HNDIT 2312- Principles of Software Engineering

Week 6: Software Design-ii(DFD/ER/UI)

Topics Covered

- Data Flow Diagram(DFD)
  - Context diagram
  - Level 1 /2 DFD
- Database design
  - ER diagram
  - Relational modeling
- UI design guideline

Steps in Software Design

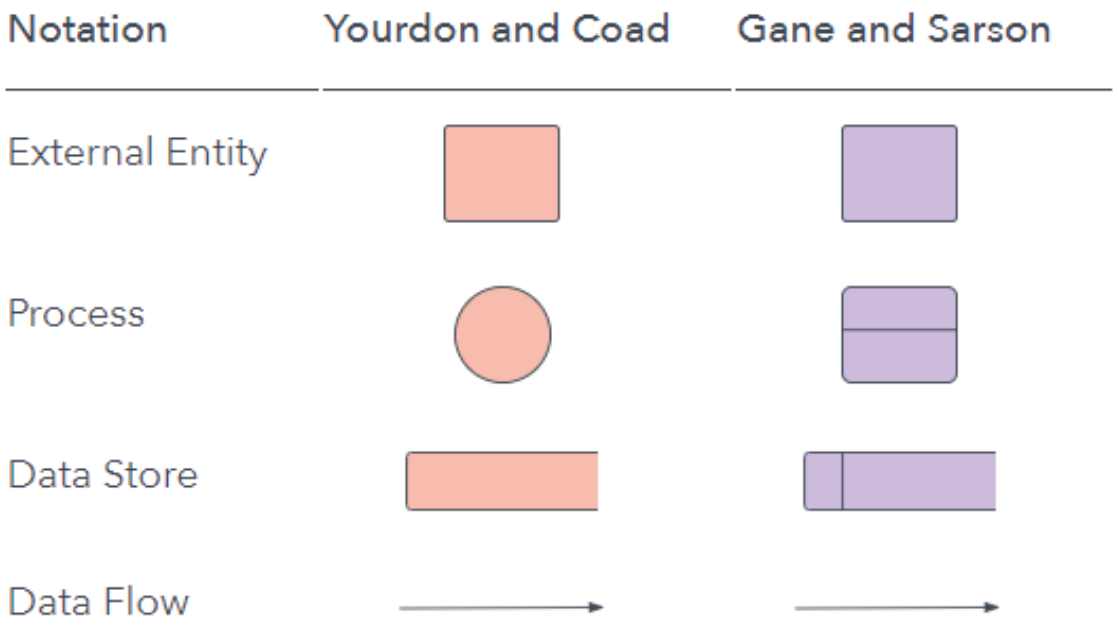
- Functional requirement(SRS)
- Use case diagram
- Data flow diagram
- Database design
- Interface design

Data flow diagram

- A data flow diagram (DFD) maps out the flow of information for any process or system.
- Data flow diagrams were popularized in the late 1970s, arising from the book *Structured Design*, by computing pioneers Ed Yourdon and Larry Constantine.

Symbols and Notations Used in DFDs

- There are two models, Yourdon-Coad and Yourdon-DeMarco.



External entity

- An outside system that sends or receives data, communicating with the system being diagrammed.
- They might be an outside organization or person, a computer system or a business system.

Process

- Any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.
- A short label is used to describe the process, such as “Submit payment.”

Data store

- Files or repositories that hold information for later use, such as a database table or a membership form.
- Each data store receives a simple label, such as “Orders.”

Data flow

- The route that data takes between the external entities, processes and data stores.
- Typically labeled with a short data name, like “Billing details.”

DFD rules and tips

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

DFD levels and layers

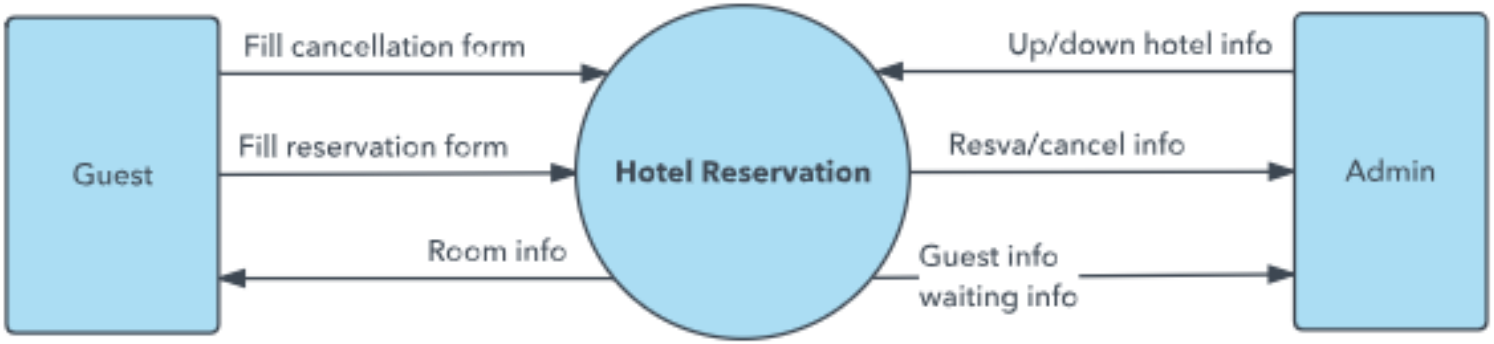
- A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece.
- DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond.

