Week 2- investigating system requirements

Understanding business case:

1. **Business case**: Justification for a proposal, aligned to organizational missions, objectives and IT needs.
2. Any relation between “Systems projects and business/organization”?
3. Where the organization is at present, where they want to be?
4. A business case should:

– Be comprehensive and easy to understand

– Describe the project clearly, provide the justification to proceed, and estimate the project’s financial impact

1. Questions answered by a business case

* Why are we doing this project?
* How much will it cost and how long will it take?
* Are there any risks involved?
* How will we measure success?
* What alternatives exist?

Systems development request:

Systems request:

* Stronger controls
* More support
* Improved Service
* Better performance
* More information
* Reduced cost

Why investigating requirements?

* Communication efficiency is low

What are requirements?

System Requirements

* **Functional** requirements: the activities the **system must perform**
  + Business uses, functions the users carry out
  + Shown as use cases
* **Non-functional** requirements: other system characteristics
  + Constraints and performance goals

|  |  |  |
| --- | --- | --- |
| Requirement categories | FURPS categories | Example requirements |
| Functional | **F**unctions | Business rules and processes |
| Nonfunctional | **U**sability  **R**eliability  **P**erformance  **S**ecurity | User interface, ease of use  Failure rate, recovery method  Response time, throughput  Assess controls, encryption |

Additional requirement categories

Design constraints

* Specific restrictions for hardware and software

Implementation requirements

* Specific languages, tools, protocols, etc.

Interface requirements

* Interface links to other systems

Physical requirements

* Physical facilities and equipment constraints

Supportability requirements

* Automatic updates and enhancement methods

**Evaluation** of systems requirements

Systems requests are evaluated by a **systems review committee** or a **computer resources committee**

**Systems Request Forms**

◦ Streamline the request process

◦ Ensure consistency

◦ Easy to understand

◦ Include clear instructions

◦ Indicate the required supporting documents

◦ Submitted electronically

Systems Review Committee

* A **broader viewpoint** enables a committee to establish priorities more effectively than an individual

• One person’s **bias** is **less** likely to **affect decisions**

* Disadvantages

• Action on requests must **wait** until the committee meets

• Members might **favor** projects requested by their **own** departments

• **Internal** political differences could **delay** important decisions

Overview of Feasibility

* Feasibility studies can be simple or exhaustive
* Effort required depends on the nature of the request
* Initial fact-finding involves:
  + Studying organizational charts
  + Performing interviews
  + Reviewing current documentation
  + Observing operations
  + Surveying users
* A **feasibility** study examines operational, technical, economic, and schedule factors.
  + **Operational**: will it be easy to learn and use?
  + **Technical**: Do we have the tech resources?
  + **Economic**: will benefits exceed costs?
  + **Schedule**: Can we do it in time?

▪ **Operational Feasibility**

– A proposed system will be used effectively after it has been developed

– Can be affected by organizational culture

– Cannot be accurately measured but requires careful study

– Questions that can help predict a system’s operational feasibility

• Is the project supported by management and users?

• Will the new system result in a workforce reduction?

• Do legal or ethical issues need to be considered?

▪ **Economic Feasibility**

– Projected benefits of a proposed system out-weigh **total cost of ownership (TCO)** – Determination of TCO requires cost analysis of:

• People, including IT staff and users

• Hardware and equipment

• Software

• Formal and informal training

• Licenses and fees

• Consulting expenses

• Facility costs

– **Tangible costs** are measured in dollars

– **Intangible costs** can significantly affect organizational performance

– **Tangible benefits** can result from a decrease in expenses or an increase in revenues

– **Intangible benefits** are important to the company despite the inability to measure them in dollars

▪ **Technical Feasibility**

– Technical resources required to acquire and use the system

– Questions analysts should ask

• Does the company have the necessary hardware, software, and network resources?

• Does the company have the required technical expertise?

• Does the proposed platform have sufficient capacity for future needs?

• Will a prototype be required?

▪ **Schedule Feasibility**

– A project can be implemented in an acceptable time frame

– Issues that can affect schedule feasibility

• Interaction between time and costs

• Can the company or the IT team control the factors that affect schedule feasibility?

• Has management established a firm timetable for the project?

• What conditions must be satisfied during the development of the system?

• Will an accelerated schedule pose any risks?

Evaluating Feasibility

▪ Identify and weed out systems requests that are not feasible

▪ Requests that are not currently feasible can be resubmitted as new hardware, software, or expertise becomes available

Prepare for investigation

▪ **Interaction with Stakeholders**

▪ Stakeholders– persons who have an interest in the successful implementation of the system

- **Internal** Stakeholders– persons within the organization

- **Externa**l stakeholders – persons outside the organization

- **Operational** stakeholders – persons who regularly interact with the system

- **Executive** stakeholders– persons who don’t directly interact, but use the information or have financial interest

▪ Steps in the preliminary investigation

**–** **Understand the problem or opportunity**

• Develop a **business profile** that describes **current business processes and functions**

• Understand how modifications will affect business operations and other information systems

• Identify the **departments, users, and business processes** involved

• Consider using a **fishbone diagram**

**– Define the project scope and constraints**

• Define the specific **boundaries**, or extent, of the project

• Define **project scope** by creating a list with sections called **must do, should do, could do, and won’t do**

• Avoid project creep

– **Project creep**: Process by which projects with very general scope definitions expand gradually, without specific authorization

• Identify constraints

– **Constraint**: A requirement or condition that the system must satisfy or an outcome that the system must achieve

– **Perform fact-finding (重点！)**

there are many techniques:

1. Gather and analyze data

▪ about project usability, costs, benefits, and schedules

▪ Pareto chart (bar diagram), XY chart (scatter diagram)

b) Analyze organization charts

c) Conduct interviews

- Select interviewees

- Design interview questions

- Prepare for the interview

- Conduct the interview

- Post interview follow-up

d) Review documentation

e) Observe operations

f) Conduct a user survey

g) **Joint Application Development**

Key stakeholders – facilitator, scribe, participants.

**– Analyse project usability, cost, benefit, and schedule data**

• Factors to consider

– What information must be obtained, and how will it be gathered and analyzed?

– Who will conduct the interviews? How many people will be interviewed?

– Will a survey be conducted? Who will be involved? How much time will it take to tabulate the results?

– How much will it cost to analyze the information and prepare a report with findings and recommendations?

**– Evaluate feasibility**

• Operational feasibility

• Technical feasibility

• Economic feasibility

• Schedule feasibility

**– Present results and recommendations to management**

• Prepare a report that includes:

– An evaluation of the systems request

– An estimate of costs and benefits

– A case for action

• Format of a report

– Introduction

– Systems request summary

– Findings

– Recommendations

– Project roles

– Time and costs estimates

– Expected benefits

– Appendix

System Requirements Prioritization

▪ Requirements must be **SMART**

- **S**pecific

- **M**easurable

- **A**ttainable

- **R**ealistic

- **T**ime-bound

▪ Techniques— Must, Should, Could, Won’t or Would

▪ Factors that Affect Priority

– Will the proposed system **reduce costs**?

– Will the system **increase revenue** for the company?

– Will the systems project result in more information or produce **better results**?

– Will the system **serve customers better**?

– Will the system **serve the organization better**?

– Can the project be implemented in a **reasonable time period**?

– Are the necessary financial, human, and technical **resources available**?

▪ Discretionary and Nondiscretionary Projects

– **Discretionary projects**: Projects where **management has a choice** in implementing them

– **Nondiscretionary projects**: **Management has no choice** in implementing a project

• Most of these projects are predictable

– Annual updates to payroll

– Tax percentages

– Quarterly changes

week 3 managing systems projects

Agile project management

**Agile** means being able to move quickly and easily

**Early software development projects** often used a waterfall approach, requirements were unknown or continuously changing

**Agile today** means using a method based on iterative and incremental development

▪ Agile Scope Management

– Scope is not well understood, but needs to be controlled

▪ Agile Time Management

– Schedule must be flexible due to changes

▪ Agile Cost Management

– Costs are more difficult to estimate

▪ Agile Risk Management

– Higher risk aspects of project are completed first

▪ Agile Quality Management

– Quality assessed after each iteration

Managing Scope in Systems Projects

Scope Management Processes

▪ Planning scope

▪ **Collecting requirements**

▪ Defining scope

▪ Creating the WBS

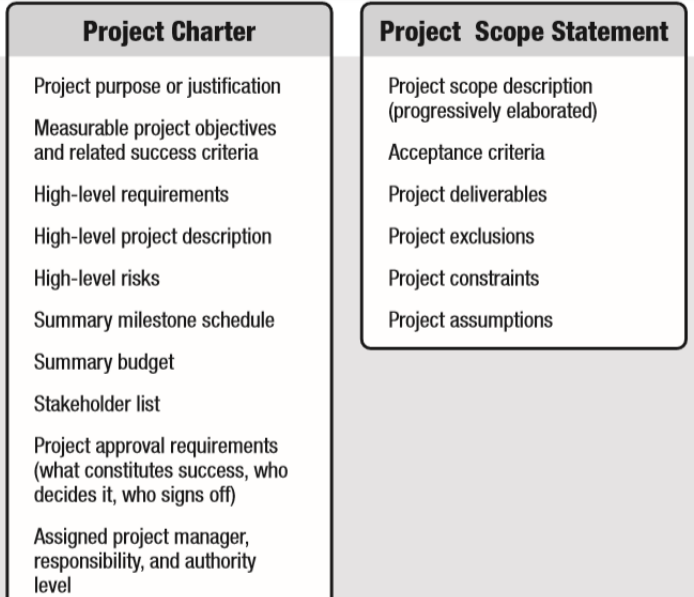
▪ Validating scope

▪ Controlling scope

Defining Project Scope

Input: Project Charter

Output: Project Scope Statement



Approaches to Developing WBSs

▪ The analogy approach

▪ The top-down approach

▪ The bottom-up approach

▪ Mind-mapping approach

Managing Time in Systems Projects

Project Time Management Processes

▪ Plan schedule management

▪ Define activities

▪ Sequence activities

▪ Estimate activity resources

▪ Estimate activity durations

▪ Develop the schedule

▪ Control the schedule

Defining Activities

▪ An **activity or task** is an element of work normally found on the work breakdown structure (WBS) that has an expected duration, a cost, and resource requirements

Milestones

▪ A **milestone** is a significant event that normally has no duration

SMART Criteria

▪ **Milestones** should be

– Specific

– Measurable

– Assignable

– Realistic

– Time-framed

Sequencing Activities

▪ Involves reviewing activities and determining dependencies

▪ A **dependency or relationship** is the sequencing of project activities or tasks

