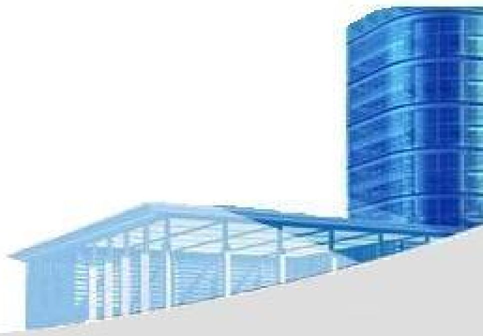
Ch.11 Requirements Modeling: Behavior, Patterns, and Web/Mobile Apps

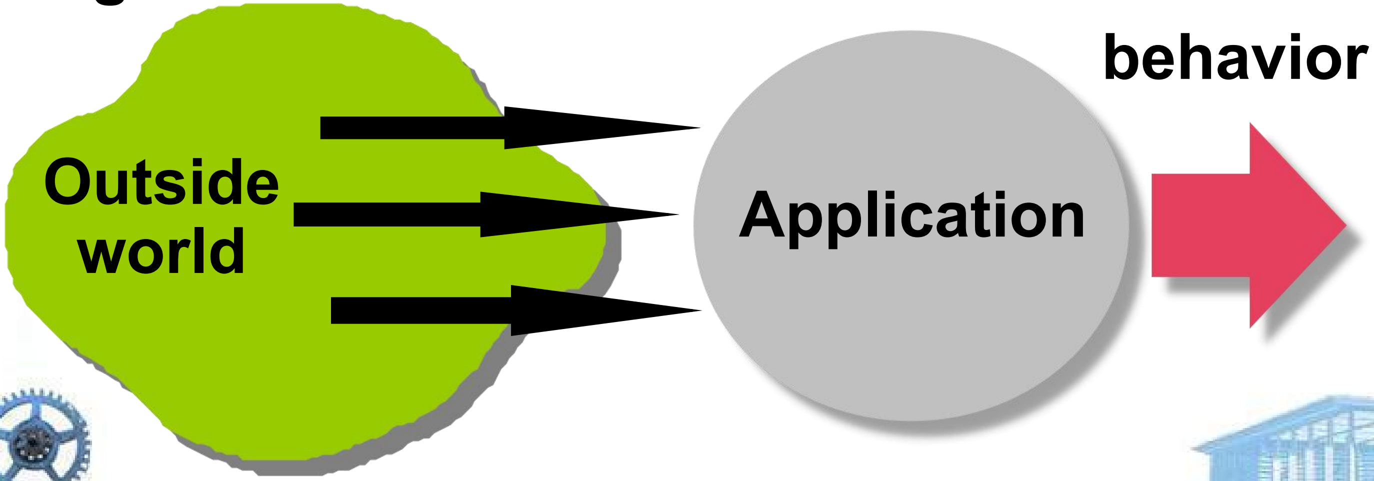


Behavioral Modeling

- The behavioral model indicates how software will respond to external events or stimuli. To create the model, the analyst must perform the following steps:
  - Evaluate all use-cases to fully understand the sequence of interaction within the system.
  - Identify events that drive the interaction sequence and understand how these events relate to specific objects.
  - Create a sequence for each use-case.
  - Build a state diagram for the system.
  - Review the behavioral model to verify accuracy and consistency.

The States of a System

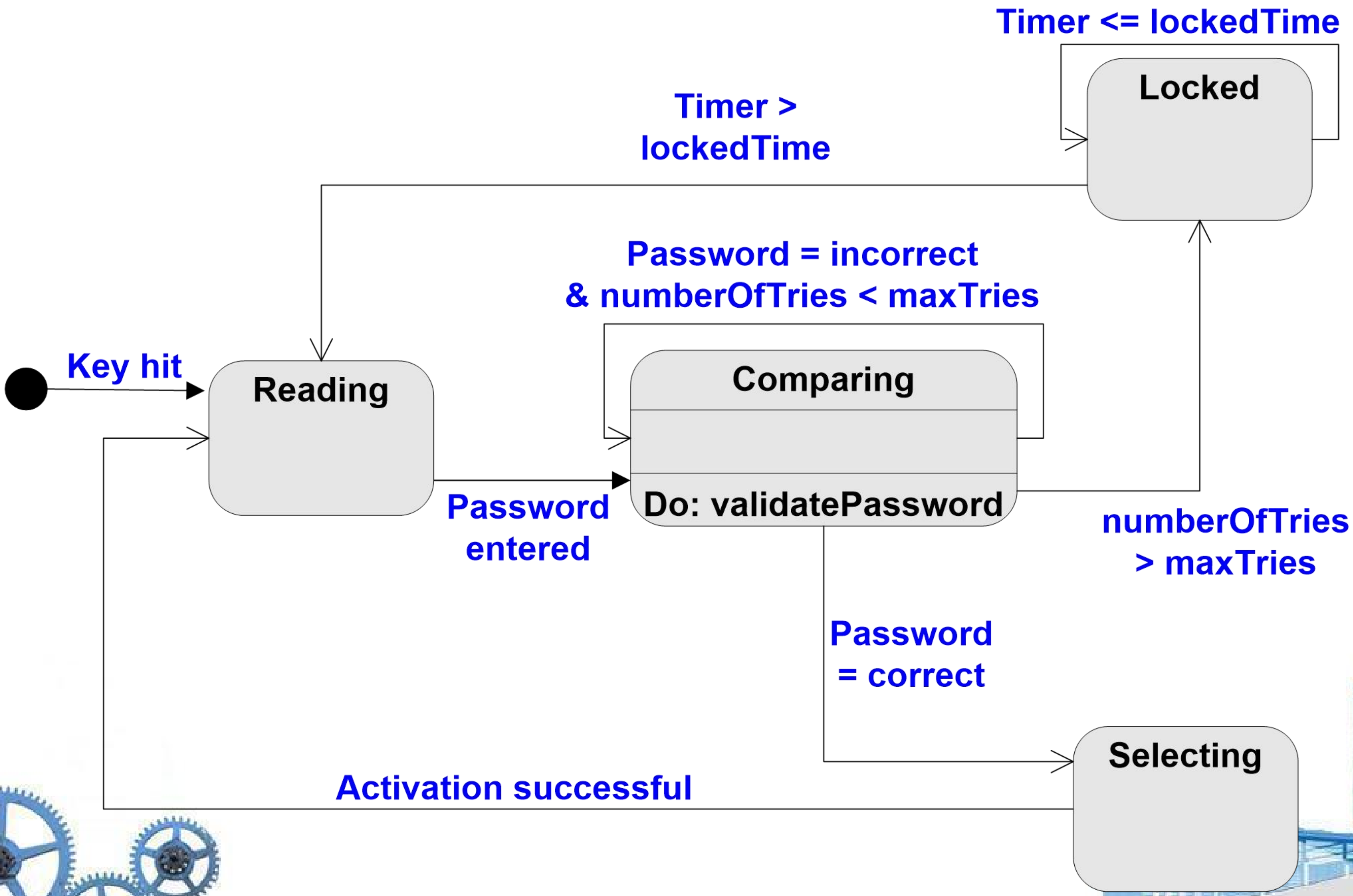
- state—a set of observable circumstances that characterizes the behavior of a system at a given time
- state transition — the movement from one state to another
- event—an occurrence that causes the system to exhibit some predictable form of behavior
- action—process that occurs as a consequence of making a transition