DFD Context diagram(Level 0)

- An abstract view of the system is represented using a context diagram.
- The entire system is shown as a single process, labeled with the name of the system.

- **Context Diagram Can Contain:**
  - One process(represents the entire system)
  - All External entities (data sources/sinks)
  - External data flows from/to external entities (inputs / outputs)

DFD Level 0(Context diagram)



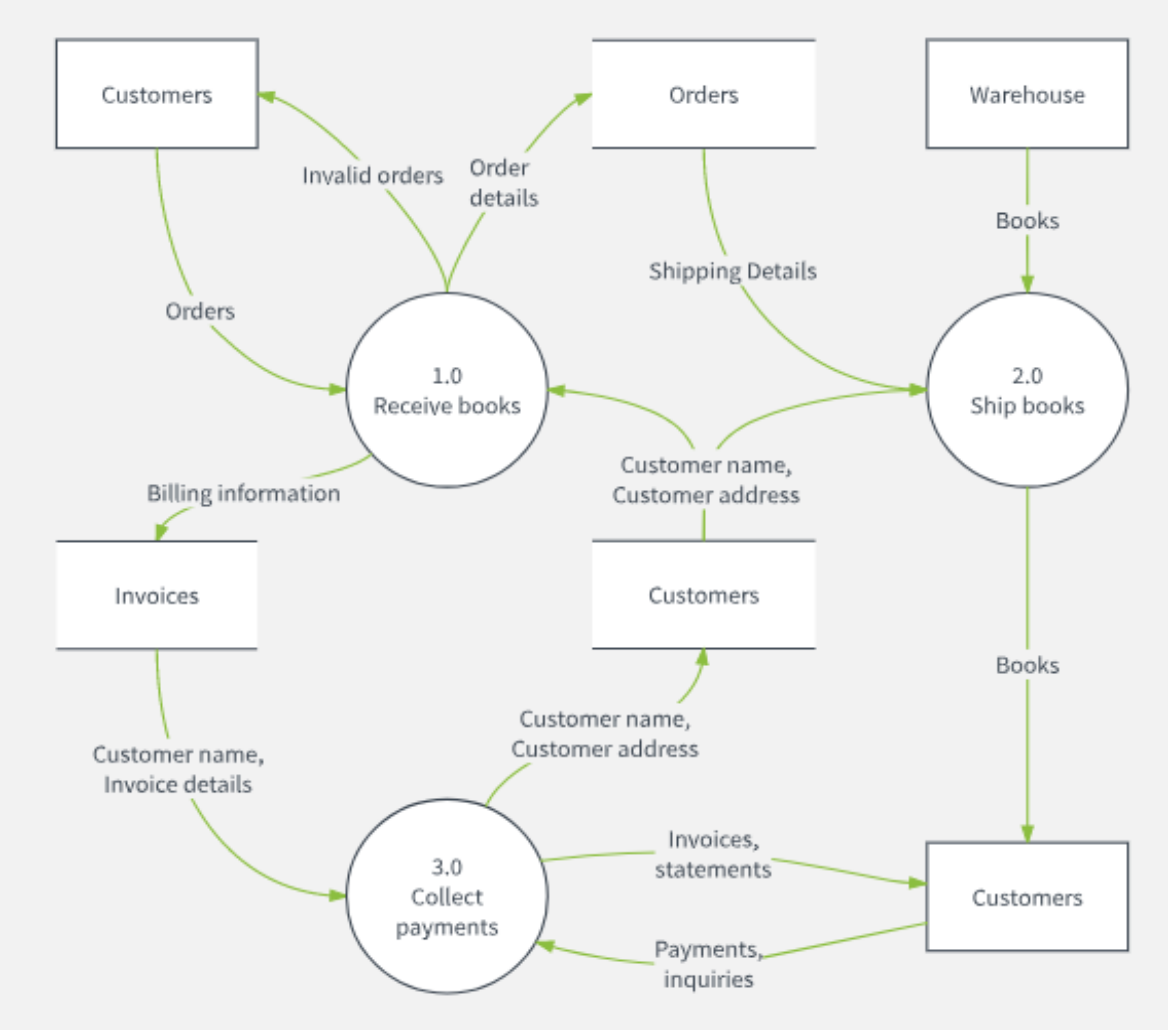
DFD Level 1

- DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram.
- You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

Drawing Level 1 DFD diagram

- Identify major functions
- Draw a Bubble for each process
  - Number processes(full numbers)
  - Too many processes i.e.> 7make the DFD difficult to understand
- Identify the major data stores
  - Link them to the relevant processes
- Balance the data flows with Context Diagram

