HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

# *System Analysis and Design*

IT3120E

ONE LOVE. ONE FUTURE.

**Part 3: Design modeling and system construction**

# *Chapter 7: Design models – System construction and deployment*

ONE LOVE. ONE FUTURE.

# Overview

- How the system operates
  - Hardware
  - Software
  - Network infrastructure
  - User interface
  - Forms and reports
  - Databases
  - Files
  - etc.
- ... highly related to analysis phase

**=> Design phase**

# Overview

- The main job of the design phase is to develop the analytic representations (analysis models) into design representations (design models).
- Planning and analysis step: responsible for developing a viable system.
- Design step: build design drawings for the system to serve the deployment.
- Through the design phase, the project team carefully considers the new system against the existing system and its surroundings.

# Content

- 7.1. Design Strategies
- 7.2. Class and Method Design
- 7.3. Data Management Layer Design
- 7.4. Human–Computer Interaction Layer Design
- 7.5. Physical Architecture

# 7.1. Design Strategies

- Issues to consider
  - Integration with existing systems
  - Exchange data
  - Effects of skill development
- Design Strategies
  - Custom Development
  - Packaged Software
  - System Integration
  - Outsourcing

# Custom Development

- Allows for meeting highly specialized requirements
- Allows flexibility and creativity in solving problems
- Easier to change components
- Builds personnel skills
- May add tax firm’s resources
- May add significant risk

# Packaged Software

- Software already written
- May be more efficient
- May be more thoroughly tested and proven
- May range from components to tools to whole enterprise systems
- Must accept functionality provided
- May require change in how the firm does business
- May require significant “customization” or “workarounds”

# System Integration

- The process of combining packages, legacy systems, and new software
- Key challenge is integrating data
- Write data in the same format
- Revise existing data formats
- Develop “object wrappers”

# Outsourcing

- Hire external firm to create system
- May have more skills
- May extend existing resources
- Never outsource what you don’t understand
- Carefully choose vendor
- Prepare contract and payment style carefully

# Selecting a Design Strategy

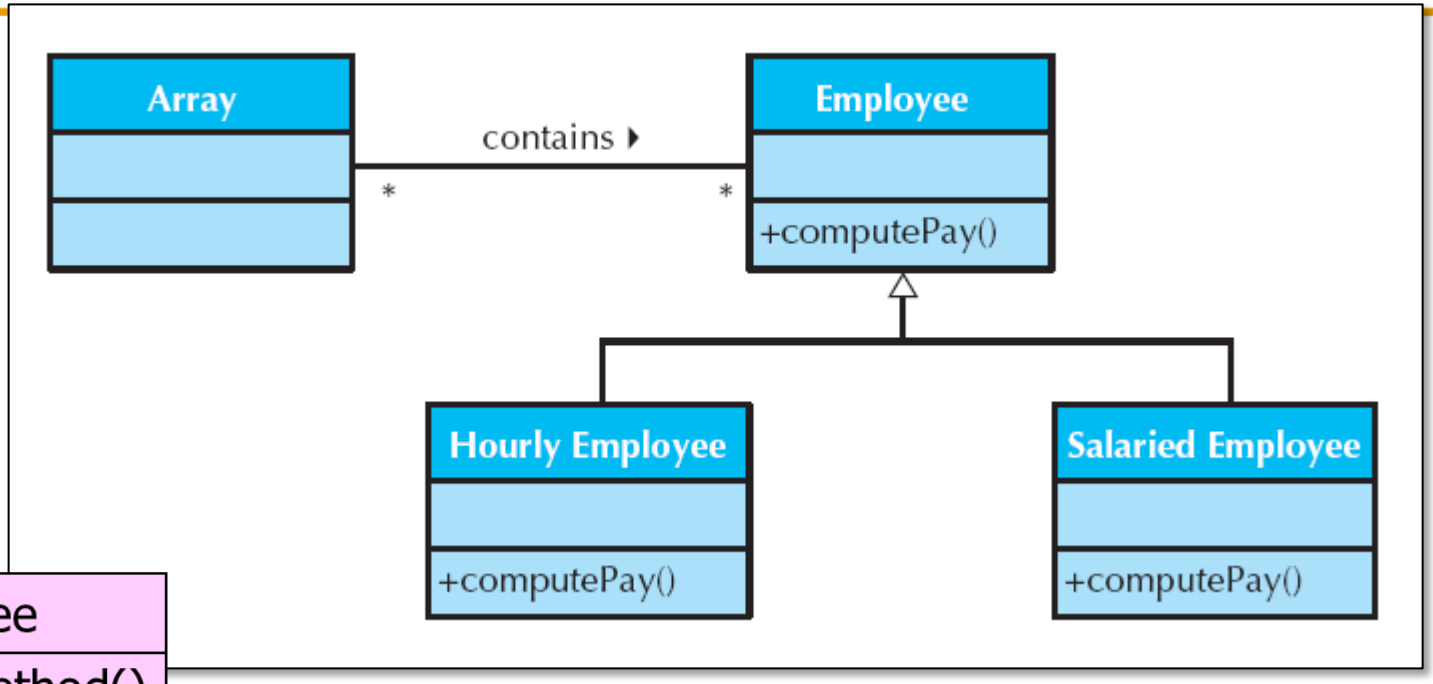
| | Use Custom Development when… | Use a Packaged System when… | Use Outsourcing when… |
|---|---|---|---|
| **Business Need** | The business need is unique | The business need is common | The business need is not core to the business |
| **In-house Experience** | In-house functional and technical experience exists | In-house functional experience exists | In-house functional or technical experience does not exist |
| **Project Skills** | There is a desire to build in-house skills | The skills are not strategic | The decision to outsource is a strategic decision |
| **Project Management** | The project has a highly skilled project manager and a proven methodology | The project has a project manager who can coordinate vendor's efforts | The project has a highly skilled project manager at the level of the organization that matches the scope of the outsourcing deal |
| **Time frame** | The time frame is flexible | The time frame is short | The time frame is short or flexible |

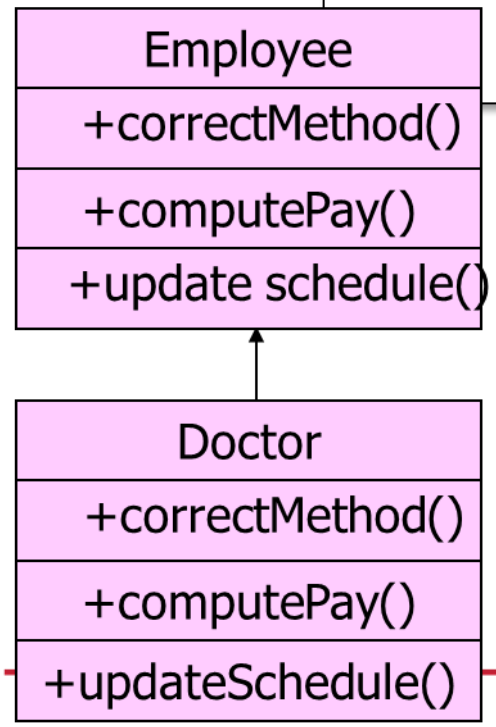
# 7.2. Class and Method Design

- Elements of OOAD: Classes, Objects, Attributes, States, Methods, Messages
- In OOT systems, changes can take place at different levels of abstraction: variables (instances), behavior (methods), classes and objects, packages, libraries, and applications.
- Changes at different levels of abstraction can affect other levels of abstraction.
- Changes can occur at different levels concurrently.
- The initialization problem domain classes must be designed correctly and completely.

- Encapsulation & Info Hiding:
  - Hiding the content of the object from outside view
  - Communication only through object's methods
  - Key to reusability
- Polymorphism & Dynamic Binding
  - Same message triggers different methods in different objects
  - Dynamic binding means specific method is selected at run time
  - Implementation of dynamic binding is language specific
  - Need to be very careful about run time errors
  - Need to ensure semantic consistency
- Inheritance
  - Single inheritance -- one parent class
  - Multiple inheritance -- multiple parent classes
  - Redefinition and inheritance conflict
  - Most inheritance conflicts are due to poor classification

# Example



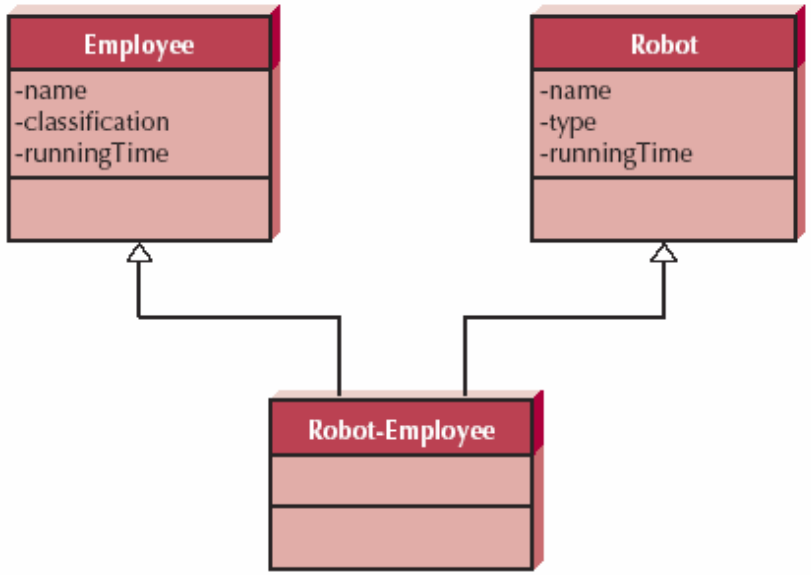




+ If Employee, i.e. superclass is modified, all of its subclasses are affected

+ Subclass can "cancel" inheritance of methods from its superclass by redefinition of inherited method

- Redefining can lead to a conflict.
- Inheritance conflicts are caused by bad classification
  - Generalization structure are compromised.
  - Inheritance mechanism violates the principle of encapsulation. For example, subclasses have direct access to the properties and methods of the parent classes.
- “Rumbaugh’s Rules” :
  - Avoid redefining query operations.
  - Methods should be redefined the semantics of the inherited methods.
  - The basic structure of the inheritance method should never be changed: the list of elements (parameters) of the inherited method should not be changed.

- Multiple Inheritance: a subclass can inherit multiple parent classes
- Types of inheritance conflicts increase.
- Three types of inheritance conflicts:
  - Two inherited methods or properties have the same name and semantics.
  - Two inherited methods or properties have different names but same semantics (synonyms).
  - Two inherited methods or prop
    different semantics (homonym



- Connectivity: shows the dependencies and internal relationships of the modules in the system (classes, objects, and behaviors).
  - Interactive coupling: resolves the internal connection between methods and objects through message exchange
- Cohesion: how to resolve a module inside a system

- Interactive coupling
  - Minimizes the number of objects that can receive messages from a given object.
  - Each object should only send messages to:
    - itself
    - The object is contained in one of its properties or an object of its superclass.
    - An object exchanged as a parameter to the method.
    - An object created by the method.
    - An object is stored in a global variable.
  - Connections will increase if:
    - The calling method passes properties to the called method.
    - The calling method depends on the response value of the called method

- Cohesion: how to resolve a module inside a system
  - Class cohesion: Cohesion between the properties and methods of the class (how to organize a class?).
  - A class should represent only one problem (employee, order...)
  - The properties and methods in the class must represent the problem.
- An ideal class cohesion:
  - Contains methods that are visible from outside the class
  - Each method performs a function
  - Add methods that reference to properties (getter/setter)
  - Add overloaded methods if needed
  - There is no control flow connection between methods

- Make sure that the classes are necessary and sufficient for solving the problem.
  - No missing (lost) properties or methods
  - No extra, unused properties/methods in each class.
- Complete visibility (hidden or visible) of the properties or behavior in each class.
- Decide correctly the signatures of all methods and classes.
  - The signature of a method: name of the method, # of arguments and the type of arguments that must be passed to
- Define the constraints that it must maintain by the objects
  - For example, the value of the property must be in a specified range.
  - There are 3 different types of constraints: pre conditions, post conditions and immutable
  - Need to determine how to deal with constraint violations (error control mechanism).

