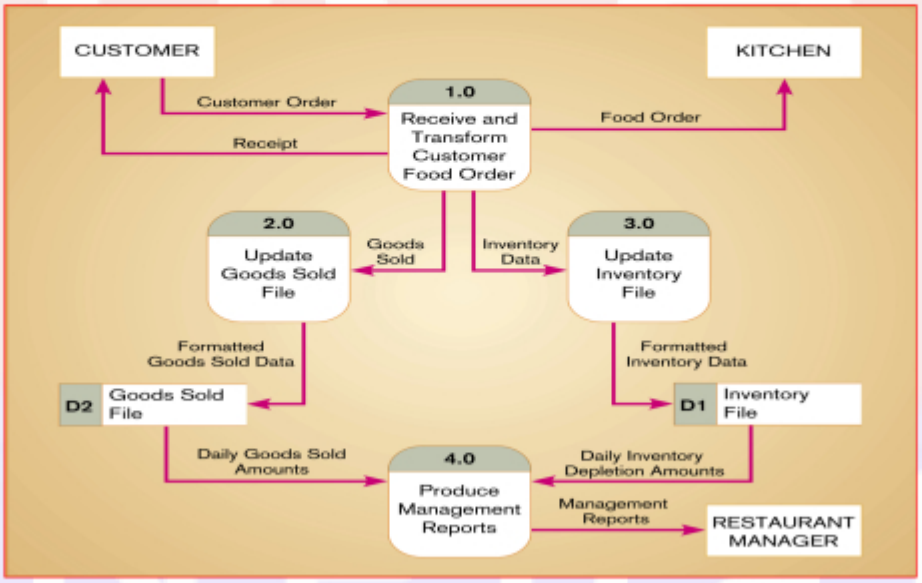
Developing DFDs: An Example

- Hoosier Burger's automated food ordering system
- Context Diagram (level 0 DFD) contains one process, no data stores, four data flows, and three sources/sinks
- Next step is to expand the context diagram to show the breakdown of processes level-1 DFD

Context diagram of Hoosier Burger's food ordering system

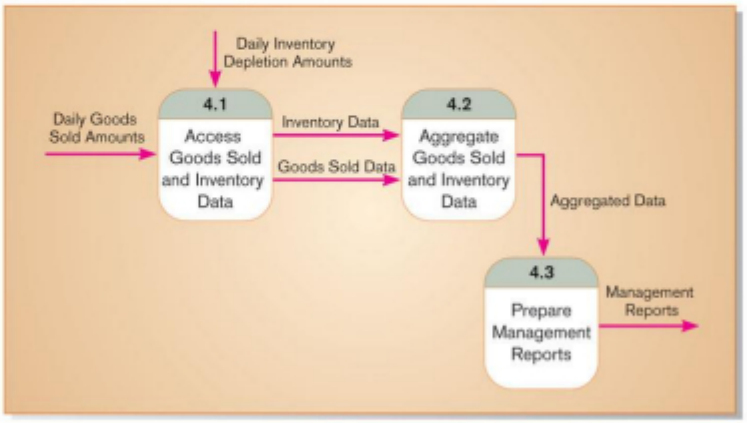


Level-1 DFD of Hoosier Burger's food ordering system



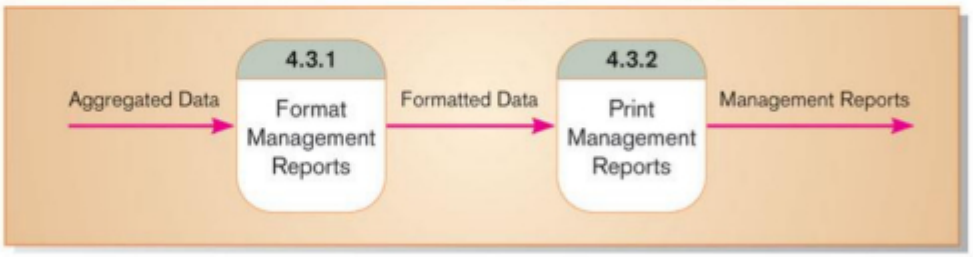
Level-2 DFD

Figure 7-8 Level-1 diagram showing the decomposition of Process 4.0 from the level-0 diagram for Hoosier Burger's food ordering system