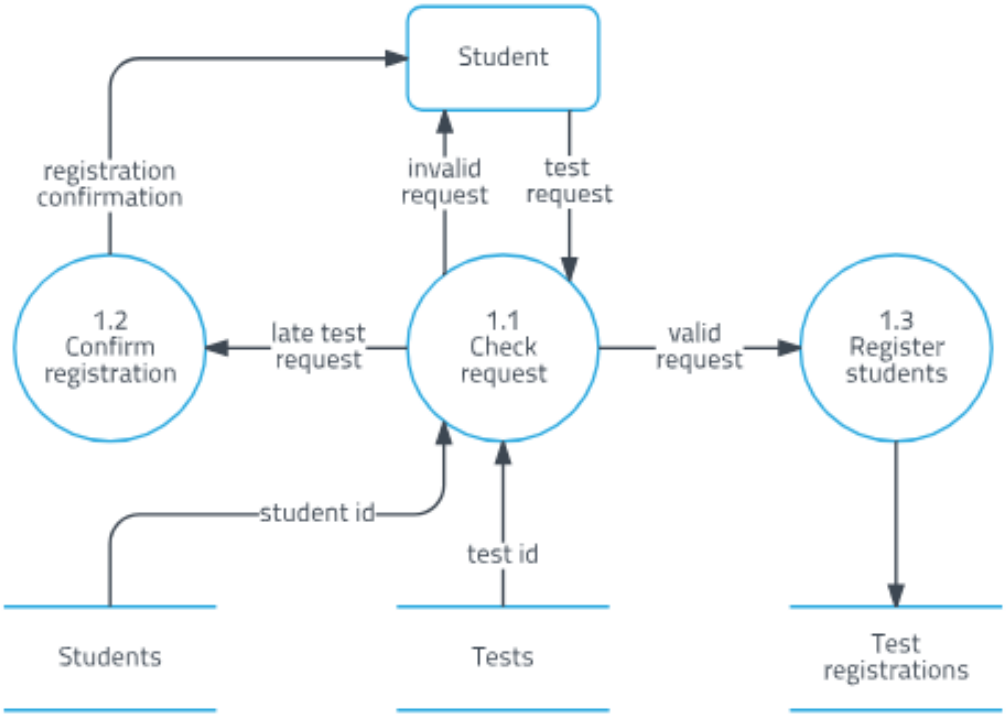
Level 1 DFD


Customers
Orders
Warehouse
Invalid orders
Order details
Books
Shipping Details
Orders
1.0 Receive books
2.0 Ship books
Billing information
Customer name, Customer address
Invoices
Customers
Books
Customer name, Invoice details
Customer name, Customer address
3.0 Collect payments
Invoices, statements
Customers
Payments, inquiries

Level 2 DFD

- Identify processes that needs Decomposition(factoring/exploding)
- When a process on the level-1 is Decomposed, it will be represented by a new Sub-Diagram (Level2) with more details.
- Give decimal numbers to the processes in the sub diagrams
- Balance the data flows (Level Balancing)

Level 2 DFD Decomposition

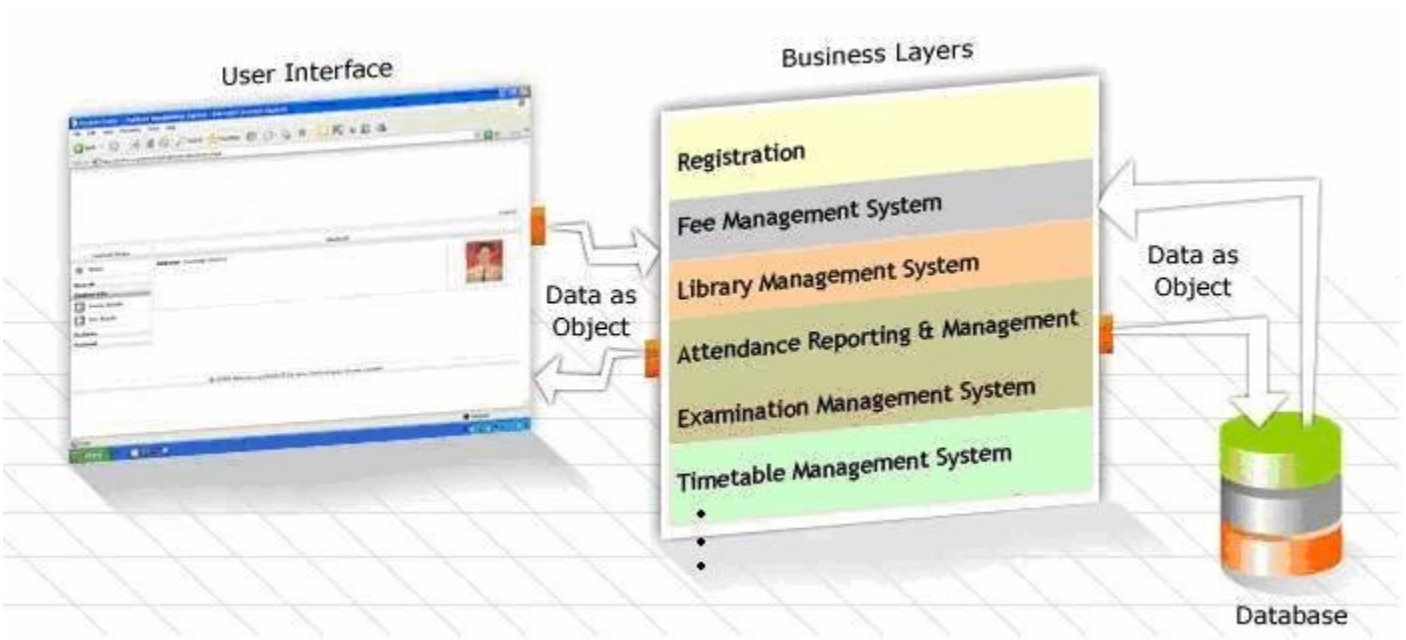


DFD vs. UML

- While a DFD illustrates how data flows through a system, UML is a modeling language used in Object Oriented Software Design to provide a more detailed view.
- A DFD may still provide a good starting point, but when actually developing the system, developers may turn to UML diagrams such as class diagrams

Why Database?