- Pattern
  - A useful combination of classes that work together to provide a solution to a common problem.
  - Analytical patterns: the main purpose is to represent the problem domain
  - Design patterns: serve a general design problem in a specific context
- Frameworks
  - Set of classes that have been deployed and can be reused as the basis for deploying an application.
  - Most frameworks allow the creation of subclasses that extend the classes in the framework.
  - The generated subclasses will have dependencies, the inheritance connection will increase between the parent class and the child class. Therefore, it will be necessary to recompile when the framework is changed by the vendor.
- Library:
  - A class library: a set of implemented classes designed for reuse.
  - The library can be devoted to support for statistical processing, arithmetic, file management, user interface development.
- etc.

- Factoring
  - Separate out shared aspects of a method or class into a new method or class
  - New class related to existing classes via inheritance, aggregation or association
- Packaging
  - Group of logically related classes/elements.
  - Define also the interdependencies between classes
- Normalization
  - Identifies classes missing from the design
  - All association and aggregation relationships converted to attributes of the class
- Need to resolve any conflicts between constructs assumed in the design to be available in programming language
  - [1] design uses multiple inheritance, language has only single inheritance
  - [2] implementation language is *object-based* and not class-based
  - [3] language is of another (less expressive) language paradigm

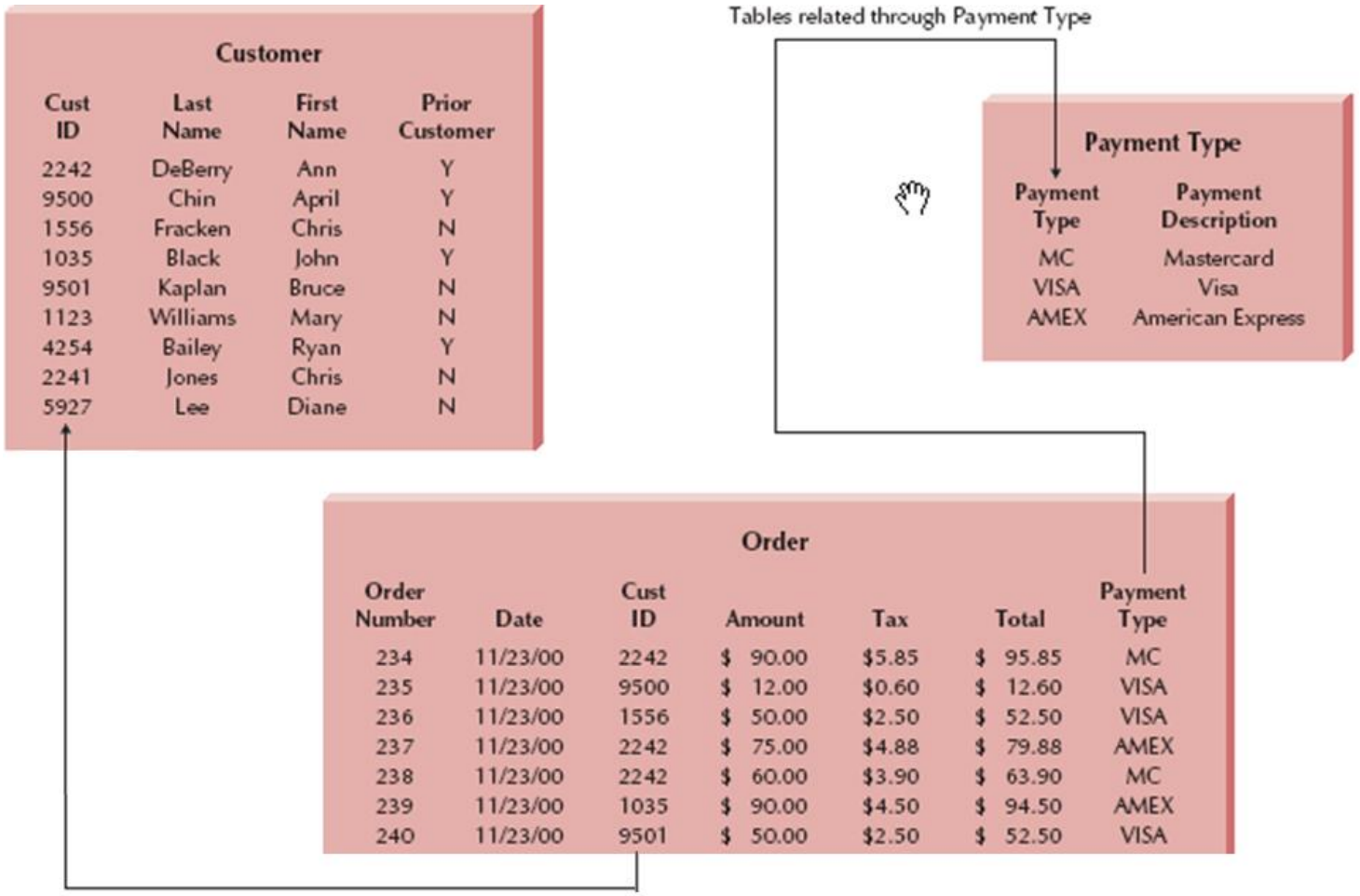
# 7.3. Data Management Layer Design

- Data management layer includes data access and management logic along with the actual design of the memory.
- The data storage component of the layer manages how the data stored and processed by the system.
- Designing the data management layer is a 4-step process:
  - Select storage format
  - Mapping domain classes to selected storage format
  - Optimize memory for efficient operation
  - Design integrated data access classes and manipulation classes.

- Files (Sequential and Random)
  - contain lists of data, e.g. bytes, characters, strings, integers, records(which contain fields), etc.
  - may be used as *masterfiles, look-up files, transaction files, audit files, history files.*
  - inefficient for searching, updating, reporting.
- More effective if both sequential AND random processing to use a *database*
  - Relational databases
  - Object-relational databases
  - Object-oriented databases

- Relational databases - **R**elational Database Management System
  - Based on a set of tables, each table containing a primary key
  - Tables that are related to each other by placing the primary key from one table to another table as foreign key.
  - Database management systems all support referential integrity, ensuring that the values that link tables together are correct and consistent.

- Relational databases



- Object-Relational database
  - a relational database with extensions for handling object storage in a relational table structure.
  - don’t support most object oriented features
- Object-Oriented database
  - data objects are stored with all of their properties in the database
  - to facilitate the storage and retrieval of object-oriented data (keep object data persistence).

# 7.3. Data Management Layer Design
## *Selecting Storage format*

- Files
  - Efficient for specific task
  - Redundant data
- RDBMS
  - Commercial technology (especially for large data sets)
  - Supports several types of data (but only data of simple types and restricted range of data)
  - No support for object orientation
- ORDBMS
  - Inherit RDBMS strengths
  - Supports more complex user-defined types
  - Limited support for object-orientation
- OODBMS
  - Support all object types (built-in and user-defined types)
  - Support object-orientation directly

*Selecting Criteria: (Data) types to be supported, Types of application, Existing storage formats, Future needs, Other (cost, concurrency control, …)*

- [Suppose that RDBMS is to be used]
- Map all concrete domain classes to RDBMS tables
  - Map single valued attributes to columns of the tables
  - Map methods to stored procedures or to program modules
  - Map single-valued aggregation and association relationships to a column that can store the key of the related table
  - Normalize data tables

- [Suppose that RDBMS is to be used]

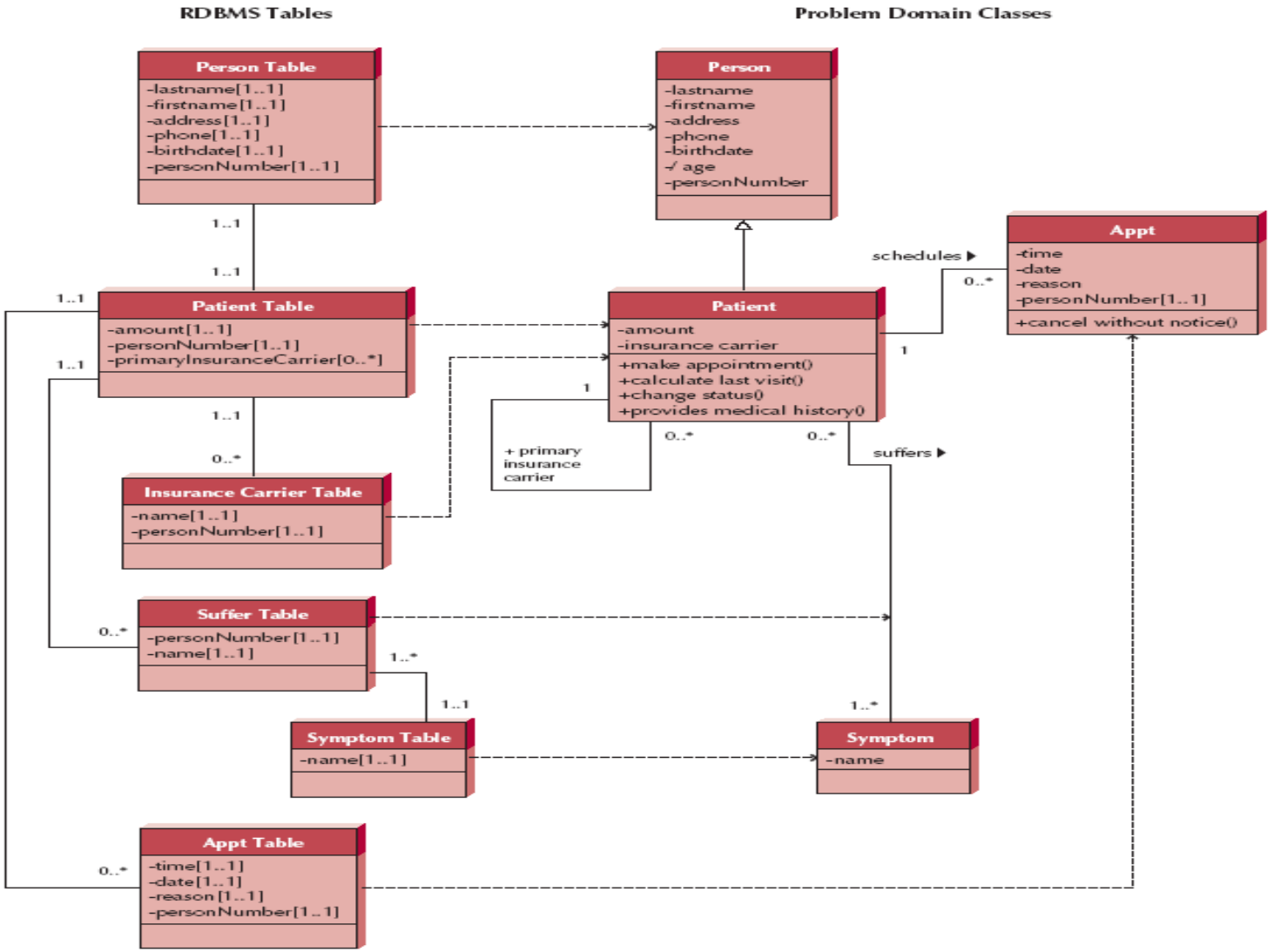



FIGURE 11-10 Mapping Problem Domain Objects to RDBMS Schema Example

- [Suppose that OODBMS is to be used]
- Each concrete domain class has corresponding object class
- Each object class has a DAM (Data Access and Management) class that manages interaction between object class and domain class
  - allows reading and writing data from/to the OODBMS
  - cannot store or invoke instances of the domain class.
  - This makes the domain classes independent of the OODBMS used.
- If multiple inheritance is used in the problem domain but not supported by OODBMS -> Multiple inheritance must be fragmented into OODBMS classes

- [Suppose that OODBMS is to be used]