Level-*n* DFD

Figure 7-9 Level-2 diagram showing the decomposition of Process 4.3 from the level-1 diagram for Hoosier Burger's food ordering system



Data Flow Diagramming Rules

- Basic rules that apply to all DFDs
  - Inputs to a process are always different than outputs
  - Objects always have a unique name
    - In order to keep the diagram uncluttered, you can repeat data stores and sources/sinks on a diagram
    - Every process has a unique name.

Data Flow Diagramming Rules

Process

- No process can have only outputs (a miracle)
- No process can have only inputs (black hole)
- No process can have partial inputs but complete output (grey hole)
- A process has a verb phrase label

Data Store

- Data cannot be moved directly from one store to another
- Data cannot move directly from an outside source to a data store
- Data cannot move directly from a data store to a data sink
- Data store has a noun phrase label

Data Flow Diagramming Rules

Source/Sink

- Data cannot move directly from a source to a sink
- A source/sink has a noun phrase label

Data Flow

- A data flow has only one direction of flow between symbols
- A fork means that exactly the same data goes from a common location to two or more processes, data stores or sources/sinks

Decomposition of DFDs

- Functional decomposition, Is an iterative process of breaking a system description down into finer and finer detail.
  - Creates a set of charts in which one process on a given chart is explained in greater detail on another chart.
  - Continues until no sub-process can logically be broken down any further.
  - Lowest level is called a primitive DFD
- Level-N Diagrams
  - A DFD that is the result of *n* nested decompositions of a series of sub processes from a process on a level-0 diagram

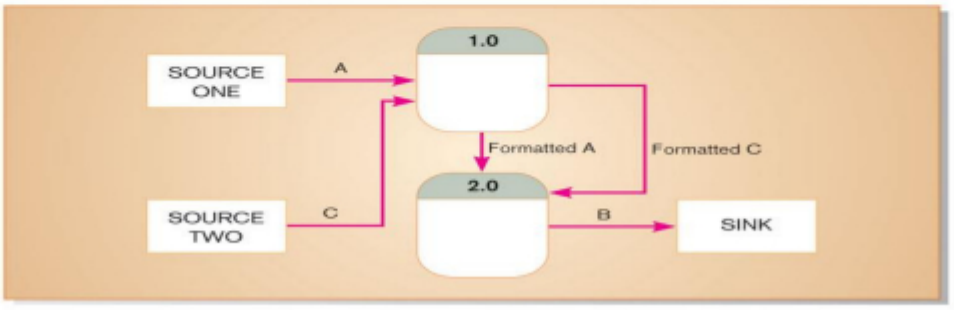
Balancing DFDs

- Conservation Principle:
  - conserve inputs and outputs to a process at the next level of decomposition.
- Balancing:
  - conservation of inputs and outputs to a data flow diagram process when that process is decomposed to a lower level.
- Balanced means:
  - Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD
  - Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD

Balancing DFDs

Figure 7-10a An unbalanced set of data flow diagrams - Context diagram

Figure 7-10b An unbalanced set of data flow diagrams - Level-0 diagram



Balancing DFDs

- Data flow splitting is when a composite data flow at a higher level is split and different parts go to different processes in the lower level DFD.
- The DFD remains balanced because the same data is involved, but split into two parts.