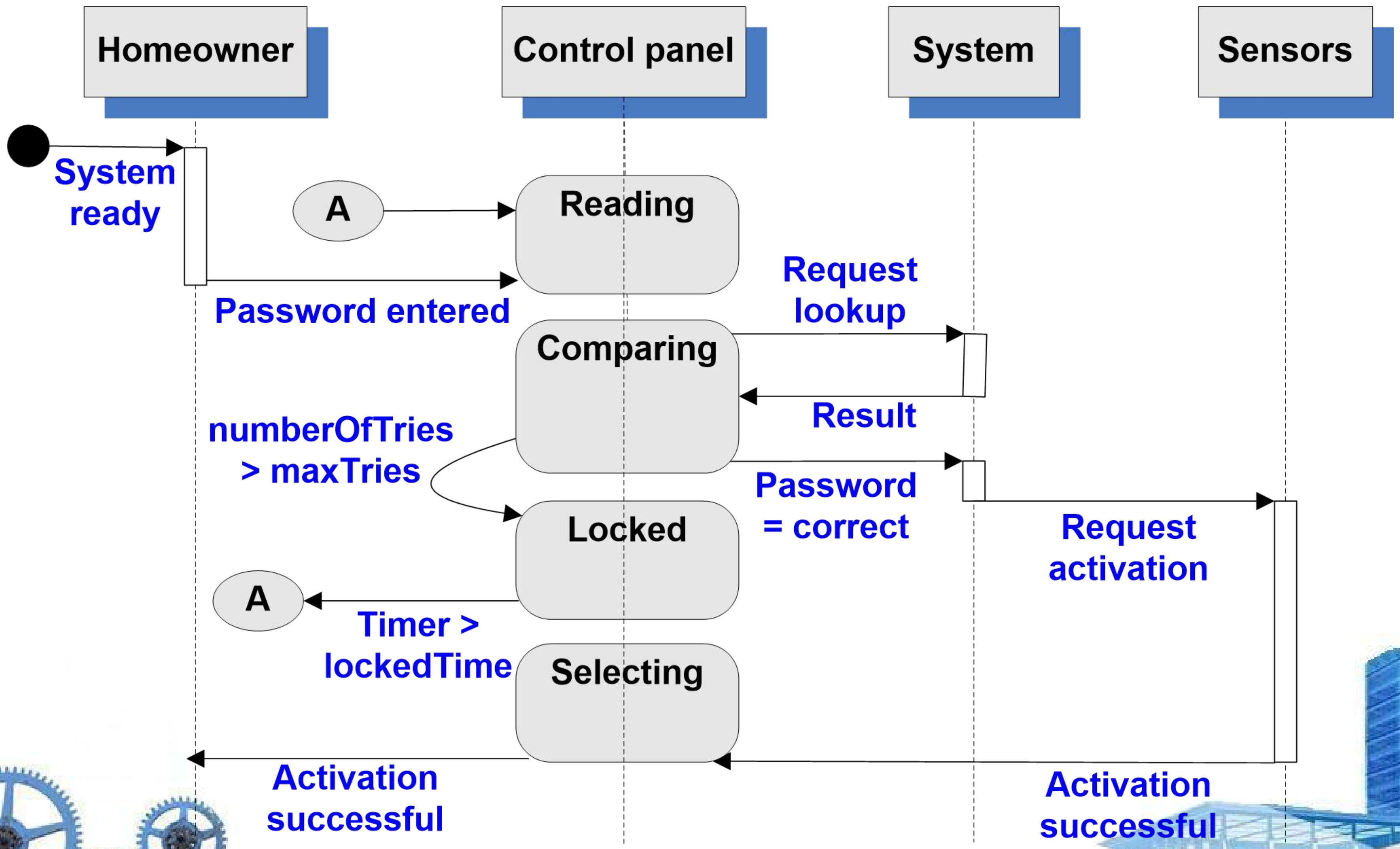
State Representations

- In the context of behavioral modeling, two different characterizations of states must be considered:
  - the state of each class as the system performs its function
  - the state of the system as observed from the outside as the system performs its function

State Diagram for the ControlPanel Class



Sequence Diagram



Data Modeling

- examines data objects independently of processing
- focuses attention on the data domain
- creates a model at the customer’s level of abstraction
- indicates how data objects relate to one another

What is a Data Object?