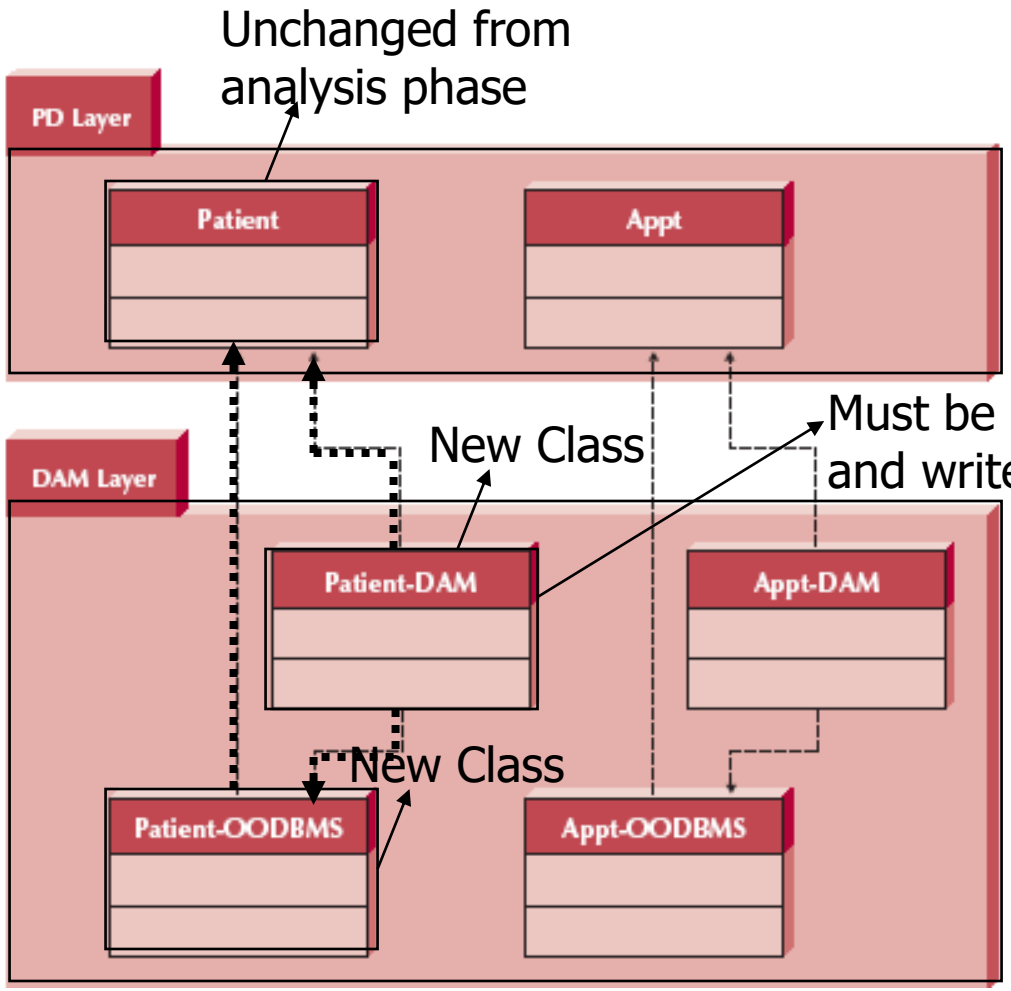
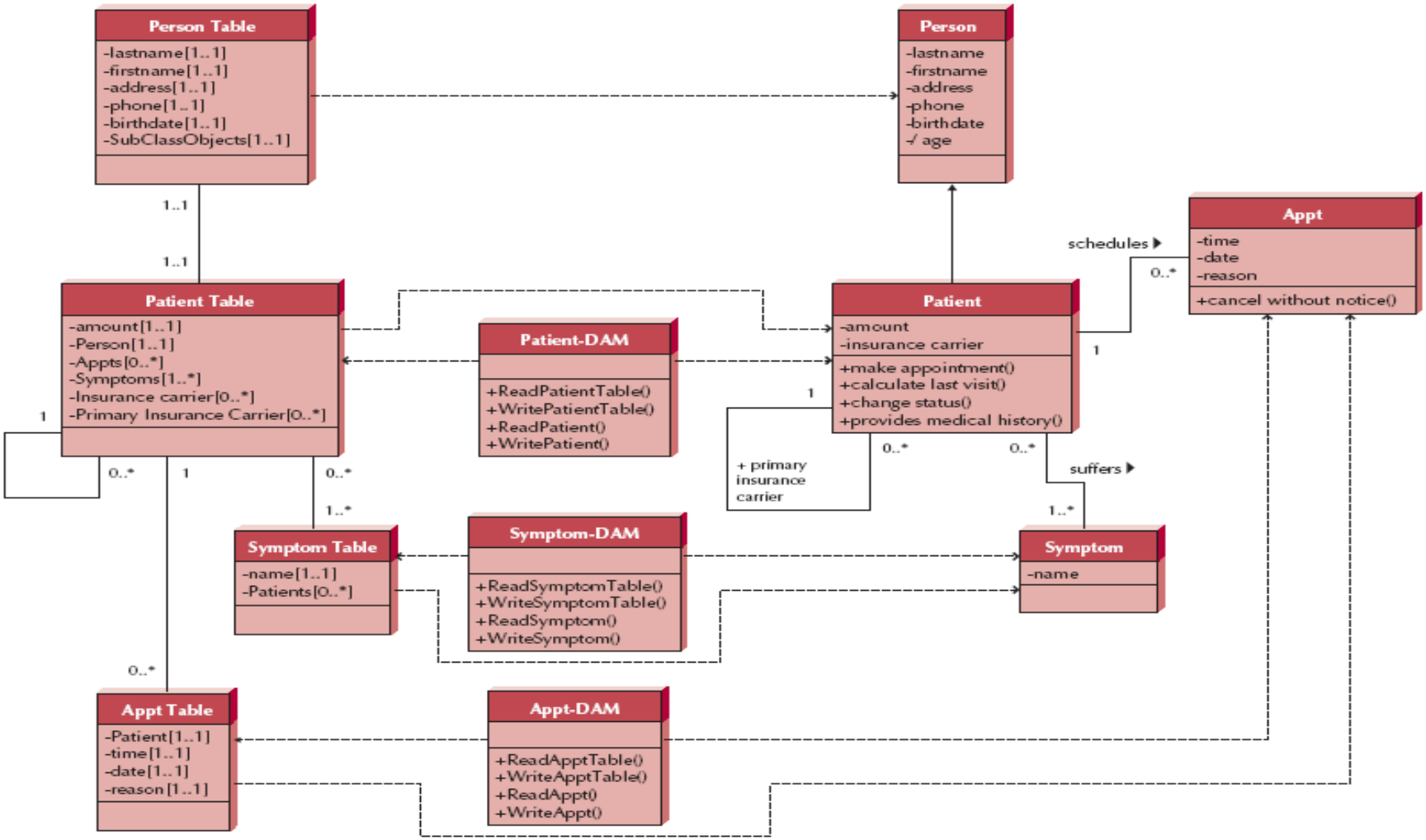



FIGURE 11-5
Appointment System Problem Domain and Data Access and Management Layers

# 7.3. Data Management Layer Design
## *Mapping domain classes to selected storage format*

- [Suppose that OODBMS is to be used]



# 7.4. Human–Computer Interaction Layer Design

- Interface design is the process of defining how the system will interact with external entities
- User-interface design includes three fundamental parts :
  - Interface structure and standards
  - I/O design: I/O accepted/generated by system
    - Input mechanism (e.g. windows containing dialogues, forms, etc)
    - Output mechanism(e.g. web pages, reports, etc)
  - Navigation design: WIMPS environment, touch screen, voice control, etc.

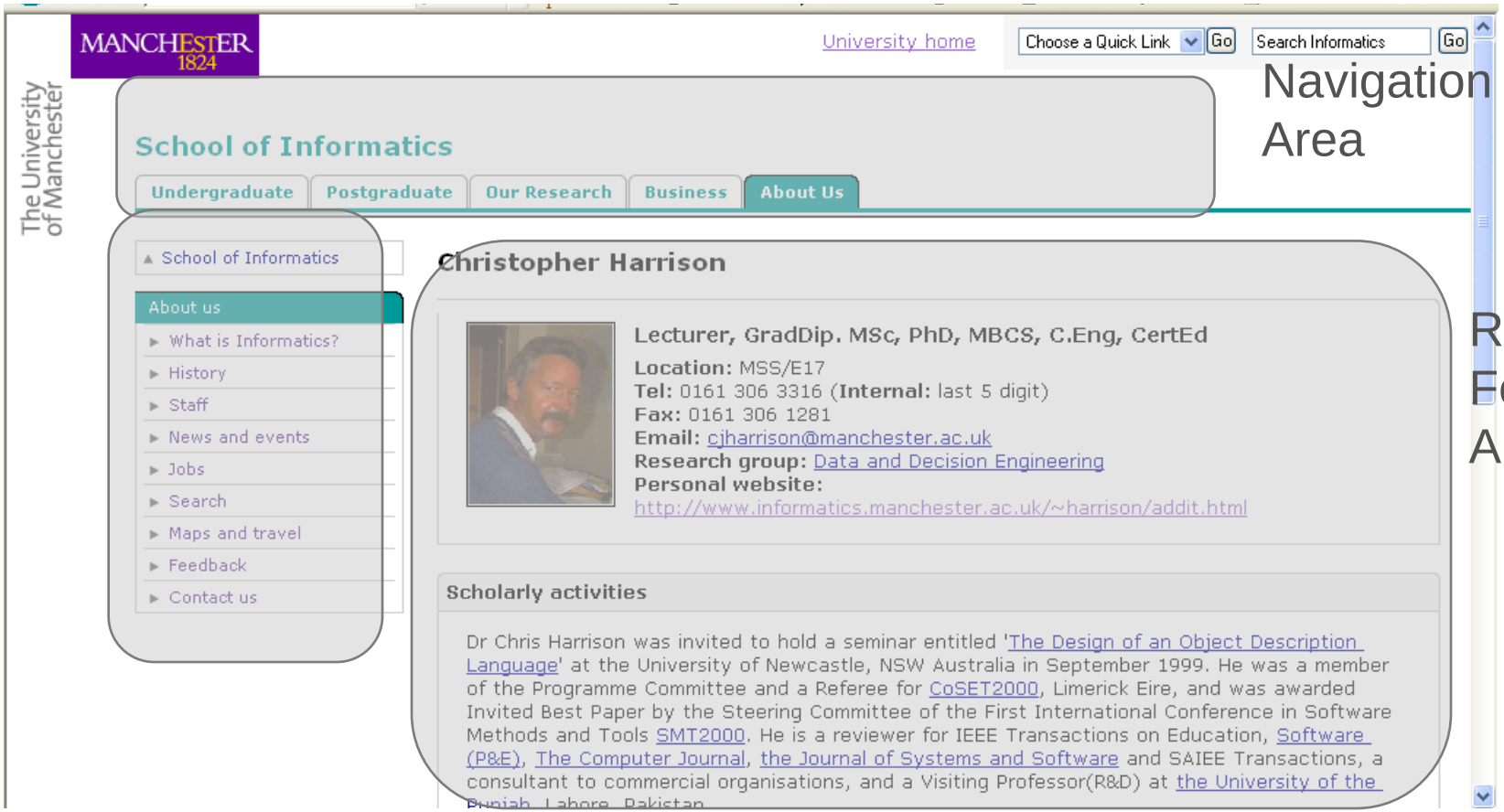
# 7.4. Human–Computer Interaction Layer Design

- Principles for User Interface Design

| Principle | Description |
|---|---|
| Layout | The interface should be a series of areas on the screen that are used consistently for different purposes—for example, a top area for commands and navigation, a middle area for information to be input or output, and a bottom area for status information. |
| Content awareness | Users should always be aware of where they are in the system and what information is being displayed. |
| Aesthetics | Interfaces should be functional and inviting to users through careful use of white space, colors, and fonts. There is often a trade-off between including enough white space to make the interface look pleasing and losing so much space that important information does not fit on the screen. |
| User experience | Although ease of use and ease of learning often lead to similar design decisions, there is sometimes a trade-off between the two. Novice users or infrequent users of software will prefer ease of learning, whereas frequent users will prefer ease of use. |
| Consistency | Consistency in interface design enables users to predict what will happen before they perform a function. It is one of the most important elements in ease of learning, ease of use, and aesthetics. |
| Minimize user effort | The interface should be simple to use. Most designers plan on having no more than three mouse clicks from the starting menu until users perform work. |