Balancing DFDs

Figure 7-11a Example of data flow splitting - Composite data flow

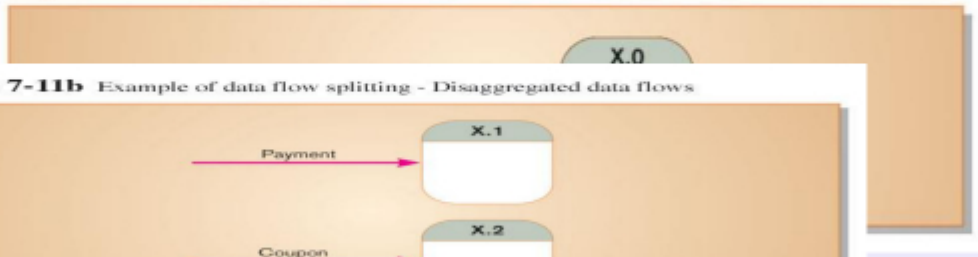


Figure 7-11b Example of data flow splitting - Disaggregated data flows

Guidelines for Drawing DFDs

- Completeness
  - DFD must include all components necessary for system
  - Each component must be fully described in the project dictionary or CASE repository
- Consistency
  - The extent to which information contained on one level of a set of nested DFDs is also included on other levels

Guidelines for Drawing DFDs

- Timing
  - Time is not represented well on DFDs
  - Best to draw DFDs as if the system has never started and will never stop.
- Iterative Development
  - Analyst should expect to redraw diagram several times before reaching the closest approximation to the system being modeled (How many?)

Guidelines for Drawing DFDs

- Rules for stopping decomposition
  - When each process has been reduced to a single decision, calculation or database operation
  - When each data store represents data about a single entity
  - When the system user does not care to see any more detail

Guidelines for Drawing DFDs

- Rules for stopping decomposition
  - When every data flow does not need to be split further to show that data are handled in various ways
  - When you believe that you have shown each business form or transaction, on-line display and report as a single data flow
  - When you believe that there is a separate process for each choice on all lowest-level menu options

Logic Modeling

- Data flow diagrams do not show the logic inside the processes
- Logic modeling involves representing internal structure and functionality of processes depicted on a DFD
- Logic modeling can also be used to show when processes on a DFD occur

Logic Modeling

- **Deliverables and Outcomes**
- Structured English representation of process logic.
- Decision Tables representation.
- Sequence diagram.
- Activity diagram
  - Structured English
  - Decision Tables
  - Decision Trees
  - State-transition diagrams
  - Sequence diagrams
  - Activity diagrams

Modeling Logic with Decision Tables

- A matrix representation of the logic of a decision which Specifies the possible conditions and the resulting actions
- Best used for complicated decision logic

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				

Modeling Logic with Decision Tables

- Consists of three parts
  - Condition stubs:
    - that part of a decision table that lists the conditions relevant to the decision.
  - Action stubs:
    - that part of a decision table that lists the actions that result for a given set of conditions.
  - Rules:
    - that part of a decision table that specifies which actions are to be followed for a given set of condition.

simplify decision table for payroll system example

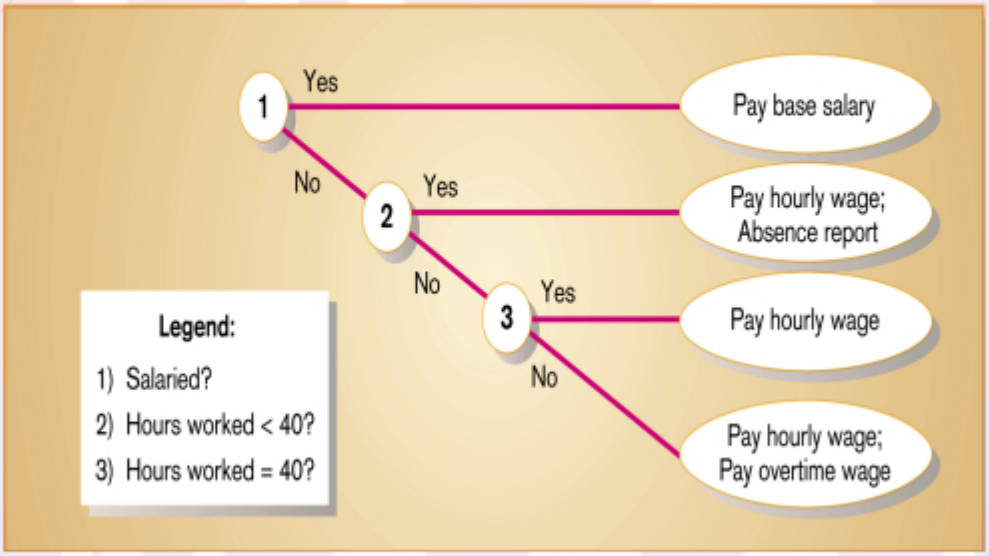
Figure 8-5 Reduced decision table for payroll system example