Object — something that is described by a set of attributes (data items) and that will be manipulated within the software system

- **Each instance of an object (e.g., a book) can be identified uniquely (e.g., ISBN#)**
- **Each plays a necessary role in the system i.e., the system could not function without access to instance of the object**
- **Each is described by attributes that are themselves data items**

Typical Objects

- external entities (printer, user, sensor)
- things (e.g, reports, displays, signals)
- occurrences or events (e.g., interrupt, alarm)
- roles (e.g., manager, engineer, salesperson)
- organizational units (e.g., division, team)
- places (e.g., manufacturing floor)
- structures (e.g., employee record)

Data Objects and Attributes

A data object contains a set of attributes that act as an aspect, quality, characteristic, or descriptor of the object

object: automobile

---

attributes:
- **make**
- **model**
- **body type**
- **price**
- **options code**

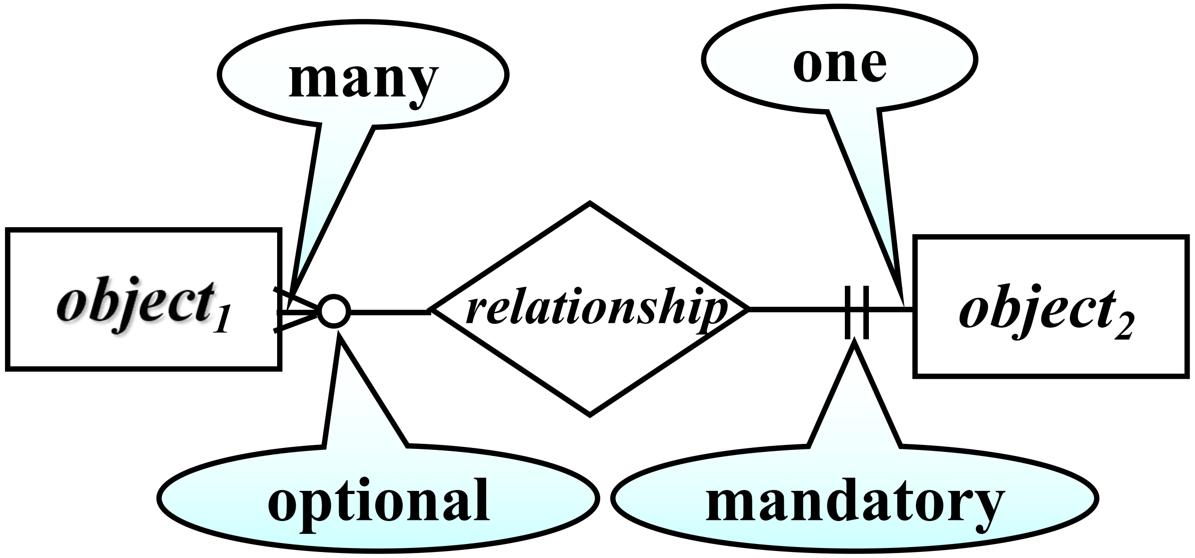
What is a Relationship?

Relationship — indicates “connectedness”; a "fact" that must be "remembered" by the system and cannot or is not computed or derived mechanically

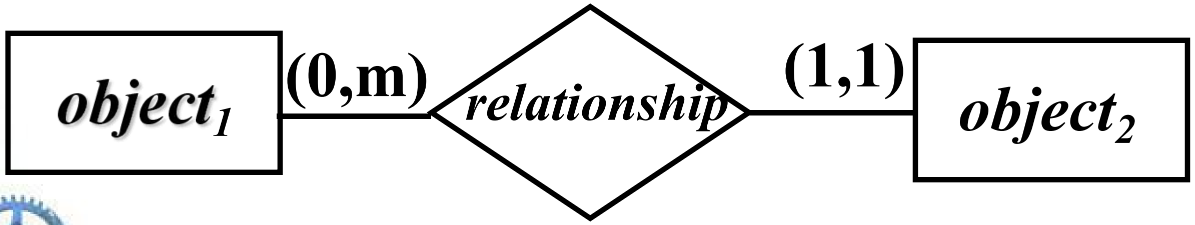
- **several instances of a relationship can exist**
- **objects can be related in many different ways**