▪ You *must* determine dependencies in order to use critical path analysis

Three types of Dependencies

▪ Mandatory dependencies（刚需）

▪ Discretionary dependencies（非刚需）

▪ External dependencies（外部需求）

Network Diagrams

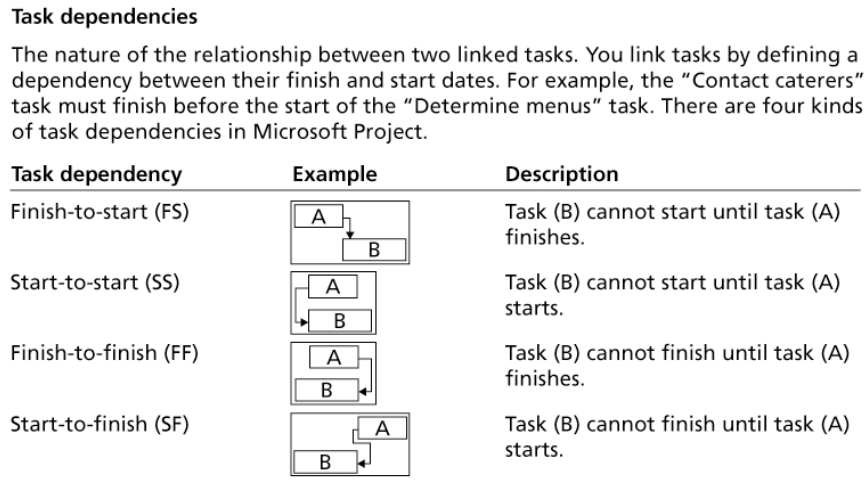
▪ **Network diagrams** are the preferred technique for showing **activity sequencing**

▪ A network diagram is a schematic display of the **logical relationships** among, or sequencing of, project activities

▪ Activity On the Node (AON)

▪ Activity-On-Arrow (AOA)（两种图论。。。）

Task Dependency Types



Estimating Activity Resources

▪ must have a good idea of the quantity and type of resources

▪ **Resources** are people, equipment, and materials

Activity Duration Estimating

▪ **Duration** includes the actual amount of **time worked** on an activity plus **elapsed time**

▪ **Effort** is the number of **workdays** or work hours required to complete a task

Factors Affecting Activity Duration

**▪ Project size**

**▪ Human resources**

**▪ Experience with similar projects**

**▪ Constraints（要求）**

Three-Point Estimates

▪ Instead of providing activity estimates as a discrete number, such as four weeks, it’s often helpful to create a **three-point estimate** – **optimistic, most likely, and pessimistic estimate**

▪ Three-point estimates are needed for **PERT**

Developing the Schedule

▪ Important tools and techniques include

**- Gantt Chart,**

**- Critical path Method (CPM),**

- Critical chain scheduling,

**- PERT analysis**

Issues With Schedule

▪ Problems with timetables and project milestones can indicate:

– Failure to recognize task dependencies

– Confusion between effort and progress

– Poor monitoring and control methods

– Personality conflicts among team members

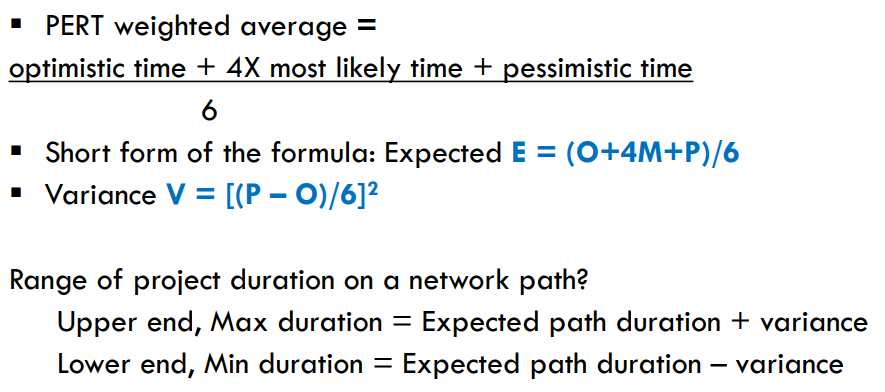
– Turnover of project personnel – Failure of an IT project

**Critical Path** Method (CPM)

▪ CPM is a network diagramming technique used to predict total project duration

▪ The critical path is **the longest path through the network diagram and has the least amount of slack or float （时间最长）**

**PERT** -- Example （重点！计算公式！）



Managing Cost in Systems Project

Cost/Benefit Analysis

▪ Cost modeling/Budget

▪ Benefit-cost ratio

▪ NPV

▪ Payback period

Managing Quality in Systems Project

DMAIC

▪ DMAIC is a systematic, closed-loop process for continued improvement that is scientific and fact based

▪ DMAIC stands for:

– **Define**: Define the problem/opportunity, process, and customer requirements

– **Measure:** Define measures, then collect, compile, and display data

**– Analyze:** Scrutinize process details to find improvement opportunities

– **Improve:** Generate solutions and ideas for improving the problem

– **Control:** Track and verify the stability of the improvements and the predictability of the solution

Managing Risk in Systems Project

Project Risk Management Processes

**▪ Planning risk management**

**▪ Identifying risks:**

**▪ Performing qualitative risk analysis:**

**▪ Performing quantitative risk analysis:**

**▪ Planning risk responses:**

**▪ Controlling risk**

Week5

Learning Objectives

▪ Understand the process of **computing requirements prioritization**

▪ Describe **requirements modeling** as the first part of systems analysis activities

▪ Explain Team-based techniques (**JAD, RAD, Agile**) for modeling requirements

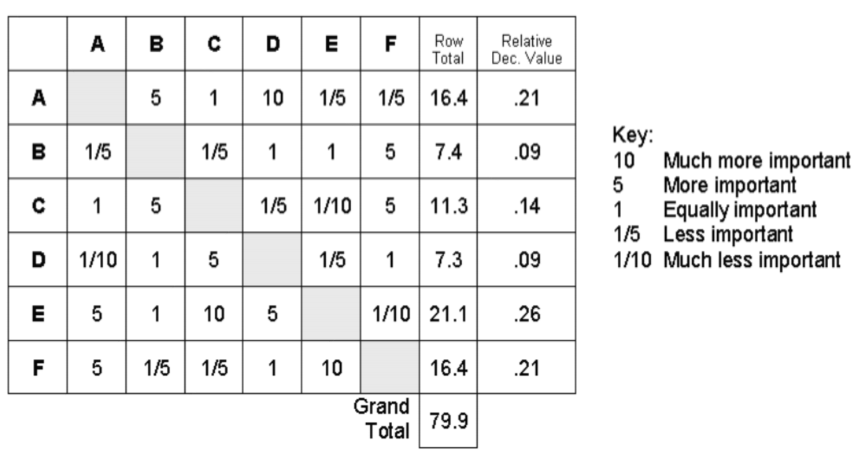
▪ Use a **functional decomposition diagram (FDD)** to model business functions and processes

▪ Describe the **Unified Modeling Language (UML)** and examples of UML diagrams

▪ List and describe **system requirements, including outputs, inputs, processes, performance, and controls**

Computing Requirements Prioritization

-- L-shaped Matrix



“b比a更重要 所以是5 相对的 a对于b就是五分之一

c和a一样重要 所以是1

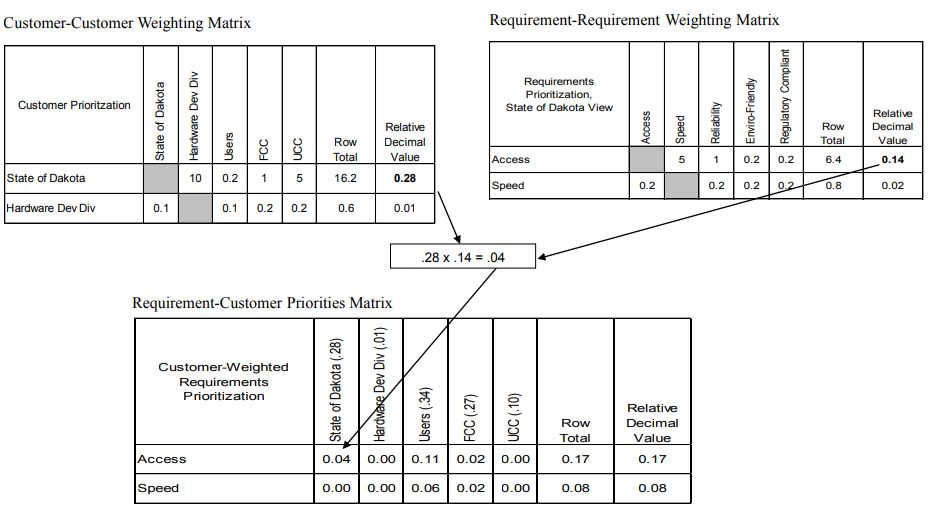
d比a重要的多 所以是10 相对的 a对于d就是十分之一 etc”

-- Customer Prioritization Matrix

-- Requirements List

▪ Requirements list needs to be unique based on identified stakeholders.

先整理出customer对customer， 再整理不同客户irement对requirement， 最后通项相乘



Requirement modelling

**Importance**:

▪ Specifications are extremely long.

▪ Specifications are complex.

▪ Visualization matters

Skills required:

▪ Strong analytical and interpersonal skills

Systems Analysis Activities

▪ Notice that the systems analysis tasks are interactive

▪ Requirements modeling

– Involves identification of the requirements for new system

– Includes – why, what, who issues

**Objectives** of Requirements Modeling:

- **Discover the boundaries** of the new system (or software) and how it must interact with its environment within the new problem domain

- **Detect and resolve conflicts** between (user) requirements

- **Negotiate priorities of stakeholders**

- **Prioritize requirements**

- **Elaborate system requirements**, defined in the requirement specification document, such that managers can give realistic project estimates and developers can design, implement, and test

- **Classify requirements** information into various categories and allocate requirements to sub-systems

- **Evaluate requirements** for desirable qualities

**Use** of requirements modelling:

▪ This is an essential task in specifying requirements

▪ Map elements obtained by elicitation to a more precise form

▪ Help better understand the problem

▪ Help find what is missing or needs further discussion

“**Why** Issues”?

▪ The goals for a new system need to be identified, analyzed, and refined.

▪ Such goals are usually obtained by analyzing problems with the current situation, identifying new opportunities, exploring scenarios of interaction, and so on.

▪ Beside functional goals (e.g., satisfaction of requests, information of the state of affairs) there are many nonfunctional ones (e.g., safety, security, performance, evolvability, etc.).

“**What** Issues”?

▪ The requirements operationalizing the various goals identified need to be defined precisely and related to each other; in parallel, the assumptions made in the operationalization process need to be made explicit and documented.

▪ Beside functional requirements about services to be provided there is a wide spectrum of non-functional requirements about quality of service.

“**Who** Issues”?

▪ The requirements need to be assigned as contractual obligations among the various agents forming the composite system-to-be.

▪ These include the software to be developed, human agents, sensors/component of a machine, existing software, etc.