- Database represent the data layers to system.



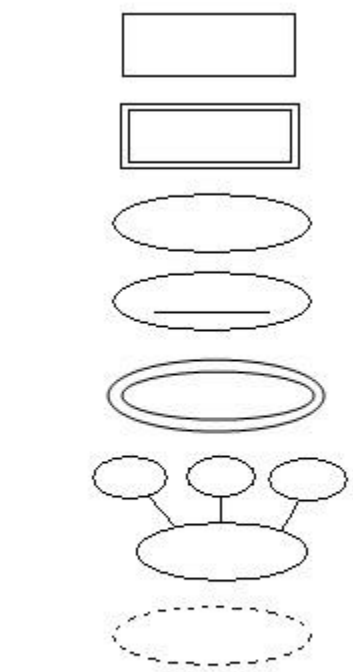
Database Design

- Requirement analysis
- Conceptual database design(ER diagram)
- Schema refinement(Relational model)
- Logical database design(Normalization)
- Physical database design(SQL)
- Security design (transactions)

Conceptual Database design(ER diagram)

- ER diagrams are used to model and design relational databases, in terms of logic and business rules .
- ER Diagrams are composed of entities, relationships and attributes.

ER diagram symbols



Entity or Entity Type

Weak Entity Type

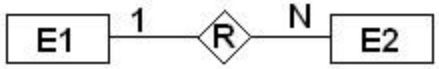
Attribute

Key Attribute

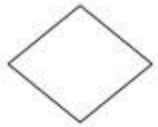
Multivalued Attribute

Composite Attribute

Derived Attribute

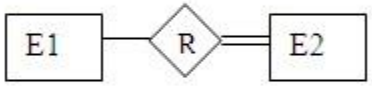


Cardinality Ratio 1:N for E1:E2 in R

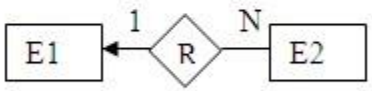


Relationship or Relationship Type

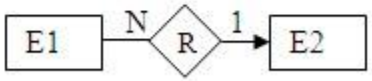
Identify Relationship Type



Total Participation



Cardinality Ratio between E1 and E2 in 1:R

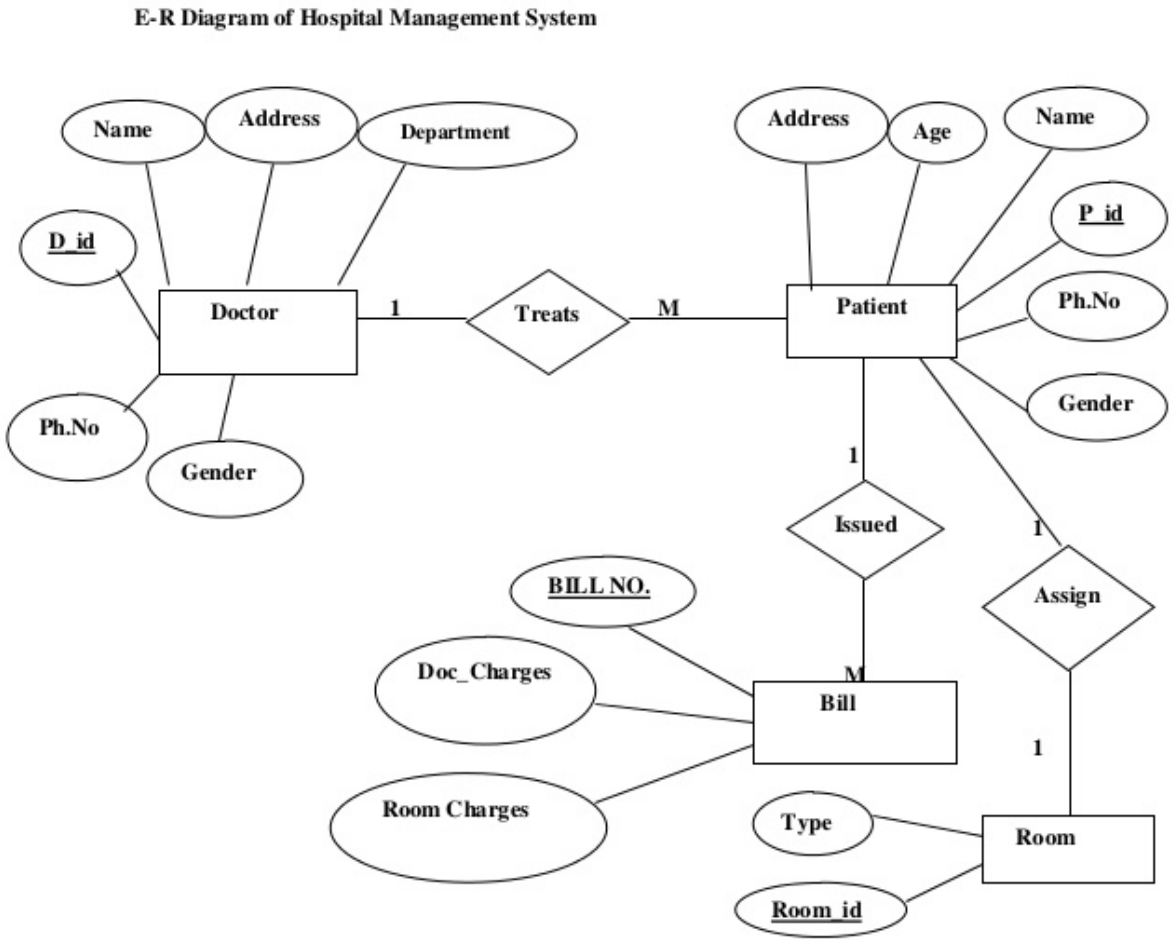


Many to One Relationship Type

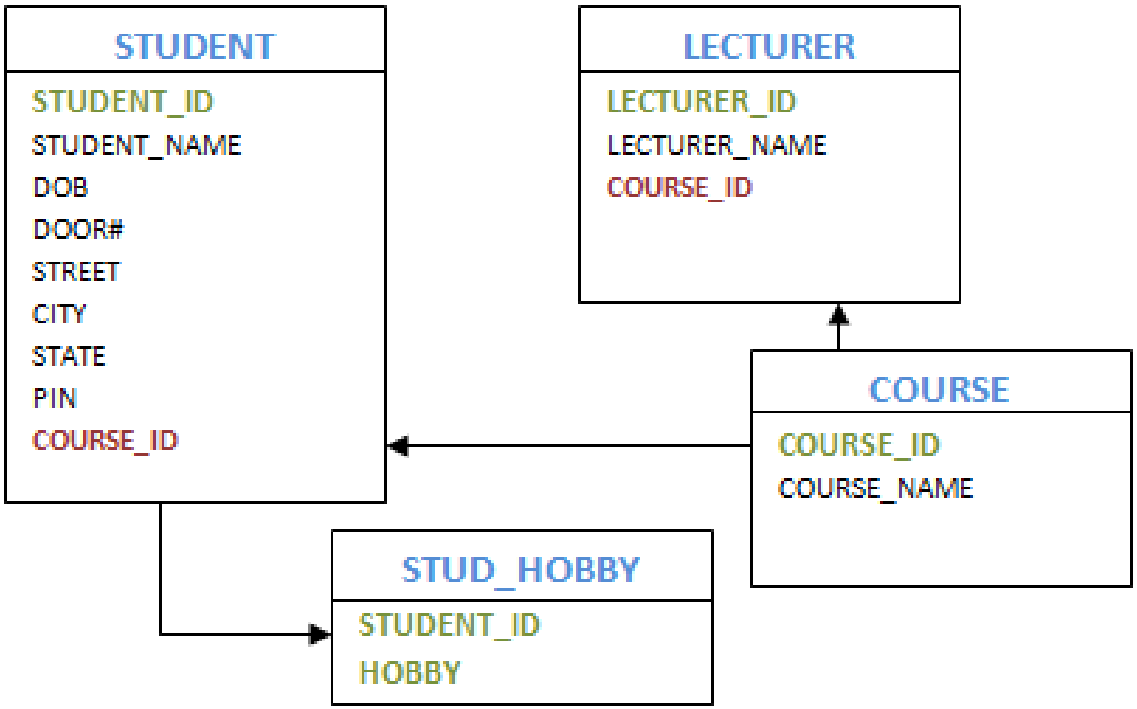
How to draw a basic ER diagram

- Identify the entities that are involved.
- Identify the relationship among entities.
- Identify the attributes of each entities including key attributes.
- Show the cardinality ratio.