ERD Notation

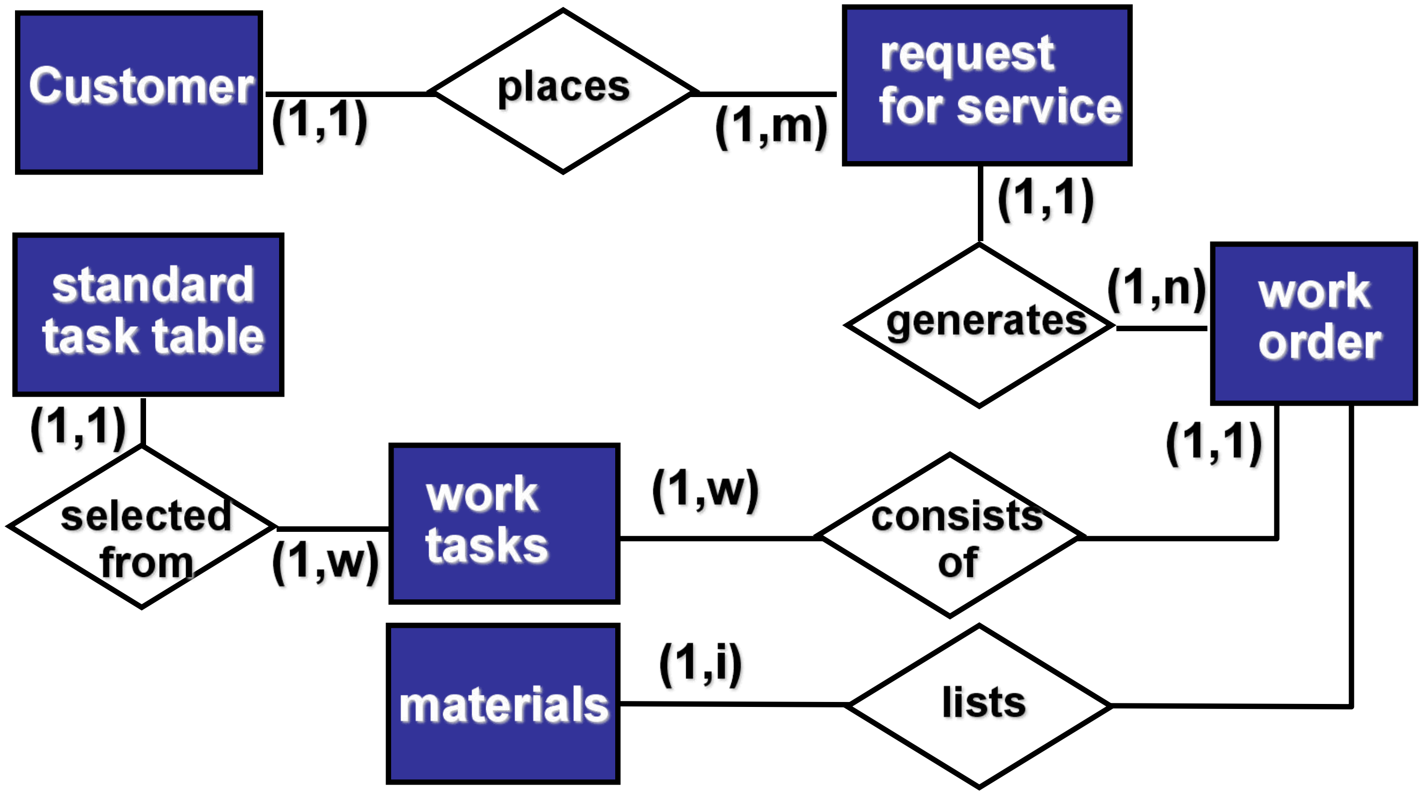
- **Cardinality**: One-to-one, One-to-many, Many-to-many
- **Modality**: Mandatory, optional



or



The ERD: An Example

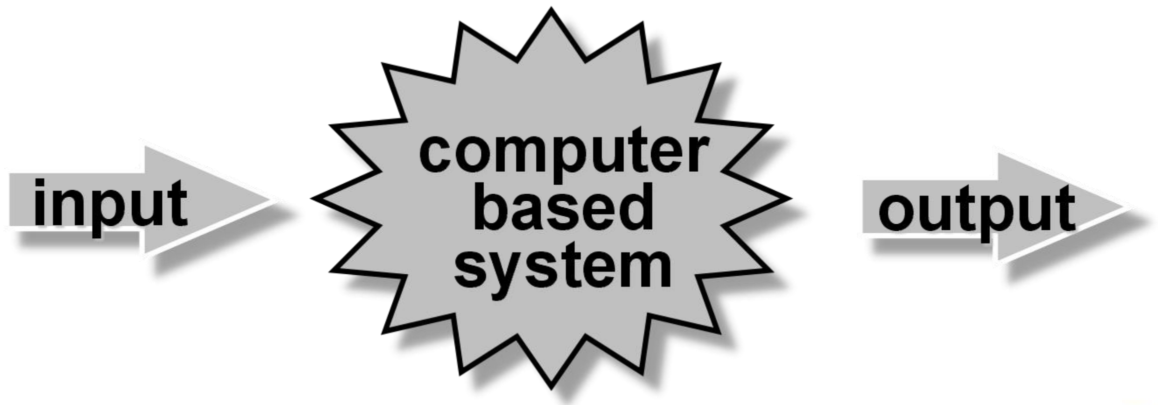


Flow-Oriented Modeling

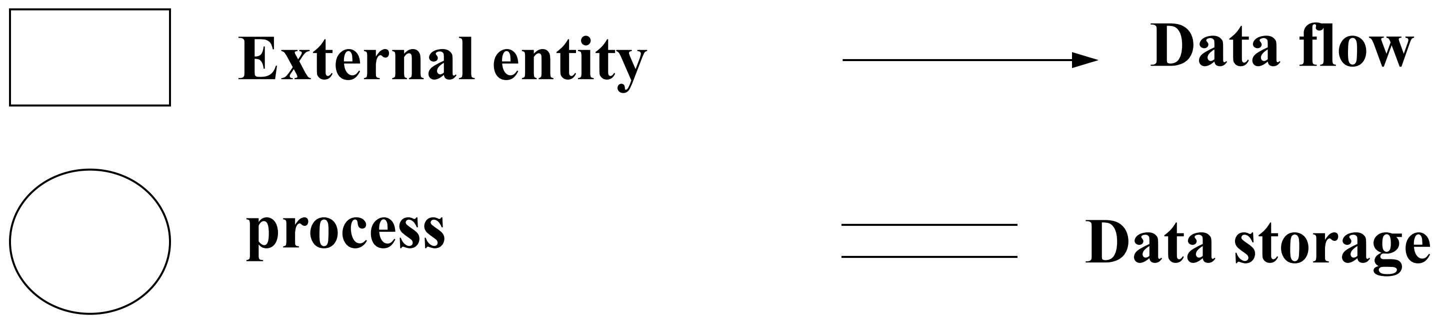
- Represents how data objects are transformed at they move through the system
- A data flow diagram (DFD) is the diagrammatic form that is used
- Considered by many to be an ‘old school’ approach, flow-oriented modeling continues to provide a view of the system that is unique — it should be used to supplement other analysis model elements

Data Flow Diagram (DFD)

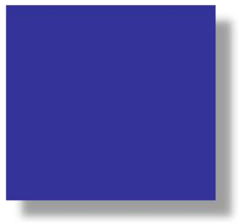
- Every computer-based system is an information transform system
- Is a graphical representation that depicts information flow and the transforms that are applied as data move from input to output
- Be used to represent a system or software at any level of abstraction