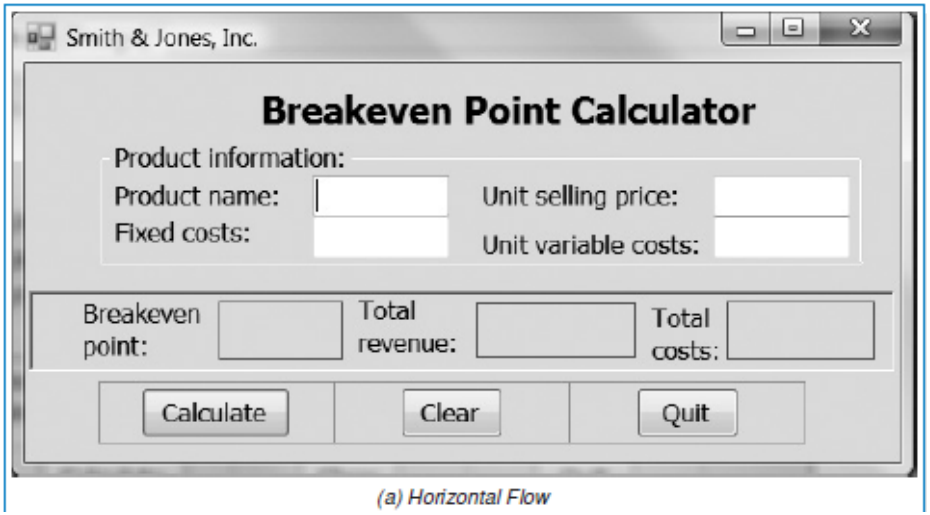
# Layout

- A series of areas on the screen: navigation area, user's work area (input information, output information, etc.), status information area

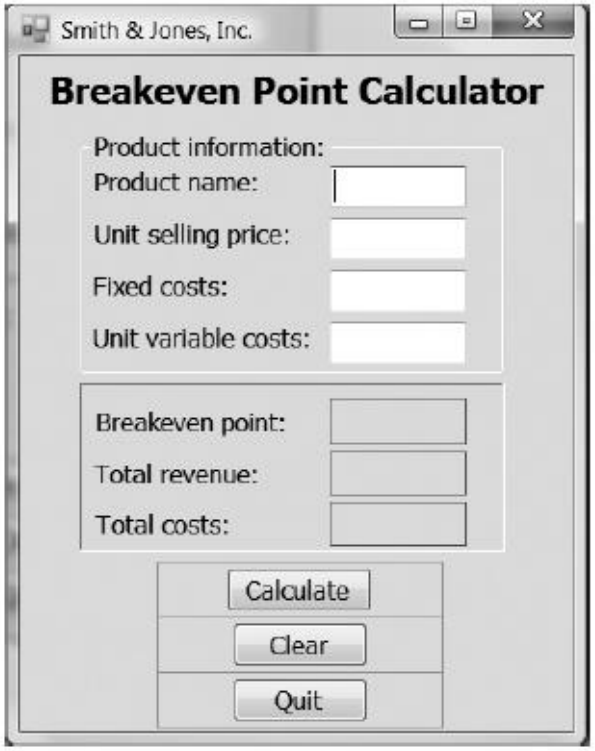


# Content Awareness

- User should always be aware of ..... with the least amount of effort.
  - where they are in the system ?
  - what am I supposed to be doing here?




*(a) Horizontal Flow*

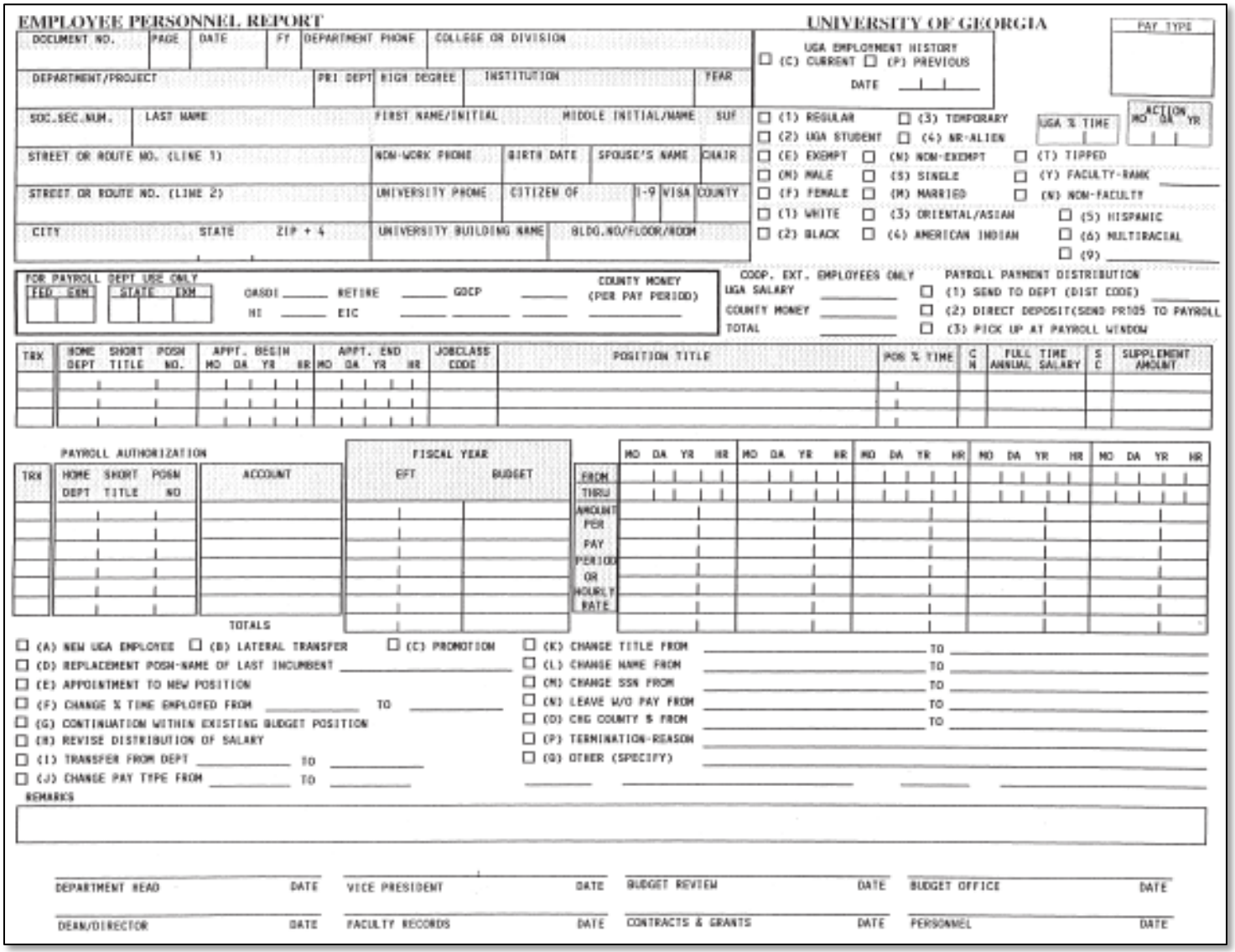



*(b) Vertical Flow*

# Aesthetics

- For pleasing to the eye
  - Color, size, text font, position etc.
- Interfaces should be functional, inviting to use, and pleasing to the eye
- In most cases, less is more (minimalist design)
- White space is important
- Acceptable information density is proportional to the user's expertise
  - Novice users prefer less than 50% density
  - Expert users prefer more than 50% density

# Bad Aesthetics



EMPLOYEE PERSONNEL REPORT

UNIVERSITY OF GEORGIA

DOCUMENT NO. | PAGE | DATE | FT | DEPARTMENT PHONE | COLLEGE OR DIVISION

DEPARTMENT/PROJECT | PRI DEPT | HIGH DEGREE | INSTITUTION | YEAR

SOC.SEC.NUM. | LAST NAME | FIRST NAME/INITIAL | MIDDLE INITIAL/NAME | SUF

STREET OR ROUTE NO. (LINE 1) | NON-WORK PHONE | BIRTH DATE | SPOUSE'S NAME | CHAIR

STREET OR ROUTE NO. (LINE 2) | UNIVERSITY PHONE | CITIZEN OF | I-9 | VISA | COUNTY

CITY | STATE | ZIP + 4 | UNIVERSITY BUILDING NAME | BLDG.NO/FLOOR/ROOM

UGA EMPLOYMENT HISTORY
☐ (C) CURRENT ☐ (P) PREVIOUS
DATE

PAY TYPE

ACTION MO DA YR

UGA % TIME

☐ (1) REGULAR ☐ (3) TEMPORARY
☐ (2) UGA STUDENT ☐ (4) NR-ALIEN
☐ (E) EXEMPT ☐ (N) NON-EXEMPT ☐ (T) TIPPED
☐ (M) MALE ☐ (S) SINGLE ☐ (Y) FACULTY-RANK
☐ (F) FEMALE ☐ (M) MARRIED ☐ (X) NON-FACULTY
☐ (1) WHITE ☐ (3) ORIENTAL/ASIAN ☐ (5) HISPANIC
☐ (2) BLACK ☐ (4) AMERICAN INDIAN ☐ (6) MULTIRACIAL
☐ (9)

FOR PAYROLL DEPT USE ONLY
FED EXM | STATE EXM
OASDI ____ RETIRE ____ GDCP ____ COUNTY MONEY (PER PAY PERIOD)
HI ____ EIC ____

COOP. EXT. EMPLOYEES ONLY
UGA SALARY
COUNTY MONEY
TOTAL

PAYROLL PAYMENT DISTRIBUTION
☐ (1) SEND TO DEPT (DIST CODE)
☐ (2) DIRECT DEPOSIT(SEND PR105 TO PAYROLL
☐ (3) PICK UP AT PAYROLL WINDOW

| TRX | HOME DEPT | SHORT TITLE | POSN NO. | APPT. BEGIN MO DA YR HR | APPT. END MO DA YR HR | JOBCLASS CODE | POSITION TITLE | POS % TIME | C N | FULL TIME ANNUAL SALARY | S C | SUPPLEMENT AMOUNT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |

PAYROLL AUTHORIZATION

| TRX | HOME DEPT | SHORT TITLE | POSN NO | ACCOUNT | FISCAL YEAR EFT | FISCAL YEAR BUDGET | | MO DA YR HR | MO DA YR HR | MO DA YR HR | MO DA YR HR | MO DA YR HR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | FROM | | | | | |
| | | | | | | | THRU | | | | | |
| | | | | | | | AMOUNT PER PAY PERIOD OR HOURLY RATE | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | TOTALS | | | | | | | | |

☐ (A) NEW UGA EMPLOYEE ☐ (B) LATERAL TRANSFER ☐ (C) PROMOTION
☐ (D) REPLACEMENT POSN-NAME OF LAST INCUMBENT ____
☐ (E) APPOINTMENT TO NEW POSITION
☐ (F) CHANGE % TIME EMPLOYED FROM ____ TO ____
☐ (G) CONTINUATION WITHIN EXISTING BUDGET POSITION
☐ (H) REVISE DISTRIBUTION OF SALARY
☐ (I) TRANSFER FROM DEPT ____ TO ____
☐ (J) CHANGE PAY TYPE FROM ____ TO ____

☐ (K) CHANGE TITLE FROM ____ TO ____
☐ (L) CHANGE NAME FROM ____ TO ____
☐ (M) CHANGE SSN FROM ____ TO ____
☐ (N) LEAVE W/O PAY FROM ____ TO ____
☐ (O) CHG COUNTY $ FROM ____ TO ____
☐ (P) TERMINATION-REASON ____
☐ (Q) OTHER (SPECIFY) ____

REMARKS

DEPARTMENT HEAD DATE | VICE PRESIDENT DATE | BUDGET REVIEW DATE | BUDGET OFFICE DATE

DEAN/DIRECTOR DATE | FACULTY RECORDS DATE | CONTRACTS & GRANTS DATE | PERSONNEL DATE

# User Experience

- Designing the user interface with the users' level of computer experience in mind
- Ease of learning
  - Significant issue for inexperienced users
  - Relevant to systems with a large user population
- Ease of use
  - Significant issue for expert users
  - Most important in specialized systems
- Sometimes ease of learning and ease of use go hand in hand