Team-Based Approaches: **JAD, RAD, and Agile** Methods

**– Goal - To deliver the best possible system at the lowest possible cost in the shortest possible time**

• Joint application development **(JAD)** **brings users into the design process** (briefly discussed in lecture 2)

• Rapid application development **(RAD)** **is a condensed version** of the system development life cycle (briefly discussed in lecture 1)

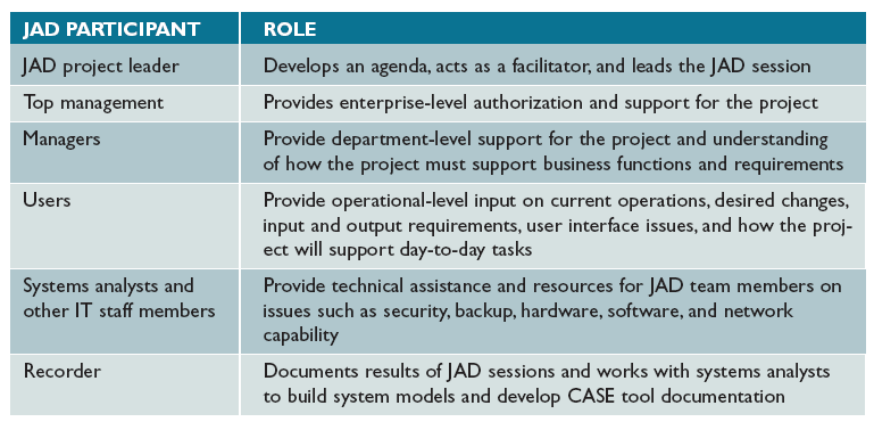
• **Agile** methods stress **intense interaction between developers and users** (discussed in lecture 3)

**JAD** Approach for Requirements Modeling

– **User Involvement** (formal or informal)

– Helps create a successful system

– **JAD Participants and Roles**



**▪ JAD Advantages and Disadvantages**

– **Disadvantages**

• **More expensive** than traditional methods

• Can be c**umbersome** if the group is too large

– Advantages

• Allows **key users to participate** effectively

• Users are more likely to feel a **sense of ownership**

• Produces a more **accurate statement** of system requirements, a better **understanding of common goals**, and a **stronger commitment** to the success of the new system

**JAD feature:**

▪ Uses a group approach

▪ End product - New information system

▪ Complete methodology

– Includes a four-phase life cycle that parallels the traditional SDLC

– Reduces cost and development time

– Increases the probability of success

– Relies on prototyping and user involvement

• Prototypes are modified based on user input

**RAD** Approach for Requirements Modeling

▪ **RAD** **Objectives**

– Cut development time and expense

• Involve users in every phase of systems development

• Must have the right IT resources, skills, and management support

▪ **RAD** **Advantages and Disadvantages**

– Advantage

– Helps develop systems **quickly with significant cost savings**

– Disadvantages

• Does **not emphasize** the company’s strategic **business needs**

• **Less time to develop quality**, consistency, and design standards

**Agile** Approach

▪ **Advantages**

– Very **flexible and efficient** in dealing with **change**

– Frequent deliverables constantly validate the project and **reduce risk**

▪ **Disadvantages**

– Team members need a **high level of technical and interpersonal skills**

– **Lack of structure and documentation** can introduce risk factors

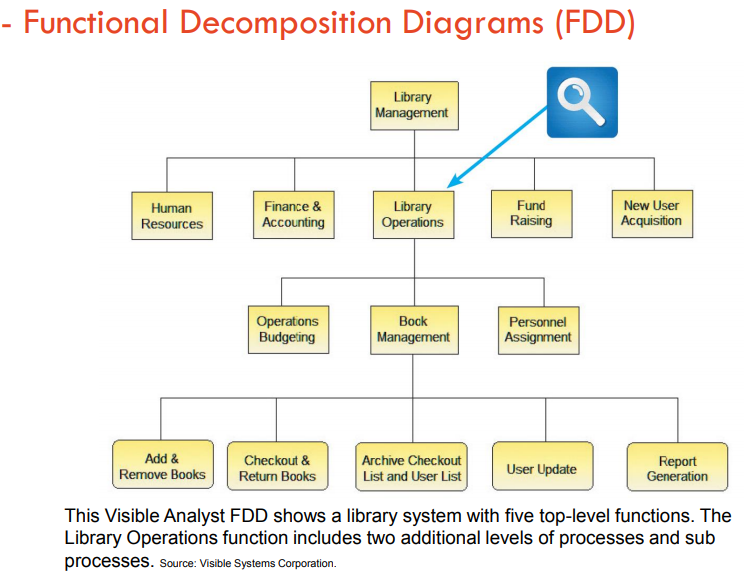
– May be subject to **significant change in scope**

Modeling Tools and Techniques

▪ **Functional Decomposition Diagrams (FDD)**

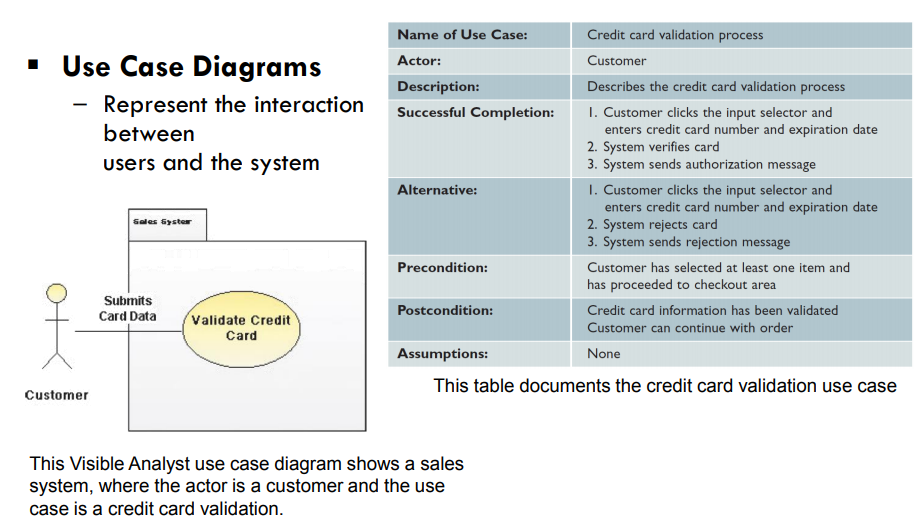
– Top-down representation of a function or process

– Help analysts show business functions and how they are organized into lower-level processes



▪ **Use Case Diagrams**

– Represent the interaction between users and the system



**UML Use Case Diagram for Modeling Functional Requirements**

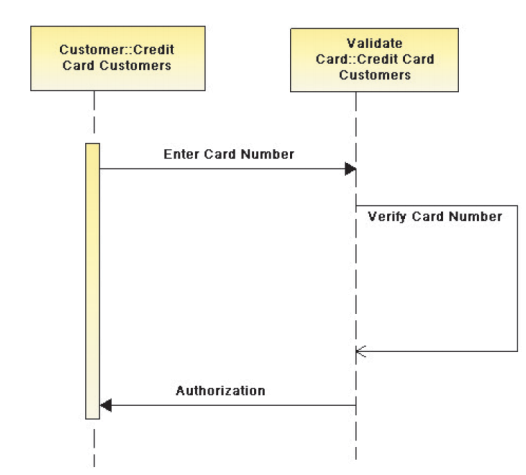
▪ Most often and traditional way to define functional requirements for the new information system is to model them using UML language and its use case diagrams.

▪ The requirements must be discussed well with the users.

▪ One of the advantage is that the symbols used are quite easy to understand even with little experiences in systems development or how systems work.

▪ **Sequence Diagrams**

– Show the timing of interactions between objects as they occur



This Visible Analyst sequence diagram shows a credit card validation process.

**System Requirements Checklist（重点！）**

▪ **Input** Examples

– The department head must enter overtime hours on a separate screen

– Student grades must be entered on machine-readable forms prepared by the instructor

– Each input form must include date, time, product code, customer number, and quantity

▪ **Process** Examples

– The student records system must calculate the GPA at the end of each semester

– The human resources system must interface properly with the existing payroll system

– The prescription system must automatically generate an insurance claim form

▪ **Output** Examples

– The Web site must report online volume statistics every four hours, and hourly during peak periods

– The contact management system must generate a daily reminder list for all sales reps

– The purchasing system must provide suppliers with up-to-date specifications

▪ **Performance** Examples

– The system must support 25 users online simultaneously

– Response time must not exceed four seconds

– The system must be operational seven days a week, 365 days a year

▪ **Control** Examples

– The system must provide logon security at the operating system level and at the application level

– The system must maintain separate levels of security for users and the system administrator

– All transactions must have audit trails

– The system must create an error log file that includes the error type, description, and time

Week6 Data and Process Modeling（重点知识）

Learning Objectives

▪ **Describe** **data and process modeling concepts and tools**, including **data flow diagrams**, a **data dictionary**, and **process descriptions**

▪ **Describe** the **symbols** used in data flow diagrams and **explain** the **rules** for their use

▪ **Draw data flow diagrams** in a sequence, from general to specific

▪ Explain how to **level and balance** a set of data flow diagrams

▪ Describe how a **data dictionary** is used and what it **contains**

▪ Use process description tools, including **structured English**, **decision tables**, and **decision trees**

▪ Describe the relationship between **logical and physical** models

Overview of Data and Process Modeling Tools

▪ **Data flow diagram** (DFD) - Uses various symbols to show how the system transforms input data into useful information

**Data Flow Diagrams**

▪ A data flow diagram (DFD) **shows how data moves through an information system** but does **not show program logic or processing steps**

▪ A set of DFDs provides a logical model that **shows what the system does**, **not how it does**

▪ DFD **Symbols**

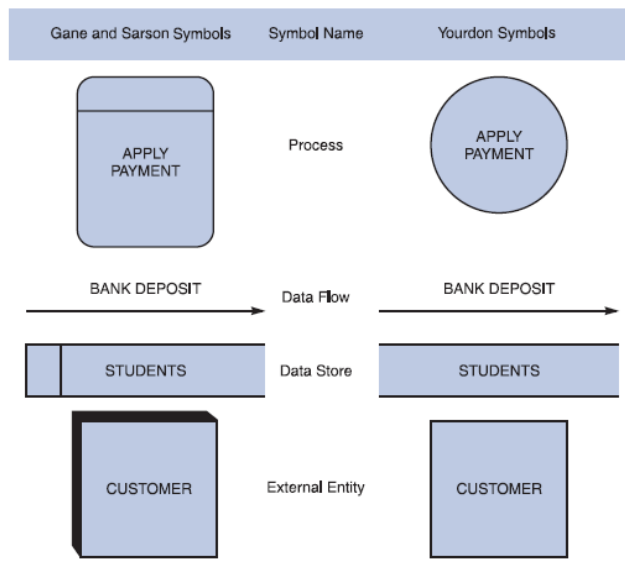
– Four basic symbols represent processes, data flows, data stores, and entities

– **Gane** and **Sarson**: Used in data flow diagrams

• Processes, data flows, data stores, and external entities all have a unique symbol

– **Yourdon**: Used in data flow diagrams

• Processes, data flows, data stores, and external entities each have a unique symbol

例子：

**Process** Symbol （圈 代表信息处理与转化）

• Must have at least one input and at least one output

• Contains **business logic** that transforms the data

• Process name identifies its function (verb)

• Examples: “apply rent payment” or “calculate commission”