Notation of DFD



External Entity



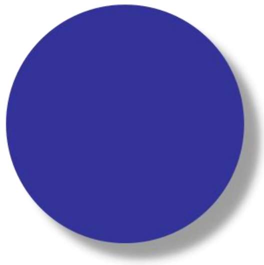
A producer or consumer of data

Examples: a person, a device, a sensor

Another example: computer-based system

Data must always originate somewhere and must always be sent to something

Process



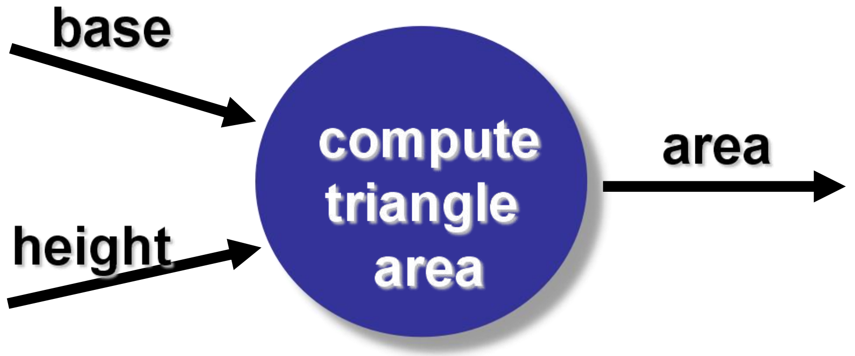
A data transformer (changes input to output)

Examples: compute taxes, determine area, format report, display graph

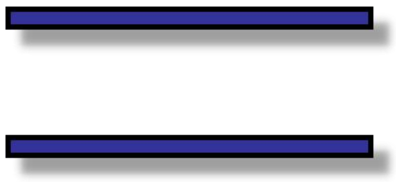
Data must always be processed in some way to achieve system function

Data Flow

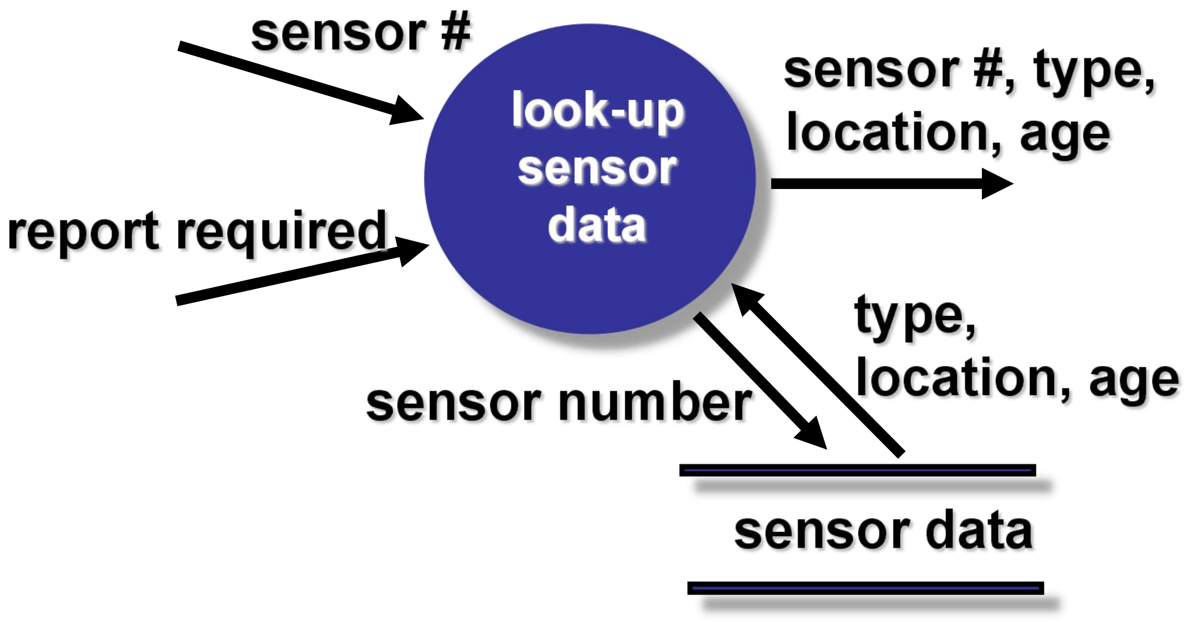
Data flows through a system, beginning as input and be transformed into output.