ER diagram example



ERD to Relational model



User Interface Design

- User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

Choosing Interface Elements

- **Input Controls**: buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field
- **Navigational Components**: breadcrumb, slider, search field, pagination, slider, tags, icons
- **Informational Components**: tooltips, icons, progress bar, notifications, message boxes, modal windows
- **Containers**: accordion

Best Practices for Designing an Interface

- **Keep the interface simple.** The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.
- **Create consistency and use common UI elements.** By using common elements in your UI, users feel more comfortable and are able to get things done more quickly.

Best Practices for Designing an Interface

- **Strategically use color and texture.** You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage.
- **Use typography to create hierarchy and clarity.** Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scanability, legibility and readability.

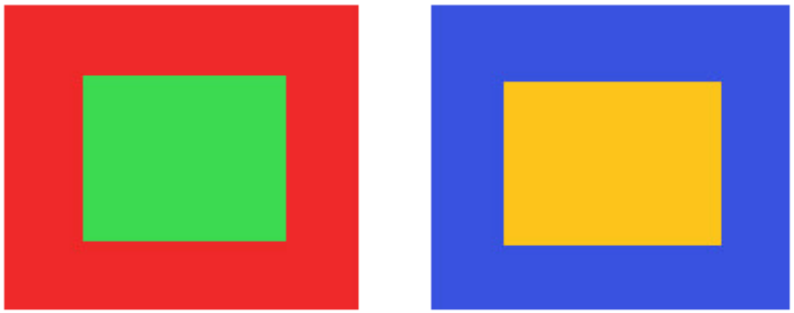
Using Light, Color and Contrast Effectively in UI Design

- Light - create depth using light and shadow.

- Think of a button. When a button is pressed, most designers tend to make it darker. Why? The reason is simple – we imagine the button as a 3D object.