• In DFDs, a process symbol can be referred to as a **black box**

**Data Flow** Symbol（箭头 代表某个数据的流动方向）

◦ Represents one or more data items

◦ The symbol for a data flow is a line with a single or double arrowhead

– Following data flow and process combinations must be avoided

◦ Spontaneous generation （两边往外 没有输入）

◦ Black holes （两边往里 没有输出）

◦ Gray holes （输入和输出没关系）

Data Store symbol （长条 代表数据库）

• Represent data that the system stores

• A DFD does not show the detailed contents of a data store — the specific structure and data elements are defined in the data dictionary

• A data store must be connected to a process with a data flow

• 必须至少一进一出， 必须链接“圆”

Entity Symbol（方块 代表与系统互动的外界元素）

• Shows how the system interfaces with the outside world

• A DFD shows only external entities that provide data to the system or receive output from the system

• DFD entities also are called terminators because they are data origins or final destinations

• Each entity must be connected to a process by a data flow（必须通过箭头链接圆 不能链接长条或自己）

**Creating a Set of DFDs（如何画dfd）**

▪ Guidelines for Drawing DFDs

– Draw the context diagram so that it fits on one page

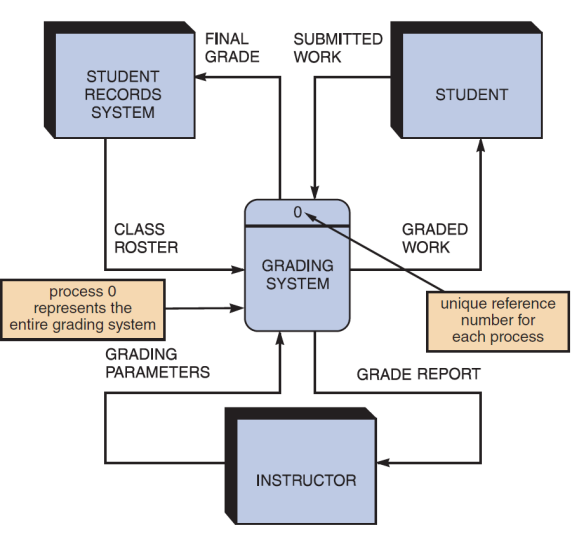
– Use the name of the information system as the process name in the context diagram

– Use unique names within each set of symbols

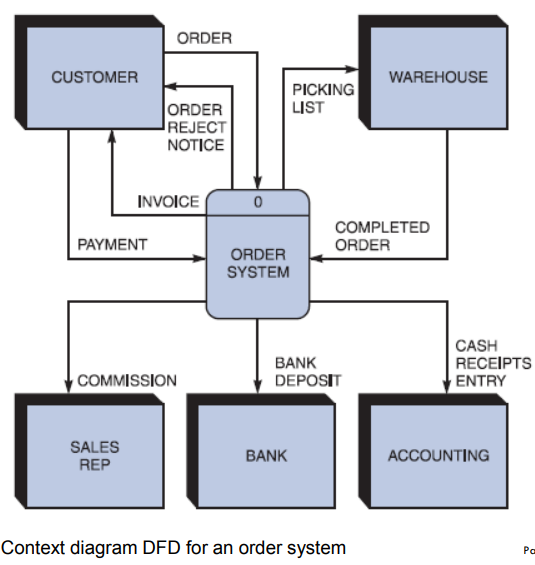
– Do not cross lines

– Provide a unique name and reference number for each process

– Ensure that the model is accurate, easy to understand, and meets the needs of its users

例子：

**▪ Step 1: Draw a Context Diagram**



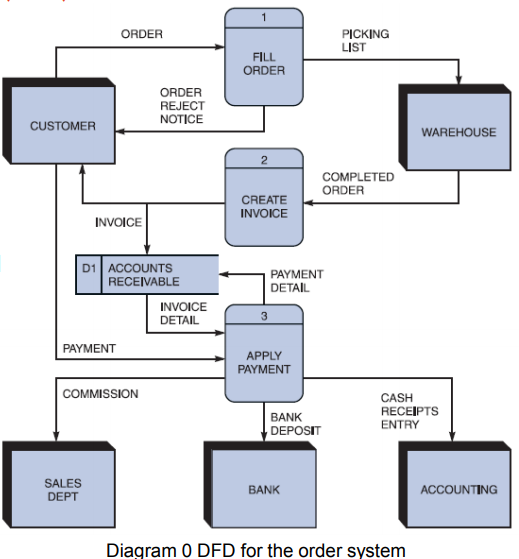
▪ **Step 2: Draw a Diagram 0 DFD**

– If same data flows in both directions, you can use a double-headed arrow

– Diagram 0 is an exploded view of process 0

– Parent diagram

– Child diagram

– Functional primitive

▪ **Step 3: Draw the Lower Level Diagrams（1.1 1.2 1.3构成的小图 解构一个大圆）**

▪ Must use **leveling**（进步划分 把一个圆解构成一张图） and **balancing（让小图的输入和输出和大图中的圆对应的一致）** techniques

▪ **Leveling** examples

– Uses a series of increasingly detailed DFDs to describe an information system

– Exploding, partitioning, or decomposing

**Data Dictionary**

▪ A data dictionary, or data repository, is a central **storehouse of** information about a **system’s data**

▪ An analyst uses the data dictionary to **collect, document, and organize specific facts** about a system

▪ Defines and **describes all** **data elements** and meaningful **combinations of data elements**

**What it contains:**

▪ **Data element**: Smallest piece of data that has meaning within an information system

– Called **data item** or **field**

– Are combined into records, also called data structures

▪ **Record**: Meaningful combination of related data elements that is included in a data flow or retained in a data store

– Called **data structures**

▪ Using CASE Tools for Documentation

– More complex the system, more difficult it is to maintain full and accurate documentation

– Modern CASE tools simplify the task

– A CASE repository ensures data consistency

▪ Documenting the **Data Elements** （记住每个东西都含什么不含什么 考试会出选择）

– Every data element in the data dictionary should be documented

– **Objective**: To provide clear, comprehensive information about the data and processes that make up a system

▪ Contains:

– Data element name and label

– **Alias**

– Type and length

– Default value

– Acceptable values - **Domain** and **validity rules**

– Source

– Security

– Responsible user(s)

– Description and comments

▪ Documenting the **Data Stores**

◦ Data store name or label

◦ Description

◦ Alternate name(s)

◦ Attributes

◦ Volume and frequency

▪ Documenting the **Entities** - Data dictionary describes all external entities that interact with the system

– Characteristics include

• Entity name

• Description

• Alternate name(s)

• Input data flows

• Output data flows

▪ Documenting the **Records**

– Record or data structure name

– Definition or description

– Alternate name(s)

– Attributes

▪ **Data Dictionary Reports** :Following can be obtained

– Alphabetized list of all data elements by name

– Report describing each data element and indicating the user or department that is responsible for data entry, updating, or deletion

– Report of all data flows and data stores that use a particular data element

– Detailed reports showing all characteristics of data elements, records, data flows, processes, or any other selected item stored in the data

**Process Description Tools**

▪ **Process description**: Documents the details of a functional primitive (at the lowest level of the DFDs and can be decomposed no further) and represents a specific set of processing steps and business logic

▪ Tools - structured English, decision tables, and decision trees

▪ Used in object-oriented development

– O-O analysis - combines data and the processes that act on the data into things called objects, and similar objects can be grouped together into classes

– O-O processes are called methods

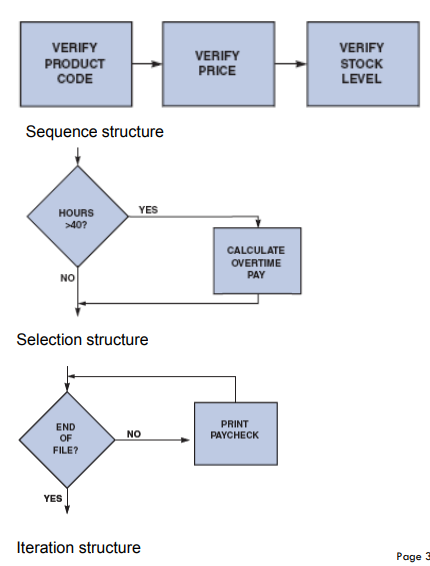
▪ **Modular Design**

– Based on combinations of three **logical structures**, sometimes called **control structures**, which serve as building blocks for the process

• Sequence

• Selection

• Iteration – looping



▪ **Structured English**

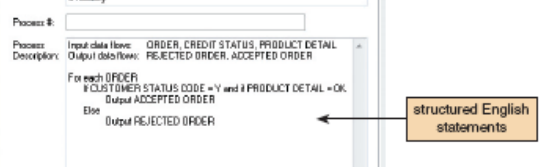
– Rules

• Use only the **three building blocks of sequence, selection, and iteration**

• Use indentation for readability

• Use a limited vocabulary

– standard terms used in the data dictionary

– Specific words that describe the processing rules

▪ **Decision Tables**

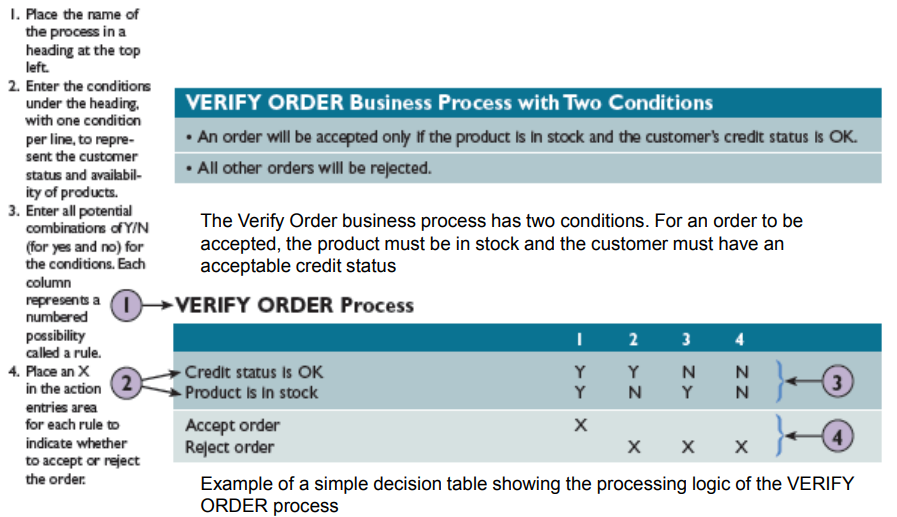
– Show a logical structure, with all possible combinations of conditions and resulting actions

• Every possible outcome should be considered to ensure that nothing has been overlooked

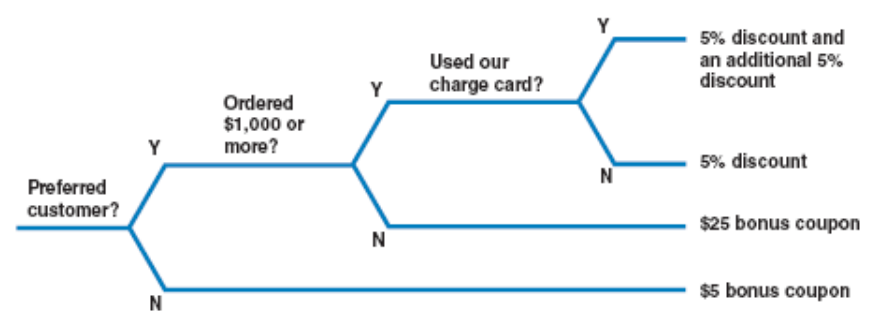
– Number of rules doubles each time a condition is added