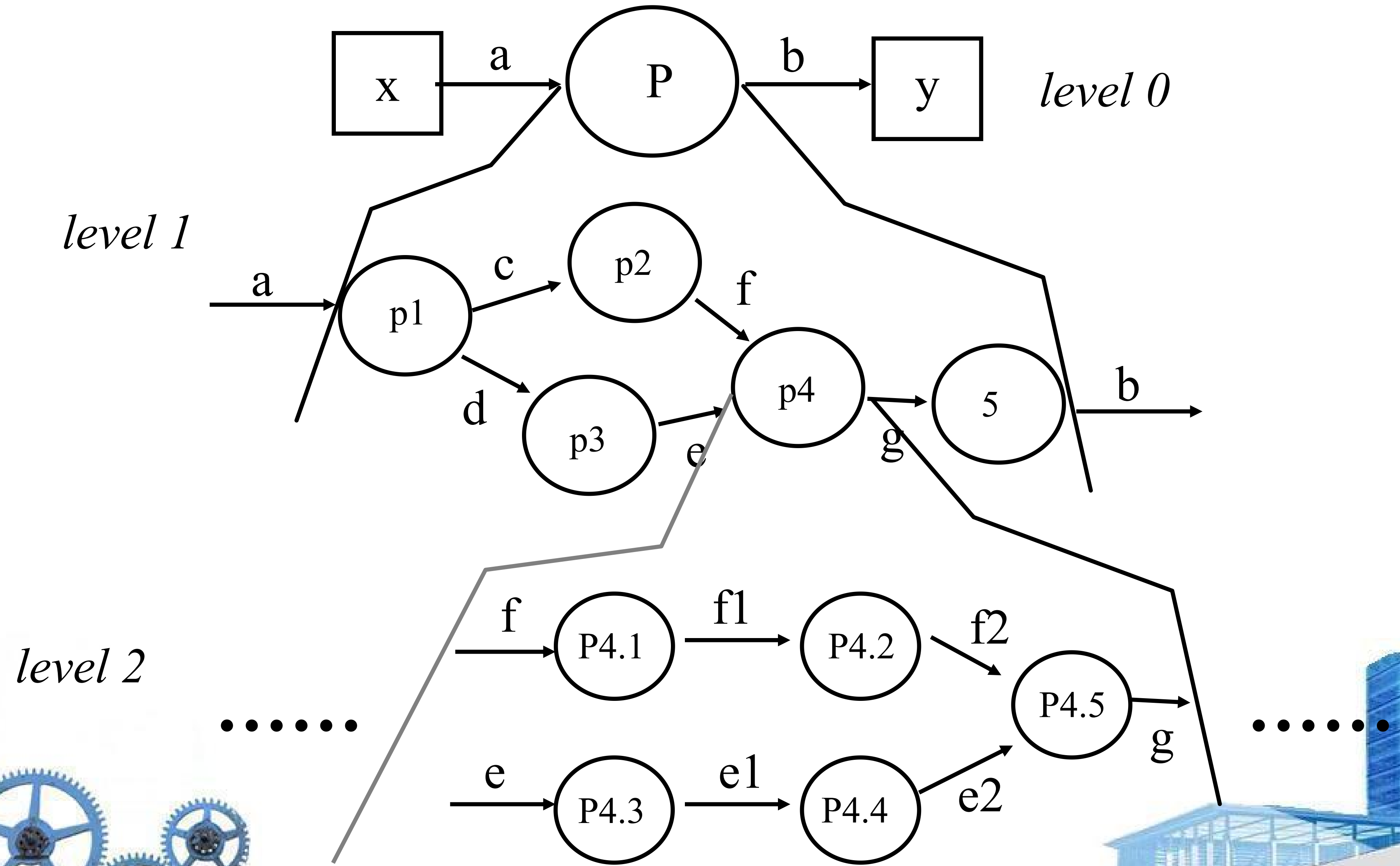
Data Stores



Data is often stored for later use.



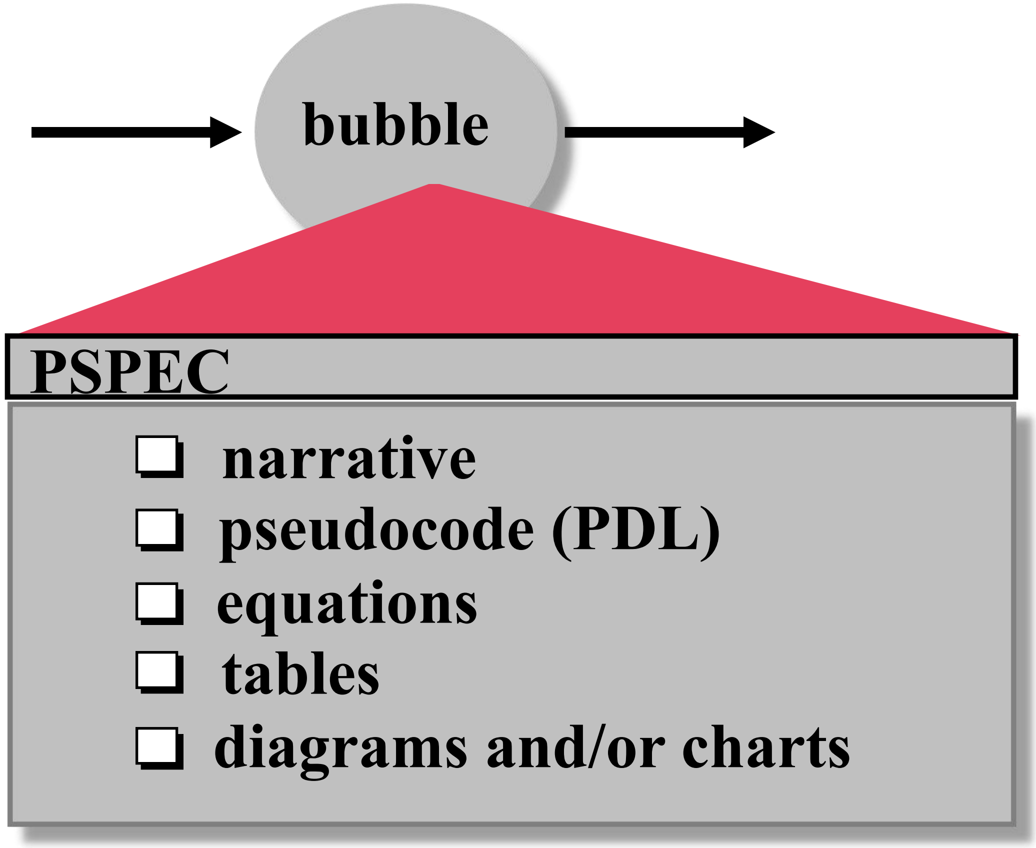
The Data Flow Hierarchy



DFD Guidelines

- always label data flow arrows, and all icons must be labeled with meaningful names
- always begin with a context level diagram (also called level 0),and show external entities at level 0
- do not represent procedural logic
- the DFD evolves through a number of levels of detail, and the information flow continuity must be maintained