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		

Modeling Logic with Decision Tables

- A graphical representation of a decision situation
- Decision situation points (nodes) are connected together by arcs and terminate in ovals
- Two main components
  - Decision points represented by nodes
  - Actions represented by ovals
- Read from left to right
- Each node corresponds to a numbered choice on a legend
- All possible actions are listed on the far right

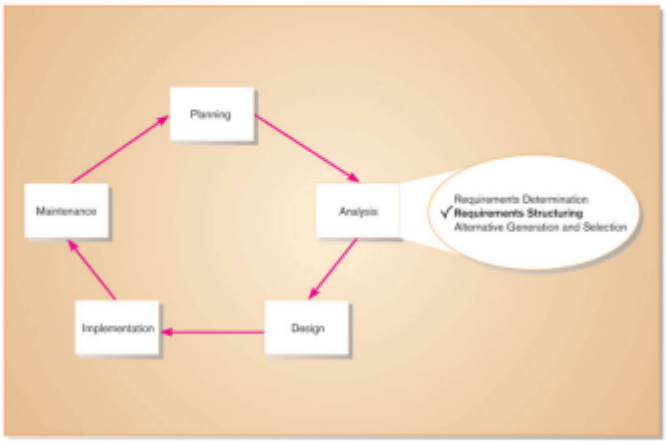
Figure 9-9
Decision tree representation of the decision logic in the decision tables in Figures 9-4 and 9-5, with only two choices per decision point



3.3. Structuring System Data Requirements

Data Modeling

Figure 9-1 Systems development life cycle with analysis phase highlighted



Conceptual Data Model

A detailed model that captures the overall structure of organizational data while being independent of any database management system or other implementation consideration

It is typically done in parallel with other requirements analysis and structuring steps during system analysis

Conceptual Data Modeling Process

- Develop a data model for the current system.
- Develop a new conceptual data model that includes all requirements of the new system.
- In the design stage, the conceptual data model is translated into a physical design.
- Project repository links all design and data modeling steps performed during SDLC.

Process of Conceptual Data Modeling

- First step is to develop a data model for the system being replaced
- Next, a new conceptual data model is built that includes all the requirements of the new system- evolve through various iteration.
- In the design stage, the conceptual data model is translated into a physical design, data storage architecture is been selected, then file and DB are defined.

Deliverables and Outcome

- Primary deliverable is an entity-relationship (E-R) diagram or class diagram.
- As many as 4 E-R or class diagrams are produced and analyzed:
  - E-R diagram that covers data needed in the project's application.
  - E-R diagram for the application being replaced
  - E-R diagram for the whole database from which the new application's data are extracted.
  - E-R diagram for the whole database from which data for the application system being replaced is drawn.

Deliverables and Outcome

- Second deliverable is a set of entries about data objects to be stored in repository or project dictionary.
  - Repository links data, process, and logic models of an information system.
  - Data elements included in the DFD must appear in the data model and vice versa.
  - Each data store in a process model must relate to business objects represented in the data model.