# Consistency

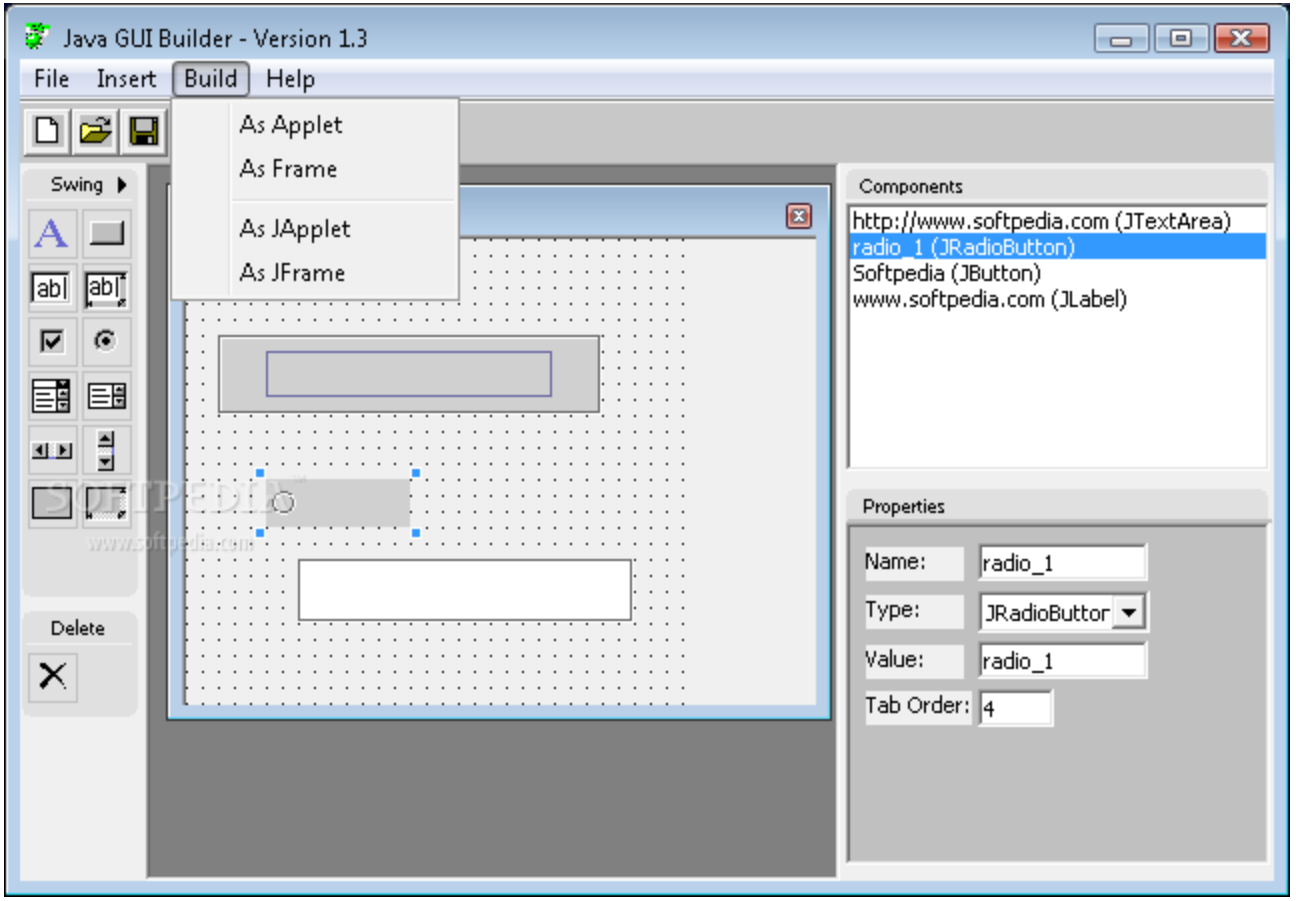
- All parts of the system work in the same way
- Key areas of consistency are
  - Navigation controls (menus, icons)
  - Terminology (customer/client)
- Probably most important concept in making the system simple because it allows the users to *predict what is going to happen*

# Minimal User Effort

- Interfaces should be designed to minimize the effort needed to accomplish tasks
- Using the fewest possible mouse clicks or keystrokes to move from one part of the system to another
- A common rule is the tree-clicks rule
  - Users should be able to go from main menu of a system to the information they want in no more than three mouse clicks

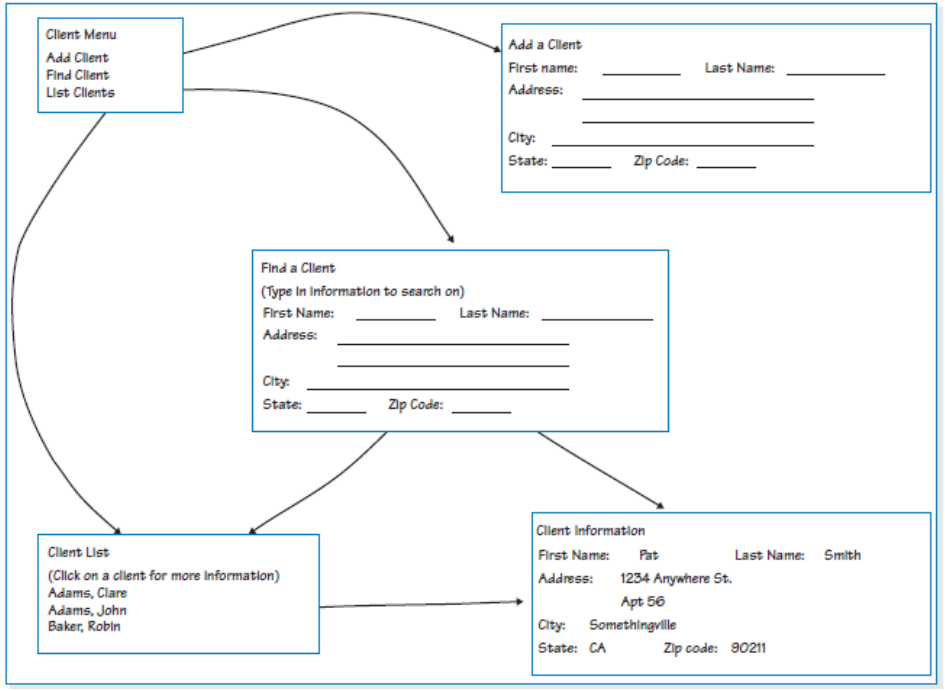
# Graphical user interface builder

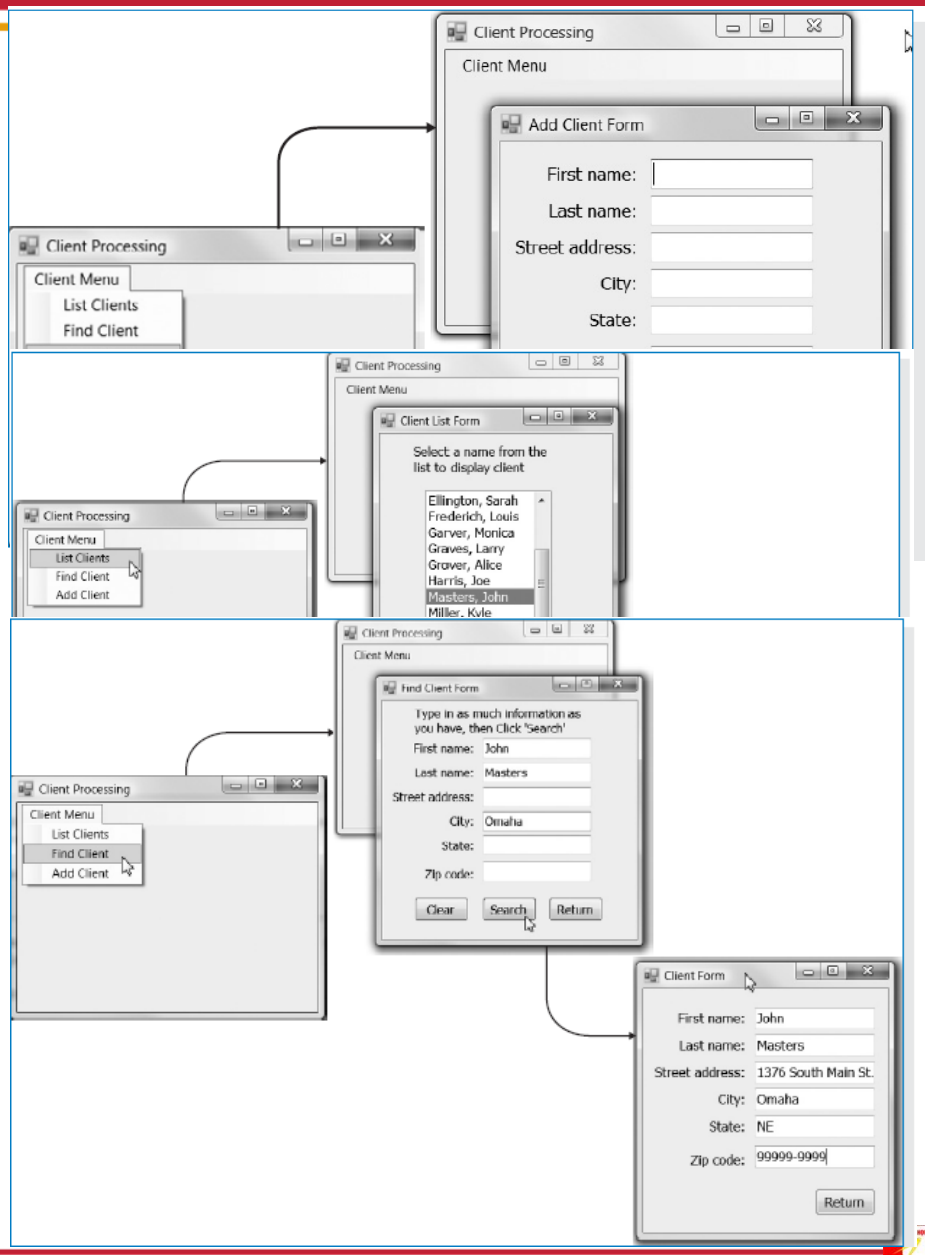
- allow the designer to arrange graphical control elements (often called widgets) using a drag-and-drop WYSIWYG editor



# 7.4. Human–Computer Interaction Layer Design

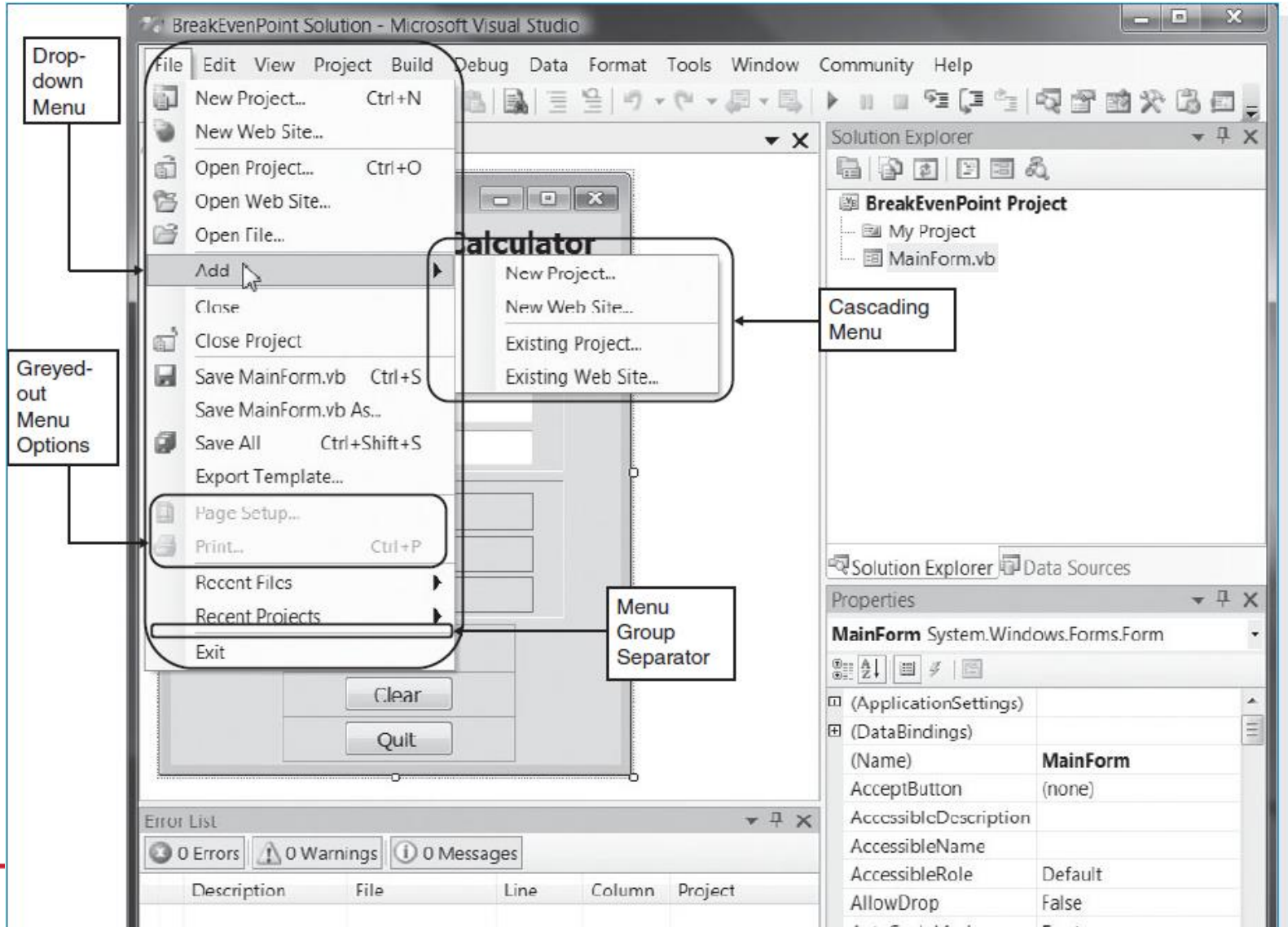
- Interface Design Prototyping
  - The storyboard