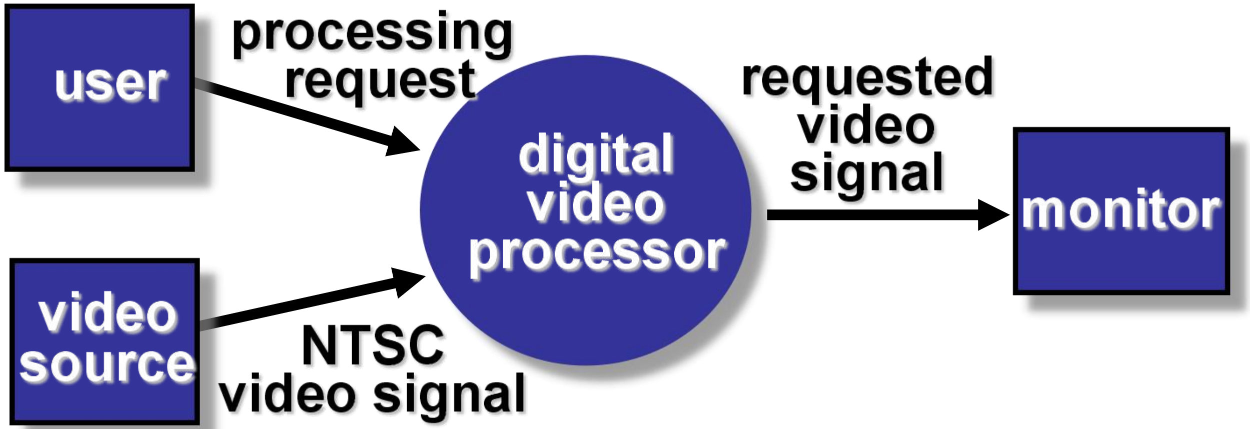
Process Specification (PSPEC)



- ❑ narrative
- ❑ pseudocode (PDL)
- ❑ equations
- ❑ tables
- ❑ diagrams and/or charts

Constructing a DFD — I

- review the data model to isolate data objects and use a grammatical parse to determine "operations"
- determine external entities (producers and consumers of data)
- create a level 0 DFD



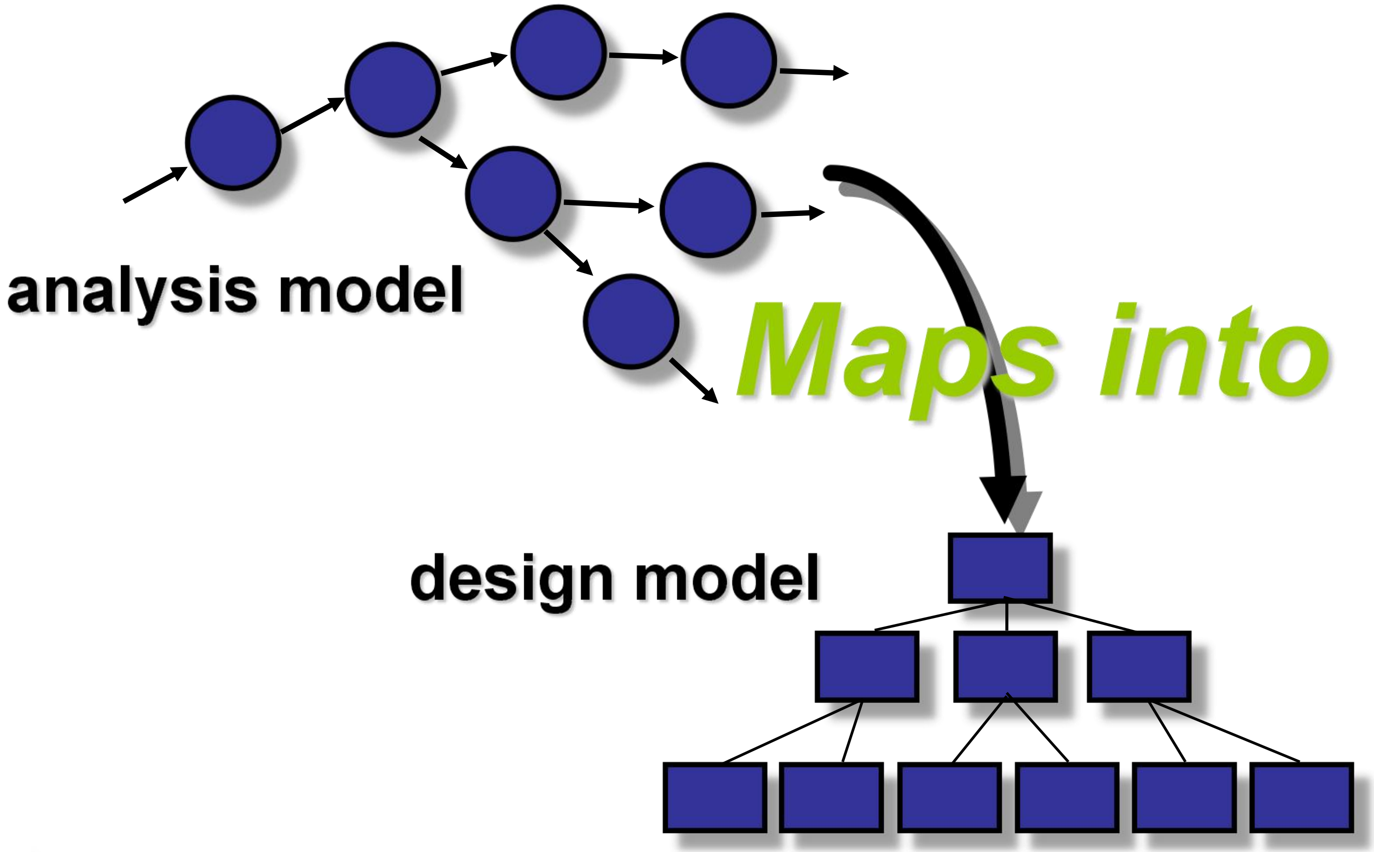
Constructing a DFD — II

- write a narrative describing the transform
- parse to determine next level transforms
- "balance" the flow to maintain data flow continuity
- develop a level 1 DFD
- use a 1:5 (approx.) expansion ratio