Gathering Information for Conceptual Data Modeling

- *Top-down approach* for a data model is derived from an intimate understanding of the business.
- *Bottom-up approach* for a data model is derived by reviewing specifications and business documents

Gathering Information for Conceptual Data Modeling

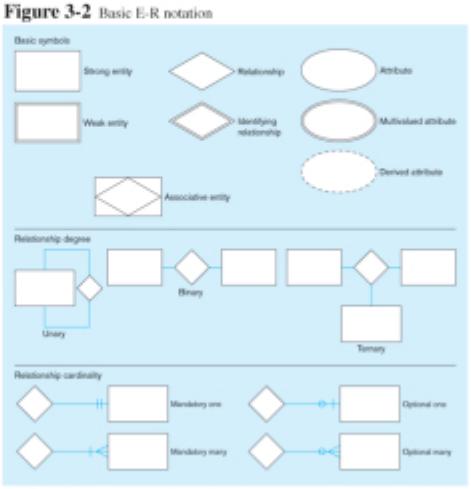
- Requirements Determination Questions for Data Modeling:
  - What are subjects/objects of the business?
    - Data entities and descriptions.
  - What unique characteristics distinguish between subjects/objects of the same type?
    - Primary key
  - What characteristics describe each subject/object?
    - Attributes and secondary keys.
  - How do you use the data?
    - Security controls and user access privileges.

Gathering Information for Conceptual Data Modeling

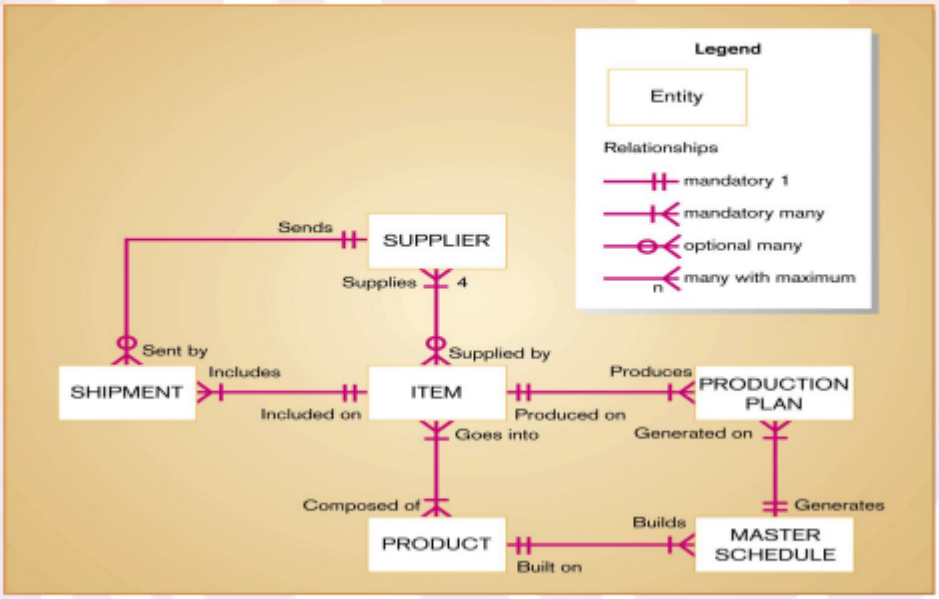
- Over what period of time are you interested in the data?
  - Cardinality and time dimensions.
- Are all instances of each object the same?
  - Supertypes, subtypes, and aggregations.
- What events occur that imply associations between objects?
  - Relationships and cardinalities.
- Are there special circumstances that affect the way events are handled?
  - Integrity rules, cardinalities, time dimensions.