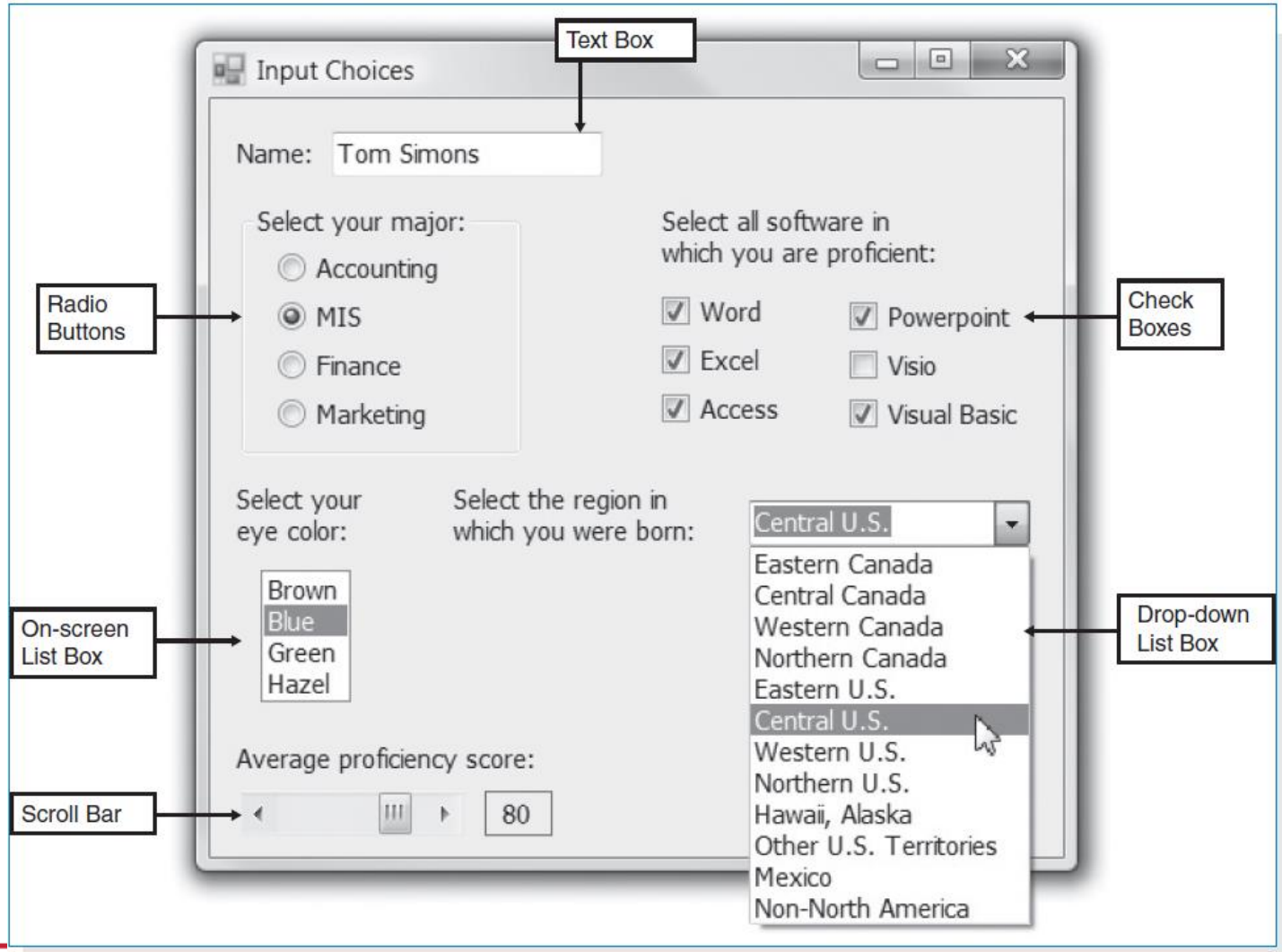
# 7.4. Human–Computer Interaction Layer Design

- Navigation Design



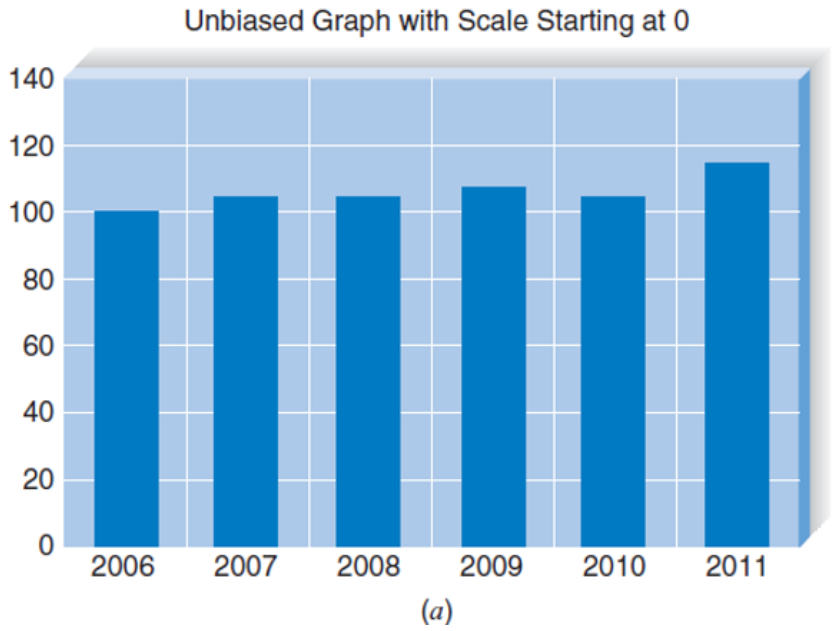
# 7.4. Human–Computer Interaction Layer Design

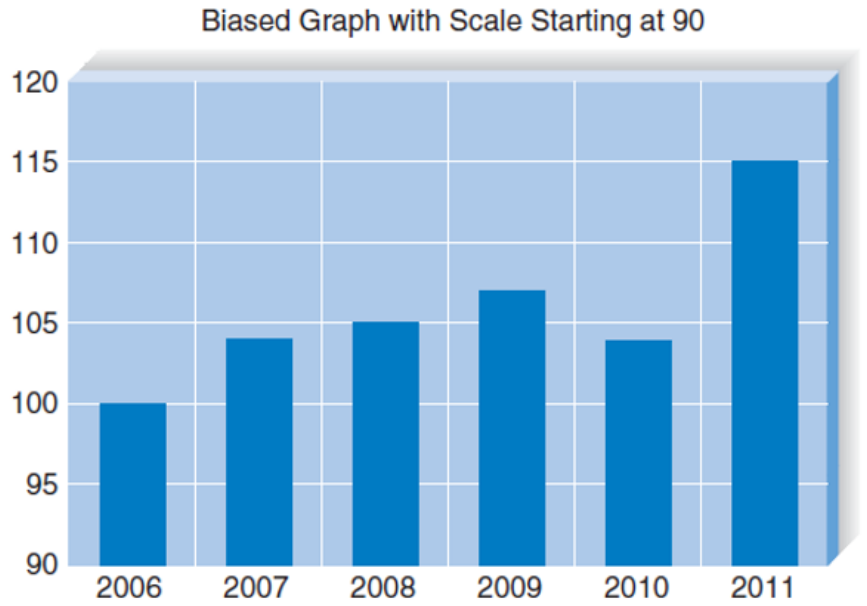
- Input design



# 7.4. Human–Computer Interaction Layer Design

- Output Design
  - Outputs are the reports that the system produces, whether on the screen, on paper, or in other media, such as the Web.
  - Understand report usage,
  - Manage information load,
  - Minimize Bias



# 7.5. Physical Architecture

- Most modern IT systems involve co-operating computers
  - Web-based systems execute in a client's browser and interact with a Web server, and possibly kinds of other server, via the Internet
  - An IT system operating entirely within a company's network may have applications on a given computer that interact with other servers, e.g. a database server, within the same network
- Most systems exploit existing hardware and software infrastructure

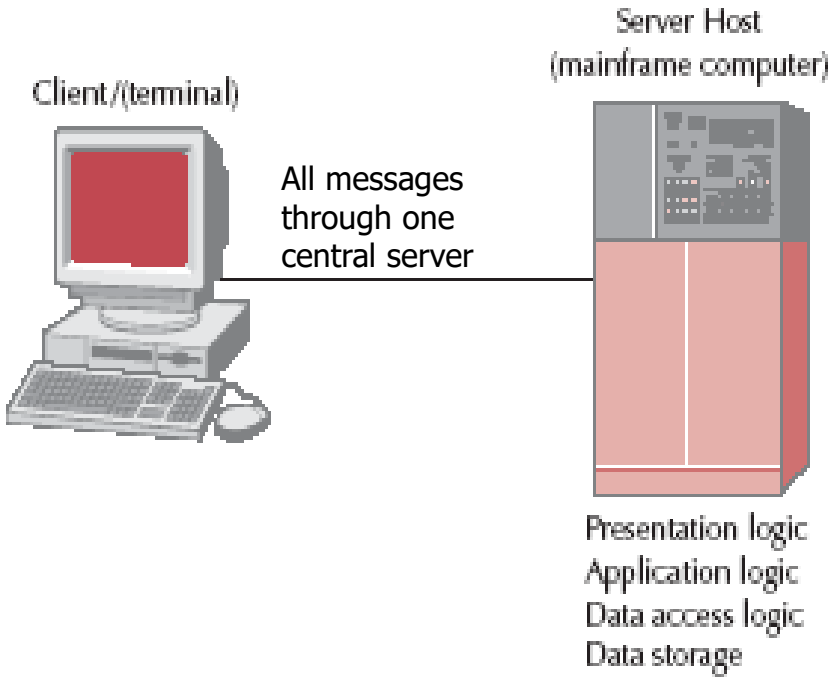
- The Physical Architecture Layer design models the system's hardware, software and network environments
  - [1] how system is distributed amongst computers
  - [2] the hardware of those computers
  - [3] the software on those computers