– Can have more than two possible outcomes

– Are the best way to describe a complex set of conditions



▪ **Decision Trees**



**Logical and Physical Models**

▪ **Logical Model**: Shows **what the system must do**, regardless of how it will be implemented physically (what)

▪ **Physical Model**: Describes **how the system will be constructed(how)**

▪ While structured analysis tools are used to develop a logical model for a new information system, such tools also can be used to develop physical models of an information system(开发logical的软件也可以开发physical)

▪ Sequence of Models

– Systems analysts **create a physical model of the current system and then develop a logical model of the current system before tackling a logical model of the new system**

**(先how后what)**

• Performing an extra step allows to understand the current system better

▪ **Four-Model Approach**

– Develop:

• A physical model of the current system

• A logical model of the current system

• A logical model of the new system

• A physical model of the new system

– Disadvantage - Additional time and cost

Week7 Object Modeling

Learning Objectives

▪ Explain how **object-oriented analysis** can be used to **describe an information system**

▪ Define **object modeling terms and concepts**, including **objects, attributes, methods, messages, classes, and instances**

▪ **Describe Unified Modeling Language (UML)** tools and techniques including – **use cases, use case diagrams, class diagrams, sequence diagrams, state transition diagrams, and activity diagrams**

Overview of Object-Oriented Analysis

▪ O-O methodology is popular because it **integrates easily with object-oriented programming languages such as Java**

▪ Programmers like **O-O code** because it is **modular, reusable, and easy to maintain**

▪ The end product of O-O analysis is an object model

▪ **Object model**: Represents the information system in terms of **objects and OO concepts**

▪ Object-Oriented Terms and Concepts

– Unified modeling language (UML)

• Method of visualizing and documenting an information system

– **Attributes: Characteristics that describe an object**

– **Methods: Tasks or functions that the object performs**

**– Message: Command to perform a specific function**

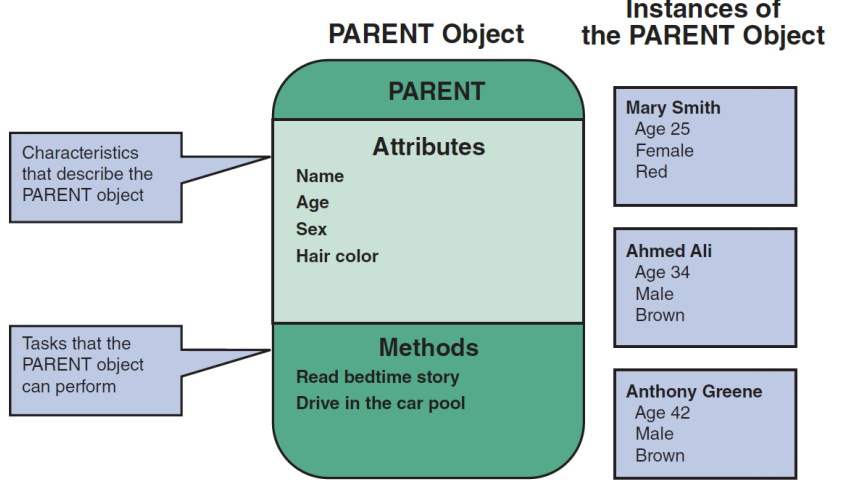
**– A class is a group of similar objects**

**• Instance: Specific member of a class**

▪ **Objects**

– Represented as a rectangle

• The object **name is at the top, followed by the object’s attributes and methods**



▪ **Attributes**

– Describe the characteristics of an object

– The **number** of attributes required depends on:

• **Business requirements** of the information system

• **Requirements of users**

– **Attributes** of an object are defined **during the system development process**

– Objects possess a **state**

• **State**: Describes the **object’s current status**

▪ **Methods**

– **Specific tasks** that an object can perform

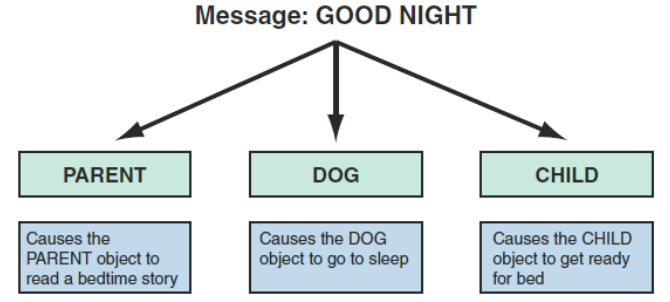
– **Identify** **functions** performed

– **Describe the functions** performed

▪ **Message**

– A **command** that **tells an object to perform a certain method**

– **Polymorphism**: Concept that **a message gives different meanings to different objects**



– A message to the object **triggers changes within the object without specifying how the changes must be carried out**

• An **object** can be viewed as **black box**

– **Encapsulation**: Idea that all **data and methods are self-contained**, as in a black box

▪ **Classes**

– **An object belongs to a group or category called a class**

• All objects within a class share **common attributes and methods**

– **Subclasses**: Categories within a class

– **Super-class**: A class belonging to a general category

Relationships Among Objects and Classes

▪ Relationships

– Enable objects to communicate and interact as they perform business functions and transactions

– Describe what objects need to know about each other

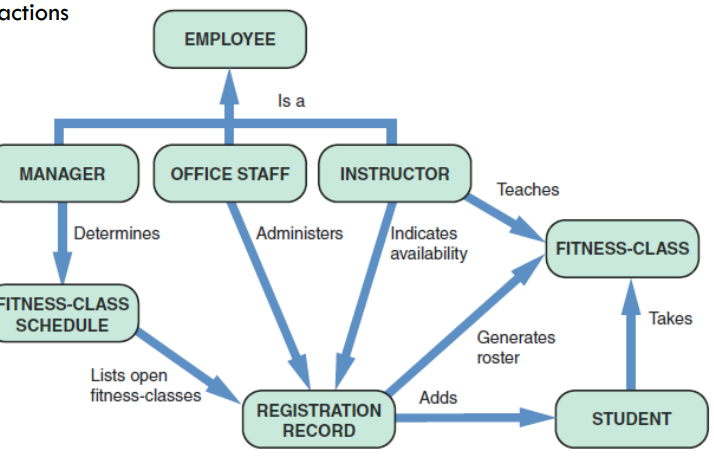
▪ Inheritance （继承）

– The strongest relationship

– Enables an object to derive one or more of its attributes from another object

▪ Object Relationship Diagram

– Displays objects and how they interact to perform business functions and transactions



Object Modeling with the Unified Modeling Language

▪ **Use Case Modeling** (用例模型)

– **Use case**: Represents the **steps in a specific business function or process**

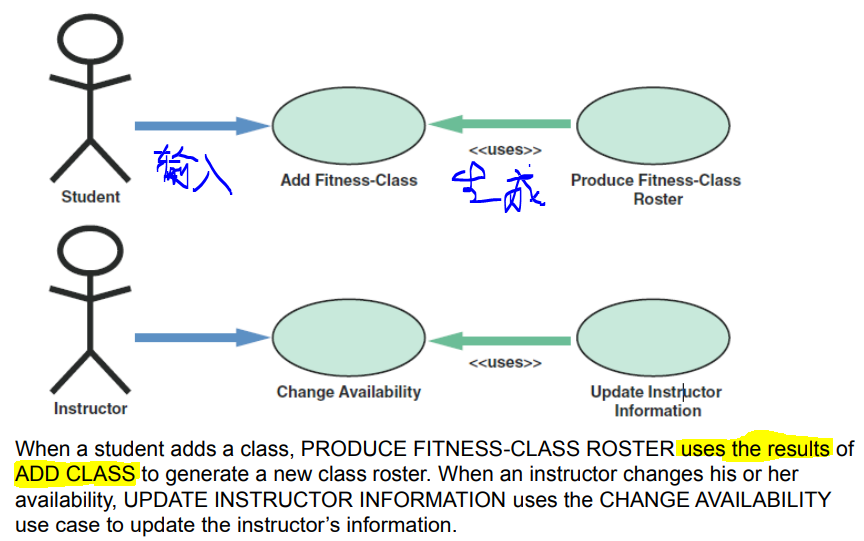
– An **external entity**, called an **actor**, **initiates a use case** by **requesting the system to perform a function** or process

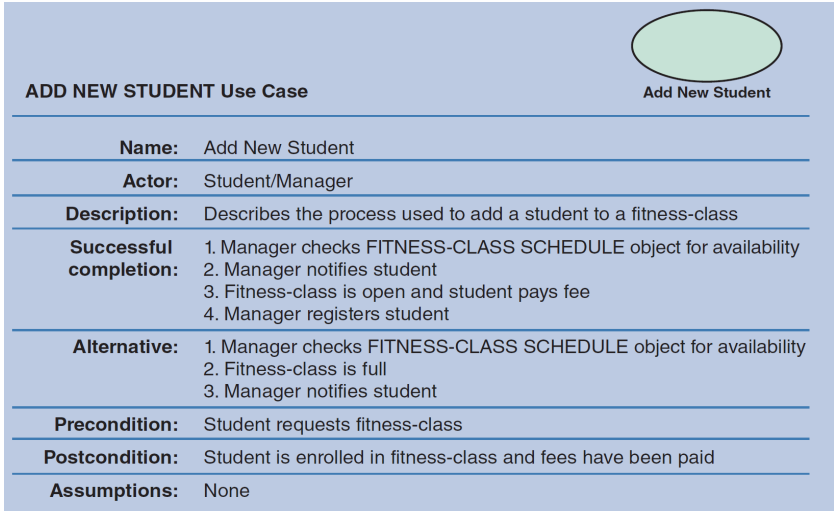
– The **actor** is shown as a **stick figure**, with a **label that identifies the actor’s role**

– **Use case description**: Documents the **name** of the use case, the **actor**, a **description** of the use case

• Provides a **step-by-step** **list** of thetasks and other key descriptions and assumptions

例子：





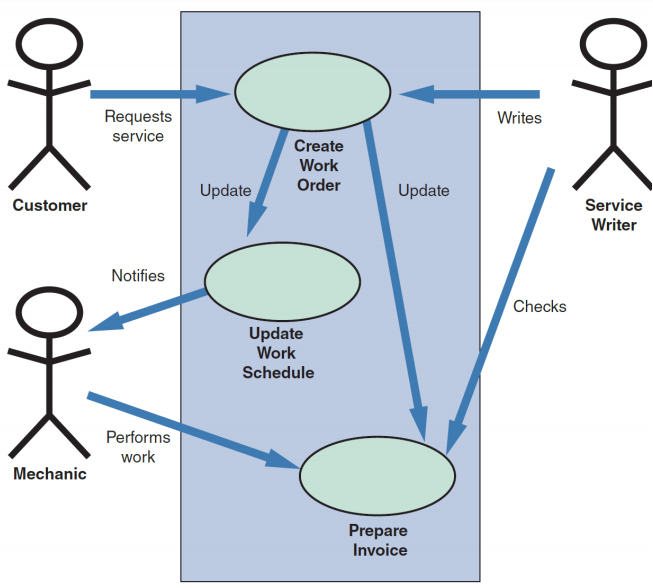
**▪ Use Case Diagrams** （用例图）

– A **visual summary** of several **related use cases** within a system or subsystem