Basic ER Notations

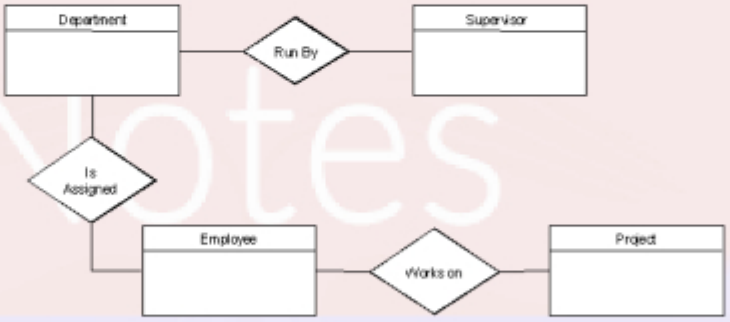
Figure 3-2 Basic E-R notation



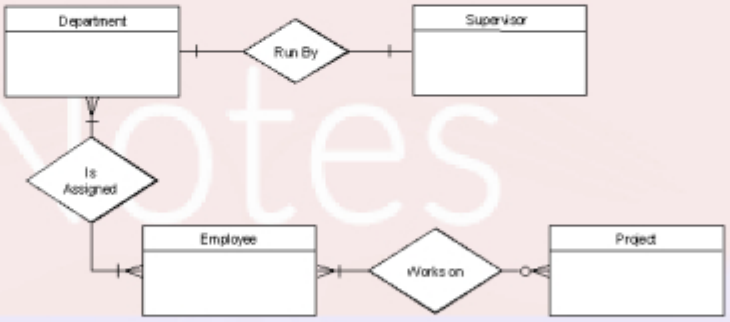
Sample conceptual data model diagram



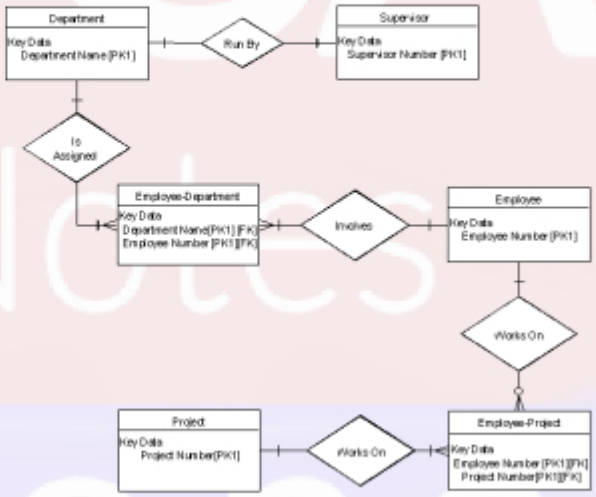
Rough ERD



Fill cardinality in ERD



Fill attribute in ERD



Entity-Relationship (E-R) Modeling

- **Entity-Relationship data model** (**E-R model**): a detailed, logical representation of the entities, associations and data elements for an organization or business area.
- **Entity-relationship diagram** (**E-R diagram**): a graphical representation of an E-R model.
- The E-R model is expressed in terms of:
  - Data entities in the business environment.
  - Relationships or associations among those entities.
  - Attributes or properties of both the entities and their relationships.

Entity-Relationship (E-R) Modeling

Key Terms

- Entity
  - A person, place, object, event or concept in the user environment about which the organization wishes to maintain data (person, place, object, event)
  - Represented by a rectangle in E-R diagrams
- Entity Type
  - A collection of entities that share common properties or characteristics- called entity class (employee, course, account)
- Attribute a named property or characteristic of an entity that is of interest to the organization.
  - Naming an attribute: i.e. Vehicle_ID.
  - Place its name inside the rectangle for the associated entity in the E-R diagram.

Entity-Relationship (E-R) Modeling

Key Terms

- Candidate key:
  - an attribute (or combination of attributes) that uniquely identifies each instance of an entity type. (Employee_id, Employee name and employee address), because we have two candidate keys we select Employee_id as identifier
- Identifier:
  - a candidate key that has been selected as the unique, identifying characteristic for an entity type.

Entity-Relationship (E-R) Modeling

Key Terms

- **Identifier**
  - A candidate key that has been selected as the unique identifying characteristic for an entity type
  - Selection rules for an identifier
    - Choose a candidate key that will not change its value.
    - Choose a candidate key that will never be null.
    - Avoid using intelligent keys.
    - Consider substituting single value surrogate keys for large composite keys.