# 7.5. Physical Architecture

- Elements of Physical Architecture Layer
  - Software system has four *functions*
    - [1] Data storage
      - Modelled by Data Management Layer as the persistent objects
      - Data is documented in structural model (CRC cards and Class Diagrams)
    - [2] Data Access Logic
      - As modelled by Data Access and Management Classes in Data Management Layer
      - Describes processing required to access stored data, e.g. SQL queries
    - [3] Application logic
      - As modelled in Domain Layer's functional and behavioural models
      - Functional Model = Activity Diagrams and Use Cases
      - Behavioural Models = Sequence and Communication diagrams, Behavioural State Machines
    - [4] Presentation Logic
      - Modelled in HCI layer
  - Hardware components are
    - [1] Client computers
      - I/O devices or system's users, e.g. desktop/laptop computers, handheld, mobile, etc
    - [2] Servers
      - Mainframes, minicomputers and high-end microcomputers
    - [3] Connecting Network
      - High-speed permanent Ethernet/T3/ATM, broadband, dialled

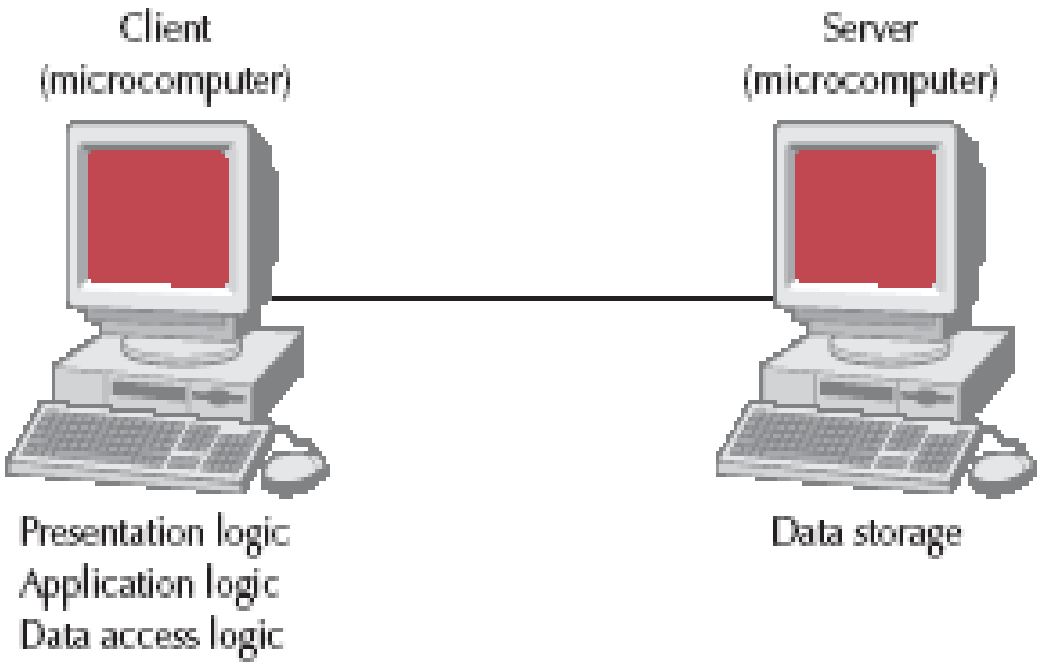
# 7.5. Physical Architecture

- Various architectures
  - Server-Based Architectures
    - Central (mainframe) computer performed all four application functions
    - Clients (dumb terminals) enables messages (keystrokes) to be sent to server and received messages from server (instructions defining what to display)
    - More machines and/or more applications slows server response times



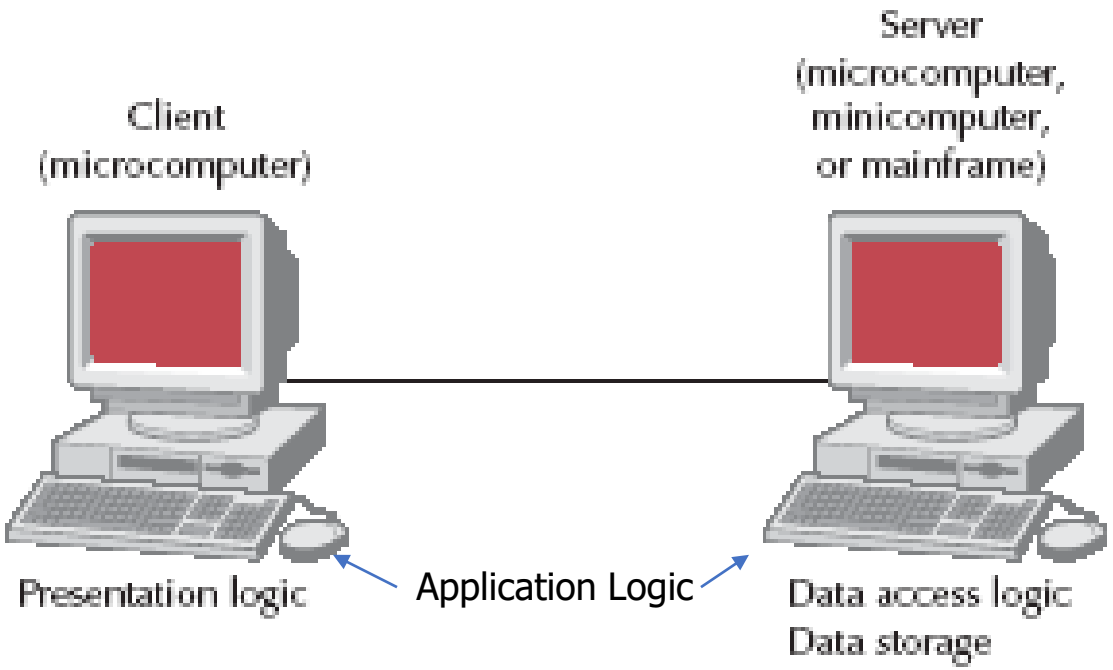
# 7.5. Physical Architecture

- Various architectures
  - Client-Based Architectures
    - Client = PC, server is (high-end) PC on same Local Area Network (LAN)
    - Client has its own processing capability
    - All data must travel from server to client for processing



# 7.5. Physical Architecture

- Various architectures
  - Client-Server Architectures
    - Client has Presentation Logic
    - Server has Data Access Logic and Data Storage
    - Application Logic
      - [1] On Client
      - [2] On Server
      - [3] Shared between Client and Server



# 7.5. Physical Architecture

- Advantages of Client-Server
  - [1] Scalability
    - Storage capacity and processing power of servers can be increased (or decreased) easily
      - If one (or more) servers performance degrades, add one (or more) servers to perform the Application Logic, Data Access Logic or Data Storage
  - [2] Support many kinds of clients and servers
    - Different computers can run different operating systems
    - Support middleware on clients and servers
  - [3] using Internet standards enable separation of Presentation Logic, Application Logic and Data Access Logic
    - Design of each can be independent
    - Can change Application Logic without changing Presentation Logic or Data
  - [4] No central point of failure can halt the entire network
    - If a server fails, other servers still available
- Disadvantage
  - Complexity and expensive due to the separation of software between Client and Server

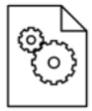
- Selecting a Physical Architecture

| | Server-Based | Client-Based | Client–Server |
|---|---|---|---|
| Cost of infrastructure | Very high | Medium | Low |
| Cost of development | Medium | Low | High |
| Ease of development | Low | High | Low–medium |
| Interface capabilities | Low | High | High |
| Control and security | High | Low | Medium |
| Scalability | Low | Medium | High |

# 7.5. Physical Architecture

- Deployment Diagram
  - Show relationships between hardware components in physical infrastructure (of information system)
  - Also used to represent software components and their deployment over physical architecture/infrastructure
  - Nodes: Hardware components
    - Client computers, servers, separate networks, individual (networked) device
    - Labelled with its name (possibly a stereotype <<text>>>)
    - Stereotype = type of node, E.g. device, mobile device, database server, web server, application server
    - Extended syntax = network node symbols
  - Artifacts: Elements of information system deployed onto physical architecture
    - E.g. Software component, subsystem (module), database table, (entire) dat layer
    - Labelled with its name (possibly a stereotype <<text>>>)
    - Stereotype = type of node, E.g. source file, database table, executable file
  - Communication paths: Links between nodes of the network
    - Stereotype = type of link OR protocol, E.g. LAN, Internet, serial, parallel, USB OR TCP/IP