– The first step is to identify the **system boundary** which is represented by a **rectangle**

• **System boundary**: Shows what is **included in the system** (inside the rectangle) and what is **not included in the system** (outside the rectangle)

例子：



▪ **Class Diagrams** （类图）

– Show the object **classes and relationships involved in a use case**

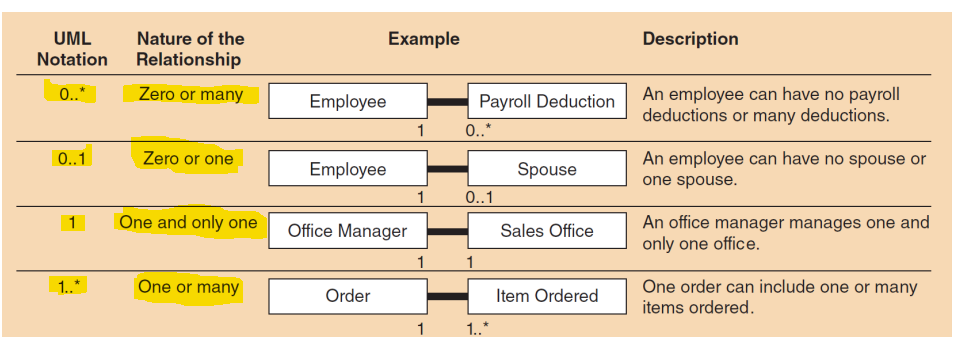
– Each **class** appears as a **rectangle**, with the **class name** at the top, followed by the class’s **attributes and methods**

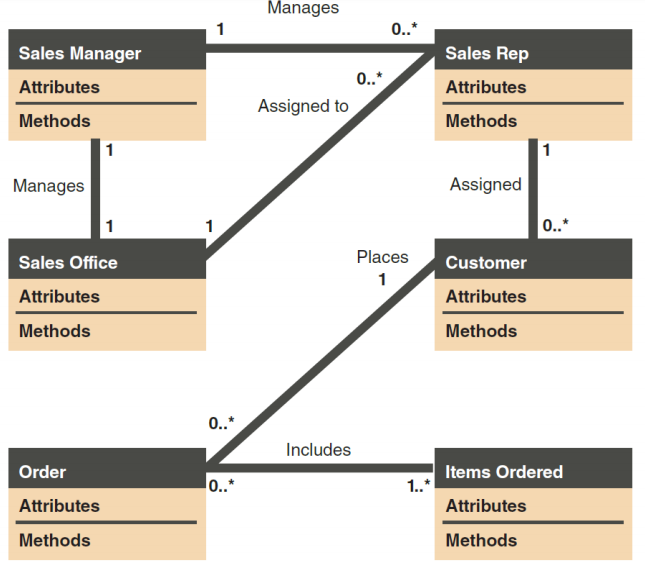
– **Lines** show relationships between classes and have **labels** identifying the action that relates the two classes

– Includes a concept called cardinality

• **Cardinality**: Describes how **instances of one class relate to instances of another class**

**图示：**



**例子：**

（class diagram）

▪ **Sequence Diagrams** （序列图）

– Dynamic **model of a use case**, showing the **interaction among classes** **during a specified time period**

– Graphically document the use case by **showing** the **classes, the messages, and the timing of the messages**

– Include symbols that represent classes, lifelines, messages, and focuses

– Classes

• Send or receive messages

– Shown at the **top** of the sequence diagram

– Lifelines

• Represent the time during which the object above it is able to interact with the other objects in the use case

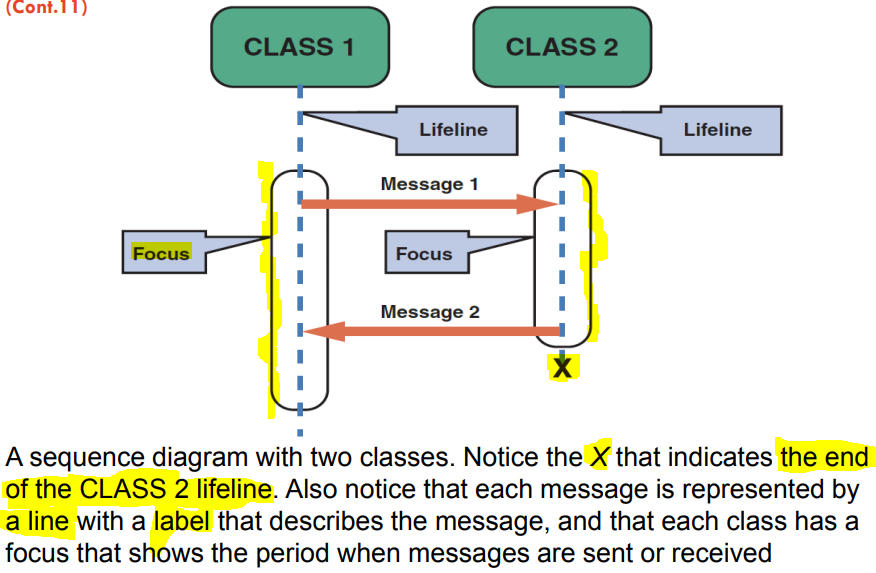
• An **X marks** the end of the lifeline

– Messages

• Include **additional information** about the contents

– Focuses

• Indicate **when** an object sends or receives message

**例子：**

**(sequence diagram)**

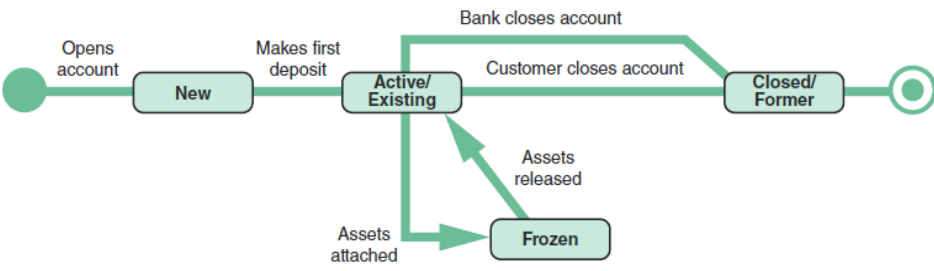
▪ **State Transition Diagrams** （状态转移图）

– Show how an **object changes from one state to another**, depending on **events** that affect the object

– **All possible states** must be documented in the state transition diagram

– States appear as **rounded rectangles** with the state **names inside**

**例子：**



**（状态转换图）**

Week8 User Interface Design

Learning Objectives

▪ Explain the **concept of user interface design** and **humancomputer interaction**

▪ Discuss **Habits of Successful Interface Designers**

▪ Explain **Guidelines for User Interface** Design

▪ Discuss **technology trends**

▪ Discuss **challenges of designing for different devices**

**What is User Interface (UI)?**

▪ The user interface is the **part of the system that you can see, hear and feel**.

▪ Describes **how users interact with a computer system**

▪ **Comprises features that affect two-way communications between the user and the computer**

▪ A physical space that allows human-computer interaction (HCI)

User Interface -- **Usability**

▪ How well the UI is designed affects the usability of the system.

▪ **Usability**: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

User Interface -- **Interface Types**

▪ **Visual interface**: Graphical User Interface (GUI)

– graphics, colour, typography, etc.

– GUI elements: buttons, icons, prompts, display cards, menus, scroll bars, form elements, etc.

▪ **Auditory interface**: Voice user interface design

▪ **Non-traditional interfaces**: e.g. Motion detection. Gestures. Voice recognition etc.

**Interaction Design**

▪ Designing interactive products to support the way people communicate

– **UI design goes with interaction design** in order to support **usability**.

▪ it can be:

– Human-Computer Interaction (HCI)

– Machine to Machine interaction

– Human-Robot Interaction

– Animal-Computer Interaction

**Designing for users**

▪ **Human-centred design**: Design should support the **user’s needs**.

– **Empathise** with the **user**

– This includes designing for **different** groups of **people**.

▪ **Design for accessibility**: Support usability for users with **reduced abilities**

e.g. short sighted, colour blind, mental and learning disabilities.

▪ **Design for different languages and cultures**: Design for different

– Languages: Scripts, accents, writing direction, etc.

– Culture: design for different social protocols, verbal and non-verbal behaviour, etc.

**Habits of Successful Interface Designers**

} **Understand the Business**

◦ The interface designer must understand:

The underlying business functions

How the system supports individual, departmental, and enterprise goals

} **Maximize Graphical Effectiveness**

◦ A well-designed interface enables rapid learning

} **Think Like a User**

◦ The designer must see the system from a user’s perspective

▪ **Use Models and Prototypes**

– Designers can present initial screen designs to users in the form of a **storyboard**

▪ **Focus on Usability**

– Include main options in the opening screen

– Offer a reasonable number of choices that a user easily can comprehend

▪ **Invite Feedback**

– Monitor system usage and solicit user suggestions

– Determine if system features are being used as intended by observing and surveying users

▪ **Document Everything**

– Document all screen designs for later use by programmers

– User-approved sketches and storyboards can be used to document the user interface

**Guidelines for User Interface Design**

} **Create an Interface That Is Easy to Learn and Use**

◦ Focus on **system design objectives**

◦ Create a design that is **easy to understand and remember**

◦ Provide **commands, actions, and system responses** that are **consistent and predictable**

◦ Allow users to **correct errors easily**

◦ Clearly **label all controls, buttons, and icons**

– Select **familiar images** that users can understand

• Provide on-screen instructions that are logical, concise, and clear

– Show all commands in a list of menu items

• Dim any commands that are not available to the user

– Make it **easy to navigate** or return to any level in the menu structure

▪ **Enhance User Productivity**:make user customise

– If available, consider a **natural language** feature that allows users to type commands or requests in normal text phrases

▪ **Provide Users with Help and Feedback**

– Ensure that help is always available on demand, and information is context-accurate

▪ **Create an Attractive Layout and Design**:highlighting and colour pattern, no complex terms

▪ **Enhance the Interface**

– Opening screen is important as it introduces the application

• The starting point can be a **switchboard** with wellplaced command buttons for navigation

– Use a **command button** to initiate an action

– Try to create customized **menu bars** and toolbars

– Add a shortcut feature that lets a user select a **menu command**

– If variable input data is needed, provide a **dialog box** that explains what is required

– A **toggle button** makes it easy to show on or off status

– Use **list boxes** that display the available choices

– Use an **option button**, or a **radio button**, to control user choices

– If **check boxes** are used to select one or more choices from a group, show the choices with a checkmark or an X

– When dates must be entered, use a **calendar control**

▪ **Focus on Data Entry Screens:**restrict user assess;allow user to change/skip before confirm.