Entity instance, a single occurrence of an entity type.(course cis339, cis381)

Entity-Relationship (E-R) Modeling

Key Terms

- Multi-valued Attribute
  - An attribute that may take on more than one value for each entity instance (multi skills for one employee)
  - Represented on E-R Diagram in two ways:
    - double-lined ellipse
    - weak entity (employee and dependent name, age, and relation).
- Repeating group: a set or two or more multivalued attributes that are logically related.

Entity-Relationship (E-R) Modeling

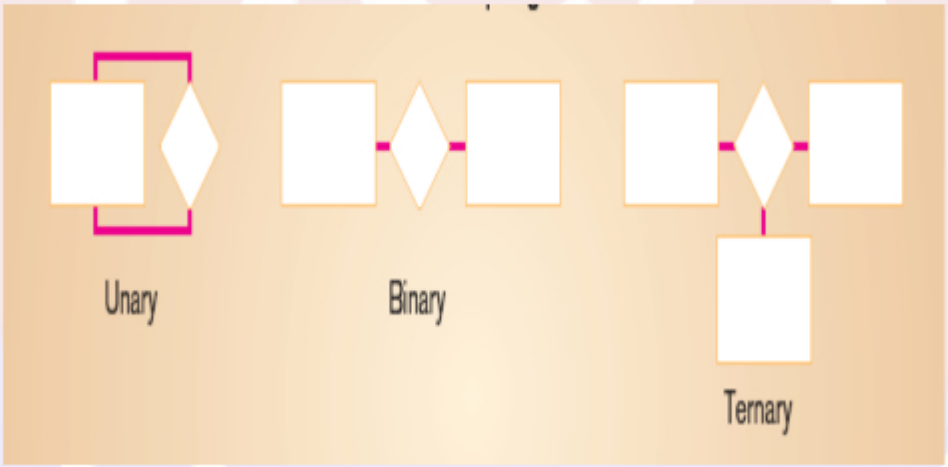
Key Terms

- Relationship
  - An association between the instances of one or more entity types that is of interest to the organization
  - Association indicates that an event has occurred or that there is a natural link between entity types
  - Relationships are always labeled with verb phrases (student *completes* course)

Degree of Relationship

- Degree
- Number of entity types that participate in a relationship, so the relationship *Student Completes Course* is of degree 2 since it has two entities *Student* and *Course*)
- Three cases
  - Unary
    - A relationship between two instances of one entity type
  - Binary
    - A relationship between the instances of two entity types
  - Ternary
    - A simultaneous relationship among the instances of three entity types (in the next example quantity can not be associated with any of the three possible binary relationship)- *associated entity.*
    - Not the same as three binary relationships

Example relationships of different degrees

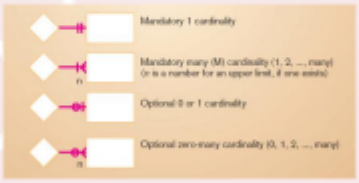


Cardinality

Cardinality:

- the number of instances of entity B that can (or must) be associated with each instance of entity A.
- Minimum Cardinality
  - The minimum number of instances of entity B that may be associated with each instance of entity A
- Maximum Cardinality
  - The maximum number of instances of entity B that may be associated with each instance of entity A.
- Mandatory vs. Optional Cardinalities
  - Specifies whether an instance must exist or can be absent in the relationship.

Cardinality Symbols



Naming and Defining Relationships

- A relationship name is a *verb phrase* and *avoid vague names*.
- A relationship definition:
  - Explains *what action is to be taken and possibly why it is important*.
  - *Gives examples to clarify the action*.
  - Explain any *optional participation*.
  - *Explain the reason for any explicit maximum cardinality* other than many.
  - *Explain any restrictions on participation in the relationship*.
  - Explain the extent of history that is kept in the relationship.
  - *Explain the extent of history that is kept in the relationship*.
  - *Explain whether an entity instance* involved in a relationship instance *can transfer participation to another relationship instance*