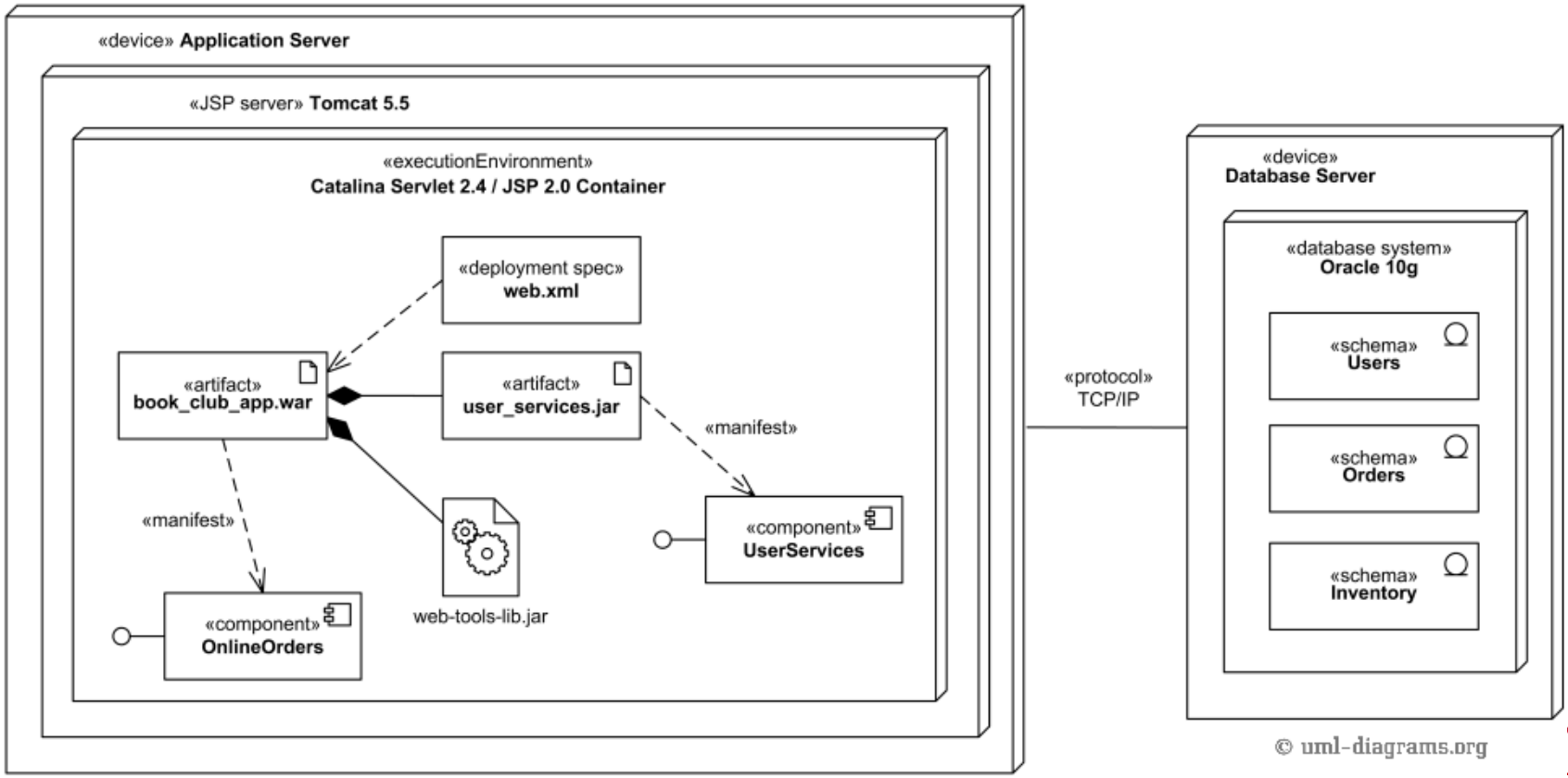




web-tools-lib.jar

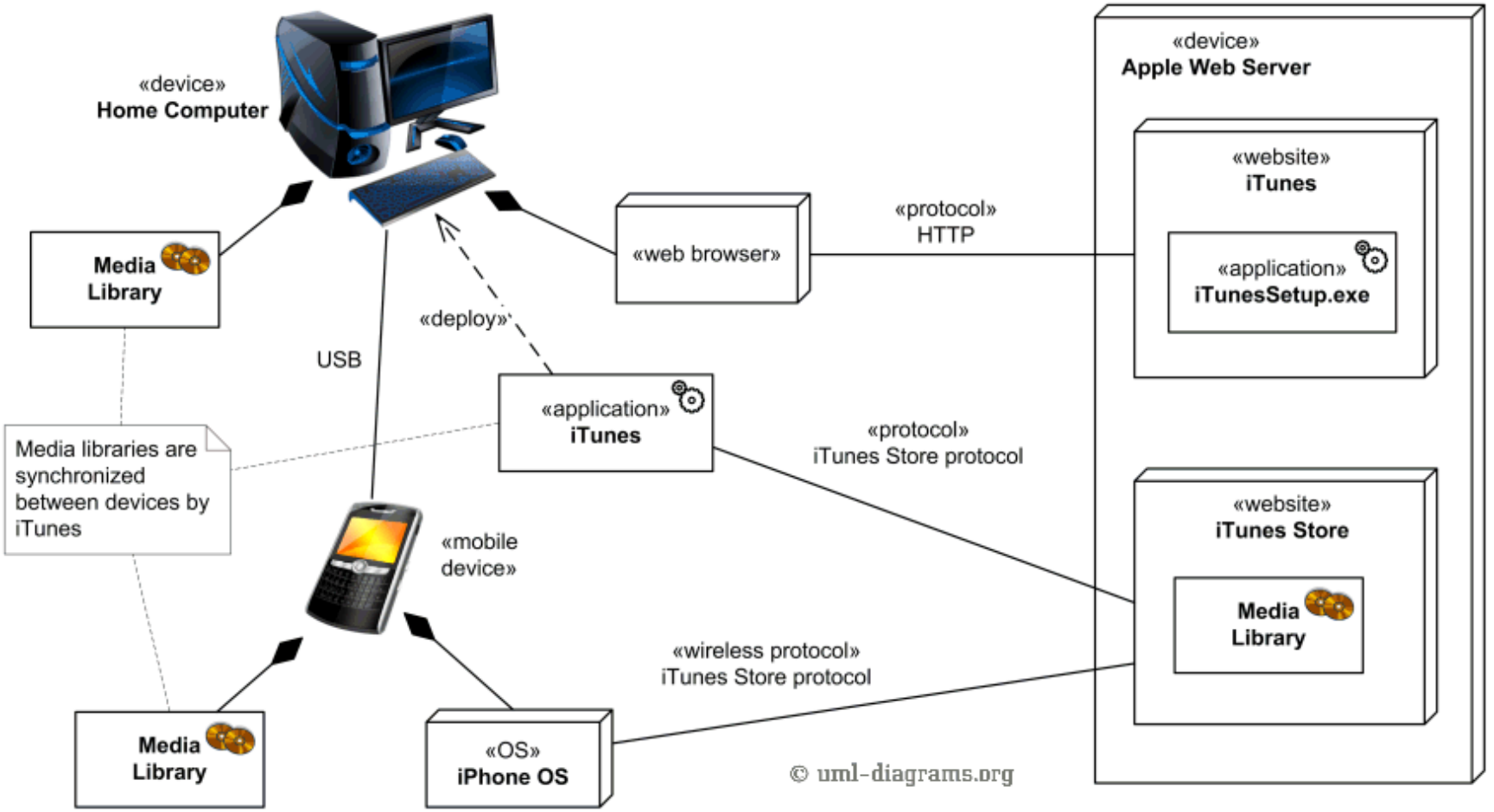
# 7.5. Physical Architecture

- Deployment Diagram
  - An example of deployment diagram for J2EE web application
  - A web application artifact book_club_app.war is deployed on Catalina Servlet 2.4 / JSP 2.0 Container which is part of Apache Tomcat 5.5 web server.
  - The Application Server «device» (computer server) has communication path to Database Server «device» (another server).



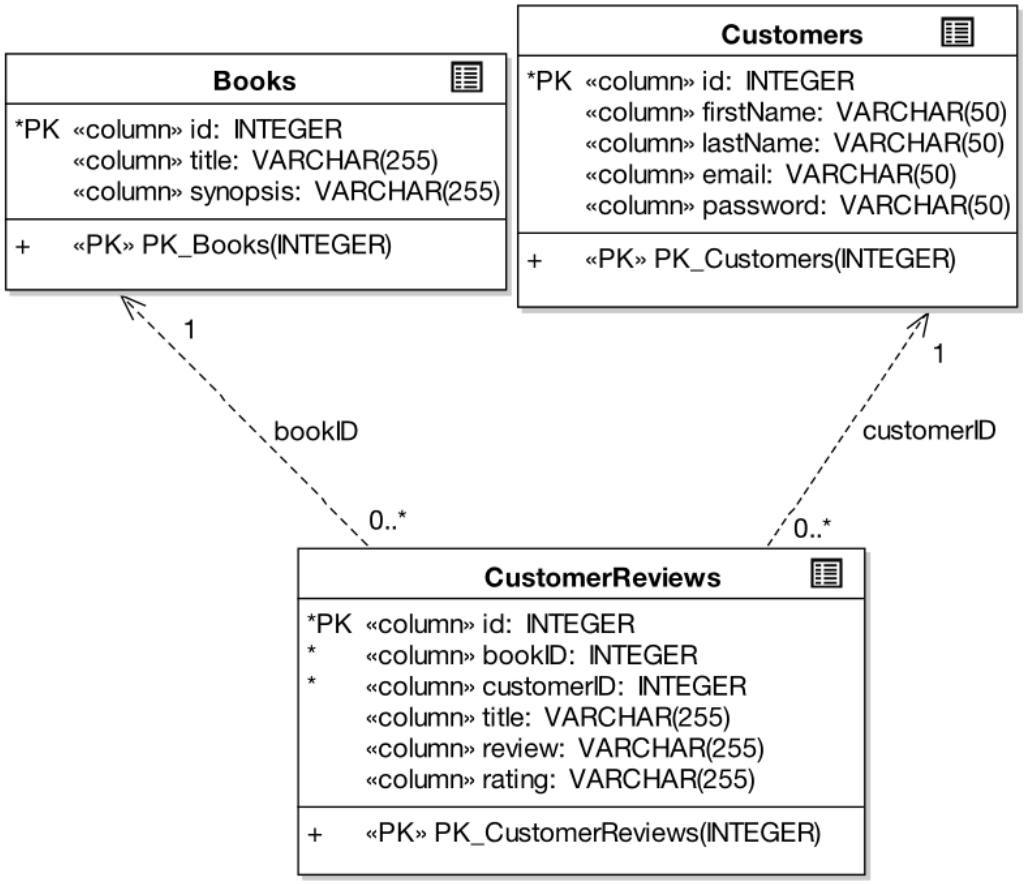
# 7.5. Physical Architecture

- Deployment Diagram
  - An example for Apple iTunes application



# CASE STUDY: Internet Bookstore

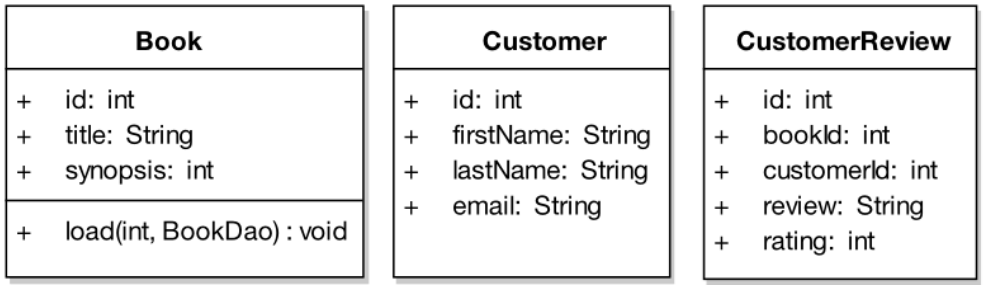
- Creating the Database




...

# CASE STUDY: Internet Bookstore

- Mapping Domain (Entity) Classes to Real Classes




```
package com.iconixsw.bookstore.domain;

import java.io.*;

public class Book implements Serializable {

    private int id;
    private String title;
    private String synopsis;

    public Book() {
    }

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getSynopsis() { return synopsis; }
    public void setSynopsis(String synopsis) {
        this.synopsis = synopsis;
    }

    public void load(int id, BookDao bookDao) {
    }
}
```

# CASE STUDY: Internet Bookstore

- Mapping Boundary Classes to Real Classes

to return the main (“home”) view page

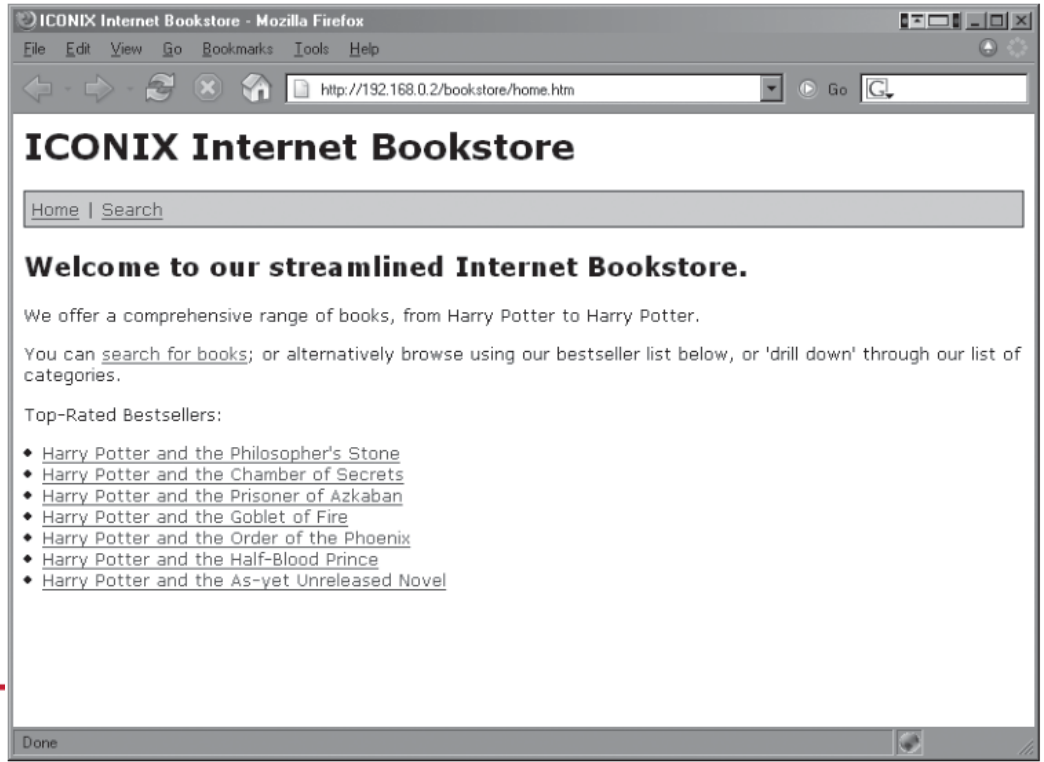
```
package com.iconixsw.bookstore.web;

// import statements omitted . . .

public class HomeController implements Controller {

    public ModelAndView handleRequest(
                        HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException
    {
        return new ModelAndView("home");
    }

}
```

Spring’s DispatcherServlet forwards the request to *home.jsp*, which will then be displayed to the user.

# CASE STUDY: Internet Bookstore

- The Home Page: home.jsp

```
<%@ include file="include/IncludeTop.jsp" %>

<h2>Welcome to our streamlined Internet Bookstore.</h2>

<p>
  We offer a comprehensive range of books, from Harry Potter to Harry Potter.
</p>
<p>
  You can <a href="search.jsp">search for books</a>; or alternatively browse
  using our bestseller list below, or 'drill down' through our
  list of categories.
</p>

<lu>
  <li>
    <a href="bookdetails.htm?id=101">
      Harry Potter and the Philosopher's Stone
    </a>
  </li>
  <li>
    <a href="bookdetails.htm?id=102">
      Harry Potter and the Chamber of Secrets
    </a>
  </li>

<!-- etc . . . -->
</lu>

</body>
</html>
```



# CASE STUDY: Internet Bookstore

- The Write Customer Review page