Using Light, Color and Contrast Effectively in UI Design

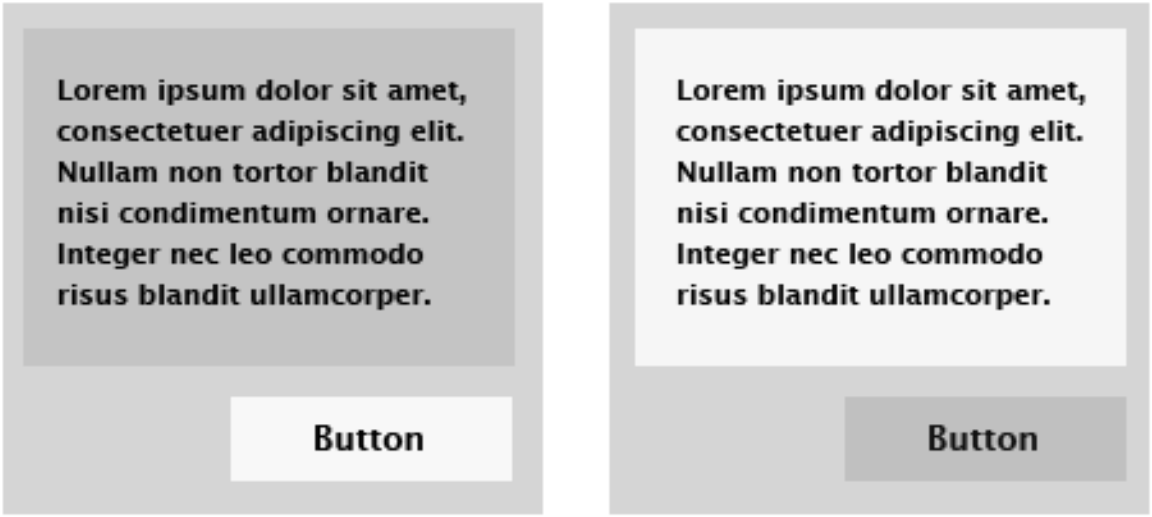
- Color - use heat to manage focus.



- Yellow is warm and warmer colors are brighter. Not only that, they also appear closer to us.

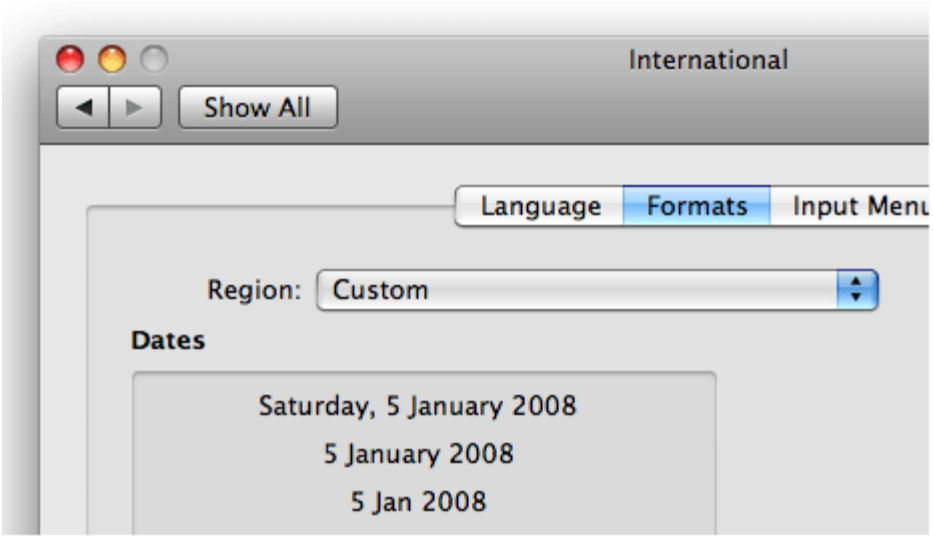
Using Light, Color and Contrast Effectively in UI Design

- Contrast - attract attention with higher contrast. Higher contrast items stand out – they catch your eye.