– Use the **form filling** method whenever possible

– Display a sample format like MMDDYY and use an **input mask**

▪ **Use Validation Rules**

– **Sequence check**: Used when the data must be in some predetermined sequence

– **Existence check**: Applies to mandatory data items

– **Data type check**: Tests to ensure that a data item fits the required data type

– **Range check**: Used to verify that data items fall between a specified minimum and maximum value

– **Reasonableness check**: Identifies values that are questionable, but not necessarily wrong

– **Validity check**: Used for data items that must have certain values

– **Combination check**: Performed on two or more fields to ensure that they are consistent or reasonable when considered together

– **Batch controls**: Totals used to verify batch input

▪ **Reduce Input Volume**: only necessary data; no calculated/pre-exist data; no constant data

**Source Document and Form Design**

▪ **Garbage in, garbage out (GIGO)**: Quality of the output depends on the quality of the input

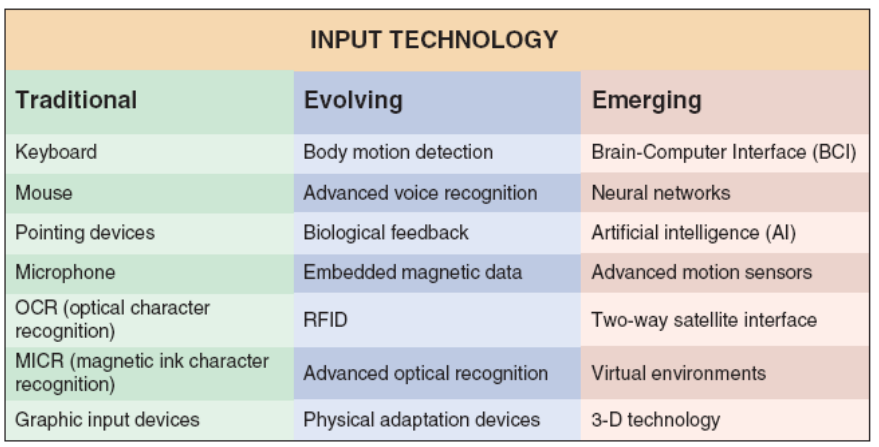
▪ **Source document**: Collects input data, triggers an input action, and provides a record of the original transaction

▪ A good **form layout** makes the form easy to complete and provides enough space

– Information should flow on a form from left to right and top to bottom

**Tech Trends**

Technology Issues: technology is rapidly evolving and with the most disruptive potential.



**Technology Trends: AI**

▪ Dictionary definitions: being human-like rather than becoming human

▪ Build systems that think exactly like humans do (“**strong AI**”)

▪ Just get systems to work without knowing how human reasoning works (“**weak AI**”)

▪ Use **human reasoning** as a **model** but not necessarily the end goal

**Technology Trends: VR**

▪ Immersive experience that provides the opportunity to be digitally transported to a different, place, time or environment.

▪ Comprised of 360 video or rendered content (like video game).

▪ physical objects/controllers that influence the digital experience

**Technology Trends: AR**

▪ Immersive experience that provide the opportunity for digital content, images, 3D models, etc. to render themselves on top / in and **around your real, physical space**.

▪ Is generally comprised of rendered content, but video concepts look promising.

▪ **Content in context** to one’s environment, task, activity.

**Technology Trends: Voice Interaction Technology**

▪ Voice as an interface is fast becoming a major piece of the digital ecosystem.

▪ Takes advantage of advances in **AI**, **speech recognition** and **Natural language processing**

▪ Amazon's Alexa, Google Home, Apple’s Siri, Microsoft’s Cortana etc.

▪ Warnings: **– Alignment of values – Transparency. – Authenticity.**

**Technology trends: Self-Driving Cars**

▪ Autonomous vehicle: A driverless vehicle capable of capable of sensing its environment and navigating without human input.

▪ Objectives: Navigate to a given destination based on passenger-provided instructions

▪ Avoid environmental obstacle

▪ Safely avoid other vehicles

▪ Obey the laws of the road

▪ **Ethical challenges**

**Challenges of designing for different devices**

▪ **Desktop Applications** (operating system specific and strict hardware requirements)

▪ **Web Applications** (client-server model, wide storage and processing resources, interactive, media-rich interfaces)

▪ **Mobile Applications** (limited storage, processing and battery)

▪ Mobile: design challenges

– Slow and Error-Prone Typing

– Less Context

– Inaccurate Clicks

– Poor Connectivity

– Slow Hardware

– Less Storage Capacity

**Tools and Techniques**

▪ **User research**- To find out the needs of the users to create new systems or improve existing systems. Some common techniques include

– ethnography

– user analytics

– usability testing and evaluation (A/B testing)

– interviews, etc.

▪ **Usability heuristics**: Heuristics used to evaluate the UI and interaction design.

▪ **Prototyping**: A model or representation of the system. Can range from low to high level functionality.

– **Low-fidelity prototype** - Mostly includes 2D mock-ups of the system .Low technical skills required. No functionality.

– **Medium-fidelity prototype** - Has the visual look and feel, portrays most of the interactions between the user and the system. No to very little functionality.

– **High-fidelity prototype** - Full or some functionality. Needs a full understanding of technical skills like coding.

Week9 Data Design

**Learning Objectives**

▪ Discuss **file-oriented systems** and **how they differ from database management systems**

▪ Explain **data design terminology**— **entities, fields, records, files, tables, and key fields**

▪ Describe **data relationships** and **Entity Relationship Diagram**

Data Design Concepts

▪ Data Structures

– Framework for organizing, storing, and managing data

– Comprises of files or tables that interact in various ways

• Each **file** or **table** contains data about people, places, things, or events

**file-oriented systems**

* **Different files** store data about **different things**
* **Inefficient, can produce errors**
* **Cost-effective in some situation**

**Relational model**

* Tables are linked by a **common field**
* Data is like in **one large table**

The Database Environment

▪ **Database management system (DBMS)**: Collection of **tools, features, and interfaces** that enables users to **add, update, manage, access, and analyze** data

▪ **DBMS advantages**

✓ **Scalability** - System can be expanded, modified, or downsized

✓ **Economy of scale** • Database design allows better utilization of hardware

✓ **Enterprise-wide application**

✓ **Stronger standards**-Standards are followed uniformly

✓ **Better security**– Different users have different levels of access

✓ **Data independence**-independent of how physical data is maintained

**Data Design Terms**

◦ **Entity** - Person, place, **thing**, or event for which **data is collected** and maintained(名词)

◦ **Table** or file: Contains a **set of related records** that **store data about a specific entity**

◦ **Field** (attribute) - Single characteristic or **fact about an entity**

◦ Tuple (**record**): **Set of related fields** that describes one instance, or occurrence, of an entity

▪ Key Fields

– **Primary key**: Field or combination of fields that uniquely and minimally identifies a particular member of an entity • Called a **combination key** （识别实体的键）

– **Candidate key**: Any field that could serve as a primary key （可作为主键）

– **Foreign key**: Field in **one table** that **must match** **a primary key value in another table** for a relationship between the two tables to exist（匹配另一个表中主键的键）

▪ Referential Integrity

– Set of rules that avoids data inconsistency and quality problems

**Entity-Relationship Diagrams（ERD）**

- 方块：名词 代表entity

- 菱形：动词 代表两个方块间的relation

- 方块必须经过菱形连方块

- 若两个方块是“多对多”的关系 则中间的菱形外面套一个方块 变成“associative entity”

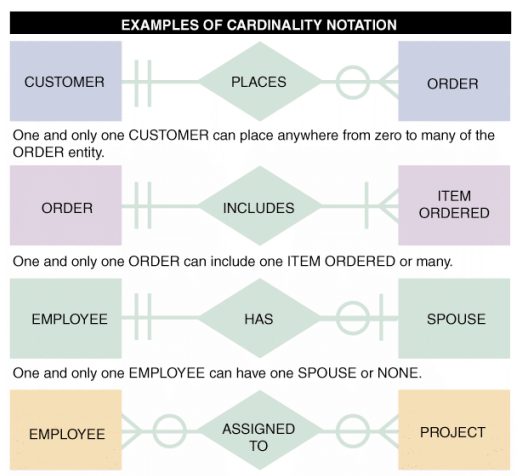
关系种类：

一对一 eg 一个项目经理管一个项目

一对多 eg 一个项目有多个组员

多对多 eg 很多个学生上很多个课（一个学生上很多课 一个课有很多学生上）

Cardinality：描述两个entity之间的数量关系（上面的）



追溯关系的时候 用贴近宾语那边的符号

Eg 第一个 一个顾客可以有 零到多 个订单（零到多 是 订单 那边的符号）

一个订单只可以对应 一个 顾客 （一个 是 顾客 那边的符号）

Week10 Systems Architecture Design

**Learning Objectives**

▪ Explain the concept of **normalization**

▪ Provide a checklist of **issues to consider** when selecting a **system architecture**

▪ Explain **client/server architecture**, including **tiers, cost-benefit issues, and performance**

▪ Compare **in-house ecommerce development** with **packaged solutions** and **service providers**

▪ Describe the **system design specification**

**Data Normalization （数据标准化）**

▪ **Normalization**: Process of **creating table designs by assigning specific fields or attributes to each table in the database （把一个表格拆成多个拥有非重复主键项的表格）**

▪ **Table design**: **Specifies fields**

◦ Identifies the primary key in a particular table or file

▪ **Standard Notation Format**

– Used to show a table’s structure, fields, and primary key

– The **primary key** field(s) is **underlined**

• NAME (FIELD 1, FIELD 2, FIELD 3)

– Recognition of repeating group fields is important

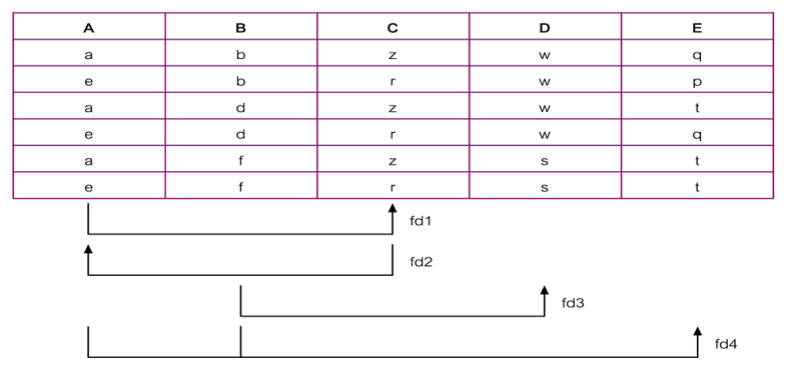
• Repeating group: Set of one or more fields that can occur any number of times in a single record

• Each occurrence would possess different values

Functional Dependencies

▪ Functional dependency describes relationship between attributes.

▪ For example, if A and B are attributes of table R, ***B is functionally dependent on A (A → B)***, if each value of ***A in R is associated with exactly one value of B in R(b依赖于a 就是a指向b 代表每个b都与一个a相对应 先有a再有b)***



Transitive Dependencies

▪ a->b, b->c, c就通过b transitive depend 于a.