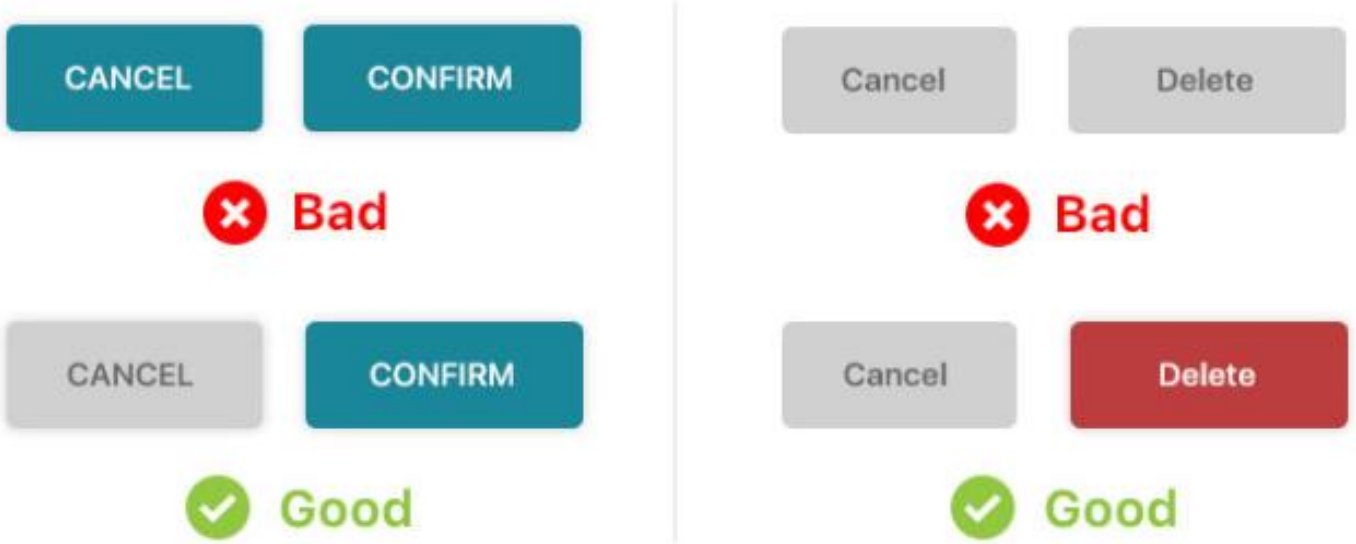
Example: The OS X interface

- Let's see a real world example of these principles from a well known usability expert – Apple.



Examples: Good vs. Bad UI design

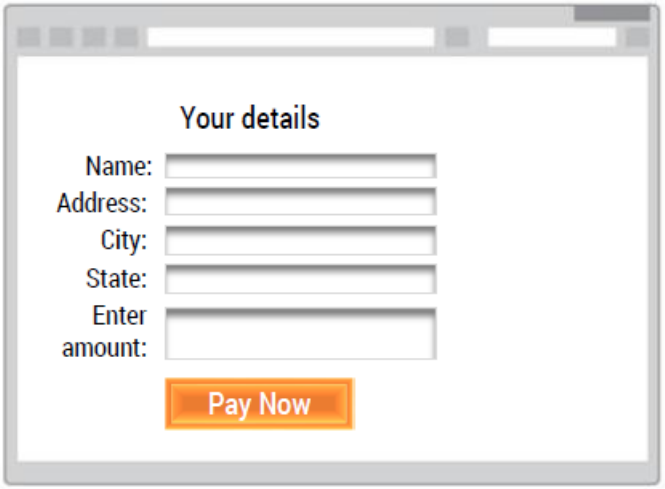
- Priority



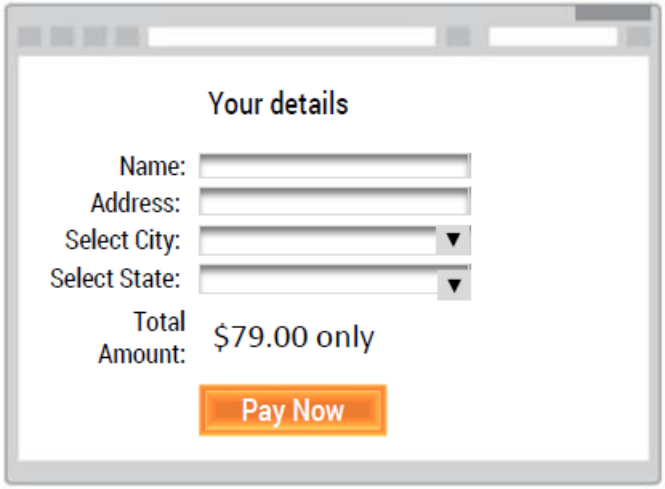
Examples: Good vs. Bad UI design

- User effort

More efforts

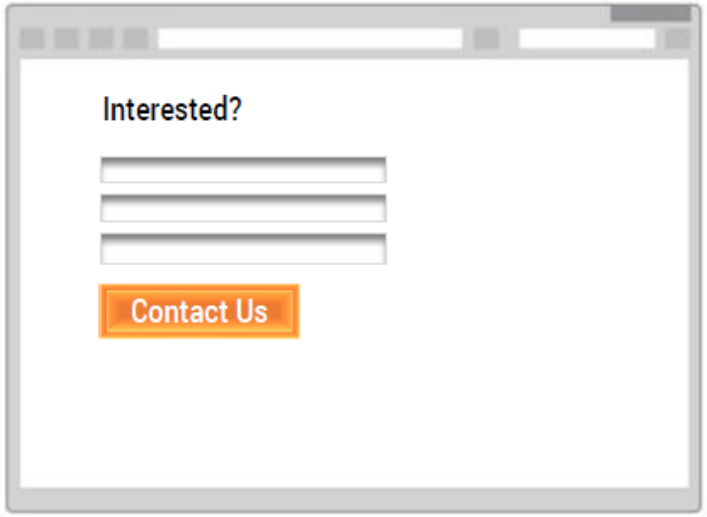


Less efforts



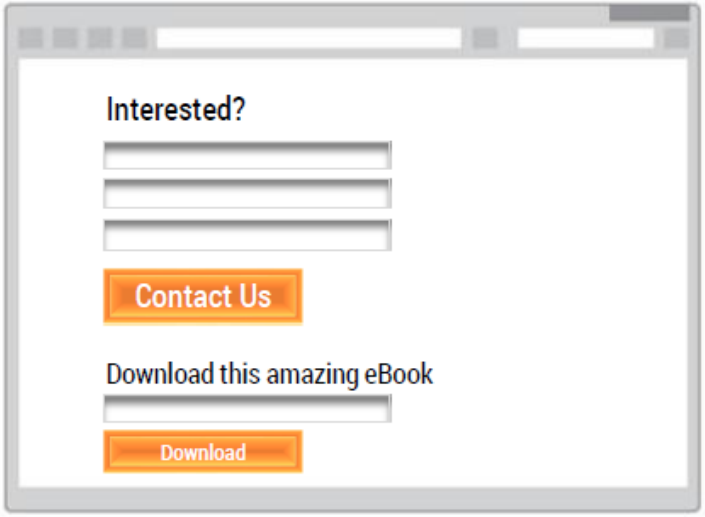
Examples: Good vs. Bad UI design

Single Call to Action



✗

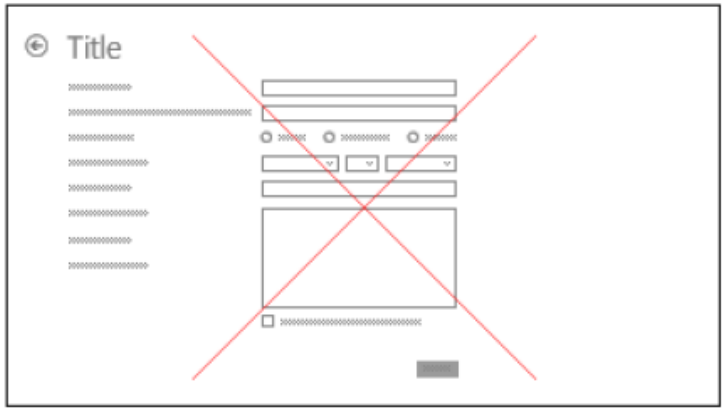
Multiple Options



✓

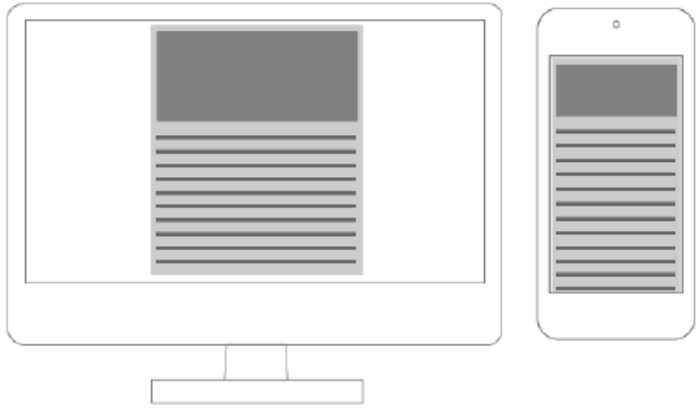
Examples: Good vs. Bad UI design

Multiple Columns



✗

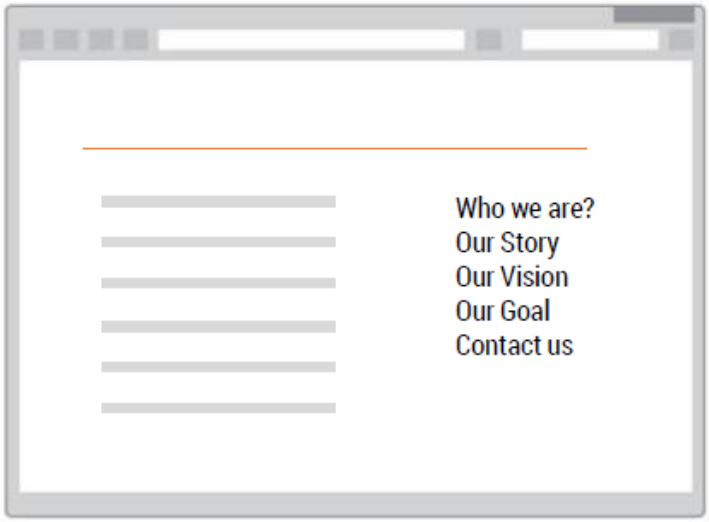
Single Column



✓

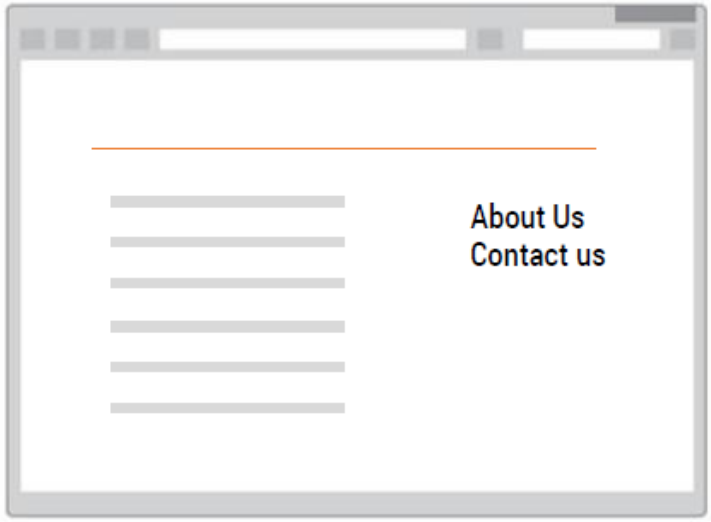
Examples: Good vs. Bad UI design

Segregated menu



✗

Merged Menu



✓

Key points

- DFD is use to represent data flow between process.
- ER diagram represent the conceptual model for the database.
- User interface (UI) design is both an art and a science.