**Normalization Process -- Stages**

▪ Unnormalized Form (**UNF**): A table that contains one or more repeating groups. （包括多个重复主键的表格）

▪ First Normal Form (**1NF**): A table in which the intersection of each row and column contains one and only one value. （每个格子都包括一个数据的表格）

▪ Second Normal Form (**2NF**): A table that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key. （是1NF且每个非主键全依赖主键）

▪ Third Normal Form (**3NF**): A table that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.（是1NF和2NF且没有非主键传递依赖于主键）

▪ **UNF to1NF**

– Nominate an attribute or group of attributes to act as the key for the unnormalized table. （把一或多个属性作为主键）

– Identify and remove the repeating group by entering appropriate data into the empty columns of rows containing the repeating data.（把空格填上）

**▪ 1NF to 2NF**

- Identify the primary key for the 1NF relation. （鉴别1nf中的主键）

- Identify the functional dependencies in the relation. （鉴别直接依赖关系）

- If partial dependencies exist on the primary key remove them by placing then in a new relation along with a copy of their determinant.（将主键外的依赖关系分离成新表格）

▪ **2NF to 3NF**

- Identify the primary key in the 2NF relation. （鉴别2NF中的主键）

- Identify functional dependencies in the relation. （鉴别直接以来关系）

- If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their dominant.（若传递依赖存在 则分离出新表格）

**Architecture Checklist （重点）**

▪ Issues that influence the architecture choice

– Corporate organization and culture:

– Enterprise resource planning (**ERP**):

– To establish a company-wide strategy for using IT that includes a specific architecture, standards for data, processing, network, and user interface design

– Initial and total cost of ownership (TCO)

- validity, effectiveness, and new trends

– Scalability

* meet the changing needs

– Web integration

* A web-centric architecture integrate new applications

– Legacy system interface requirements

* interface with old systems for data and compatibility

– Processing options

* process data online or in batches

– Security issues

– Corporate portals

* access for customers, employees, suppliers, and the public

Client/Server Designs

▪ **Client/Server Architecture**

◦ Includes systems that divide processing between **one or more networked clients** and **a central server**

**Client** handles the entire **user interface**

**Server stores data** and provides data access and **database management functions**

– **thick client** : Locates all or most of the application processing **logic at the client**

– **Thin client** : Locates all or most of the processing **logic at the server**

• Provides **better performance** as the program code resides on the server

▪ **Client/Server Tiers**

– **Two-tier design**

• User interface resides on the client

• Data resides on the server

• **Application logic** can run **either on the server or on the client**, or be divided between the client and the server

– **Three-tier** (n-tier) design

• User interface runs on the client

• Data is stored on the server

• Has **a middle layer(application server) between the client and server**

– **Processes the client requests** and **translates them into data access commands**

▪ Middleware

• Used to connect two or more software components in a federated system architecture

▪ **Cost-Benefit Issues**

– Client/server systems offer the **best combination of features** to meet information system requirements

Internet-based architecture

▪ Cloud Computing

– The concept envisions a cloud of remote computers providing a total online software and data environment that is hosted by third parties

– Eliminates compatibility issues and provides scaling on demand

**E-commerce Architecture**

▪ **In-House Solutions**

A unique website, with a **look and feel consistent** with the company’s other marketing efforts

Complete **control over the organization** of the site

A **scalable structure** to handle increases in sales and product offerings in the future

More **flexibility to modify** and manage the site

The opportunity to **integrate** the firm’s web-based business systems **with its other information systems**

▪ **Packaged Solutions**

– Viable **alternative for medium- to large-sized firms**

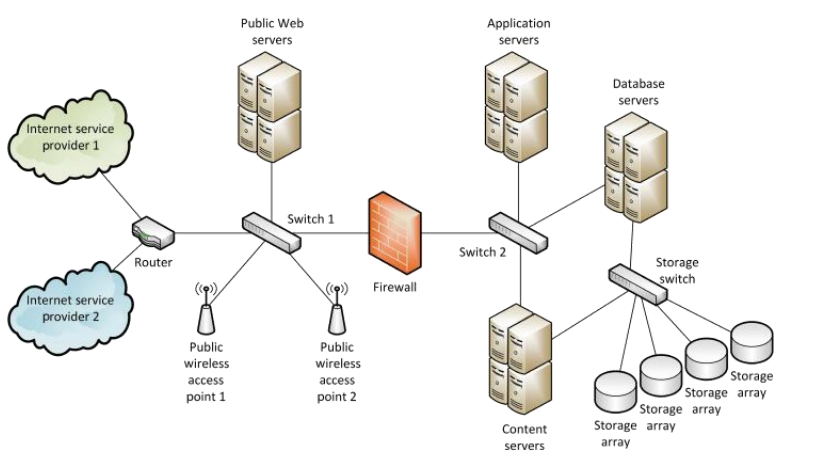
– **Less complex** than an in-house effort

▪ **Service Providers**

– Application service provider (ASP) - **Provides applications** or access to applications **by charging a fee**

• Many ASPs **offer full-scale Internet business services** for companies that decide to outsource functions

Network Diagram for System Architecture



**System Design Specification**

– Document that presents the complete design for a new information system

• Contains detailed costs, staffing, and scheduling for completing the next SDLC phase

– Used as a baseline to measure the operational system

– Sections in a system design specification

• Management summary

• System components

• System environment

• Implementation requirements

• Time and cost estimates

• Additional material

User Approval

Presentation to management

Week11 Systems Implementation

**Learning Objectives**

▪ Discuss the concept of **system implementation**

▪ Explain **unit**, **integration**, **system**, and **user acceptance** **testing**

▪ Discuss the main steps in **system installation** and **evaluation**

▪ Develop **training plans** for various user groups, compare **in-house and vendor** training options, and describe **effective training techniques**

▪ Describe **data conversion and changeover** methods

▪ Explain **post-implementation evaluation** and the **final report** to management

**Systems Implementation**

▪ System implementation: defining how the system would be built to make sure that the system is operational and meets that quality standards.

▪ System implementation uses the structure (developed during systems design) and the results of system analysis

Software Quality Assurance: International Organization for Standardization (**ISO**)

▪ Application development: constructing code modules as the building blocks of system.

**Systems Implementation Processes and Techniques**

▪ Prepare for System Implementation

▪ Deploy System

▪ Transition to Performing Organization

**Systems Implementation Plan**

▪ Components to be Implemented – break system into components

▪ The Implementation Strategy – how system will be implemented

▪ Data Conversion Strategy – how legacy data is handled

▪ Deployment Strategy – deliver system

▪ The Testing Strategy

▪ The Knowledge Transfer and Training Strategies

**Testing Concepts**

▪ **Testing**: the process of examining a component to determine its operational characteristics and whether it contains any defects

▪ **Unit Testing**

– Testing individual module

– Aims to identify and eliminate:

• Execution errors

• Logic errors

– **Test data** should contain both accurate and erroneous data

– Stub testing is required

• A test plan is created

▪ **Integration Testing**

– Testing two or more programs that depend on each other

▪ Procedures

– Build and unit test the components to be integrated

– Create test data

– comprehensive test data, must be coordinated between developers

– Conduct the integration test – Assign resources and responsibilities. Plan frequency and procedures

– Evaluate the test results – Identify valid and invalid responses

– Log the test results – Log valid test runs. Also log errors

– Correct the code and retest

▪ **System Testing**

–test of an entire system or independent subsystem

– Objectives

• Verify that the system will handle all data properly

• Ensure that the IT staff has the documentation and instructions needed to operate the system properly

• Verify that all system components are integrated properly

• Confirm that the information system can handle predicted volumes of data in a timely and efficient manner

– performance test

• Response time

• Throughput

• Business functions

• Stability

• Resource Usage

• Speed

▪ **User Acceptance Testing**

–test whether the system fulfills user requirements

– Preparation and Pre-UAT Activities

• Develop test data – data entry and database records

• Plan and schedule specific tests

• Set up test environment

– Manage and execute the UAT

• Much like a mini-project

• Assign responsibilities

• Document and track results (especially errors and fixes)

• Rework the plan for re-testing as required

▪ need management approval

– Test results should be described

– Status of all required documentation should be updated

– Input from users be summarized

– Detailed time schedules, cost estimates, and staffing requirements

**System Installation and Evaluation**

▪ Prepare a separate operational and test environment

▪ Provide training for users, managers, and IT staff

▪ Perform data conversion and system changeover

▪ Carry out a post-implementation evaluation of the system

▪ Present a final report to management

▪ **Operational** or **production** environment

◦ Environment for the actual system operation

▪ **Test** environment

◦ Environment that analysts and programmers use to develop and maintain programs

◦ A separate test environment is necessary to maintain system security and integrity and protect the operational environment

**Training**

▪ Training provided by vendor

– Required if the system includes the purchase of hardware or software

• Scope is limited to a standard version of the vendor’s software or hardware

▪ Outside Training

– Viable alternative if vendor training is not practical

▪ **Training Tips**

– Train people in groups

– Select the most effective place for training

– Provide for learning by hearing, seeing, and doing

– Rely on previous trainees

– Effective training is interactive, self-paced, and multimedia-based

– Online training

• Effective when it is more realistic

• Sophisticated training systems offer interactive sessions

• Training material should include a reference section that summarizes all options and commands, lists all possible error messages, and what actions the user should take when a problem arises

• A full-scale test, or simulation must be conducted after the training is complete

**Data Conversion**

▪ **Data Conversion**: process which existing data is loaded on to a new system

▪ Data Conversion Strategies

- Automate the data conversion process if possible

- develop a standard format, use it or convert into it

▪ Data Conversion Security and Controls

– Maintain strict input controls

– Ensure that all system control measures are in place

– Ensure that error-free data is fed into the new system

**System Changeover**

▪ Process of putting the new information system online and retiring the old system

▪ Can be rapid or slow

▪ Types

▪ Direct Cutover

– Enables changeover when the new system becomes operational

– Least expensive changeover method

▪ Parallel Operation

– Both the old and the new information systems operate fully for a specified period

– Advantage - Lower level of risk • The old system can be used as a backup

– Not practical if the old and new systems are incompatible

▪ Pilot Operation

– Implements the complete new system at a selected location of the company

– Pilot site: Group that uses the new system first

– Reduces the risk of system failure

– Less expensive than a parallel operation

▪ Phased Operation

– Implements the new system in stages, or modules

**Post Implementation Tasks**

▪ Post-Implementation Evaluation – Assesses the overall quality of the information system

– Includes:

• Accuracy, completeness, timeliness of information system output, and user satisfaction

• System reliability and maintainability

• security

• efficiency, performance

• effectiveness of database

• performance of IT team

• Completeness and quality of documentation

• Quality and effectiveness of training

• Accuracy of cost-benefit estimates and development schedules

▪ Guidelines of evaluating a system:

– Interview members of management and key users

– Observe users and computer operations personnel actually working with the new information system

– Read all documentation and training materials

– Examine all source documents, output reports, and screen displays

– Use questionnaires to gather information and opinions from a large number of users

– Analyze maintenance and help desk logs

▪ Final Report to Management

– Submitted at the end of the SDLC phase

– Includes:

• Final versions of all system documentation

• Planned modifications and enhancements to the system that have been identified

• Recap of all systems development costs and schedules

• Comparison of actual costs and schedules to the original estimates

• Post-implementation evaluation, if it has been performed