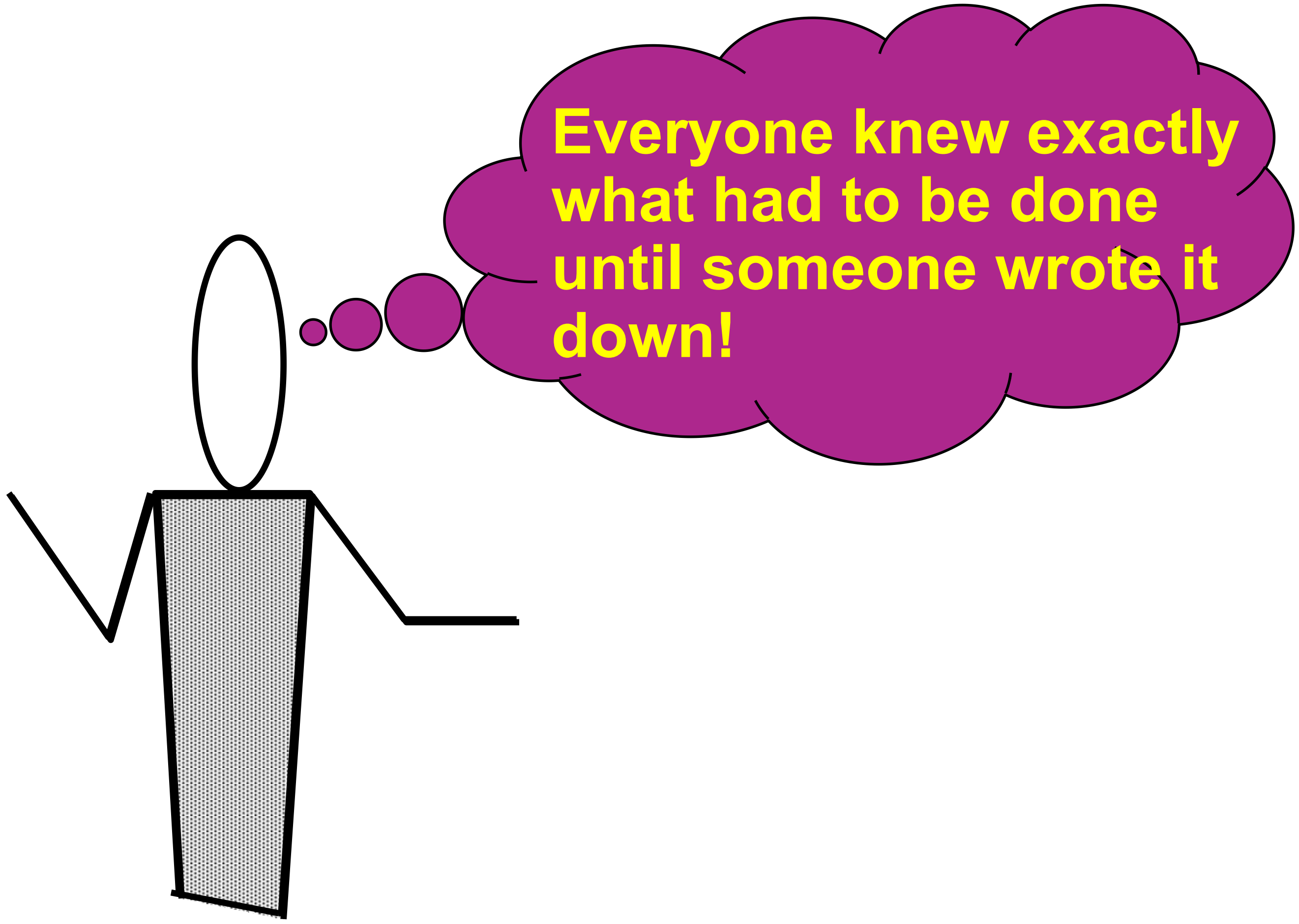
Flow Modeling Notes

- each bubble is refined until it does just one thing
- the expansion ratio decreases as the number of levels increase
- most systems require between 3 and 7 levels for an adequate flow model
- a single data flow item (arrow) may be expanded as levels increase (data dictionary provides information)

DFDs: A Look Ahead



Writing the Software Specification


Everyone knew exactly what had to be done until someone wrote it down!

Patterns for Requirements Modeling

- Software patterns are a mechanism for capturing domain knowledge in a way that allows it to be reapplied when a new problem is encountered
  - domain knowledge can be applied to a new problem within the same application domain
  - the domain knowledge captured by a pattern can be applied by analogy to a completely different application domain.
- The original author of an analysis pattern does not “create” the pattern, but rather, discovers it as requirements engineering work is being conducted.
- Once the pattern has been discovered, it is documented

Discovering Analysis Patterns

- The most basic element in the description of a requirements model is the use case.
- A coherent set of use cases may serve as the basis for discovering one or more analysis patterns.
- A semantic analysis pattern (SAP) “is a pattern that describes a small set of coherent use cases that together describe a basic generic application.” [Fer00]

Requirements Modeling for WebApps

- **Content Analysis.** The full spectrum of content to be provided by the WebApp is identified, including text, graphics and images, video, and audio data. Data modeling can be used to identify and describe each of the data objects.
- **Interaction Analysis.** The manner in which the user interacts with the WebApp is described in detail. Use-cases can be developed to provide detailed descriptions of this interaction.
- **Functional Analysis.** The usage scenarios (use-cases) created as part of interaction analysis define the operations that will be applied to WebApp content and imply other processing functions. All operations and functions are described in detail.
- **Configuration Analysis.** The environment and infrastructure in which the WebApp resides are described in detail.
- **Navigation Analysis.** The organization of web page link.

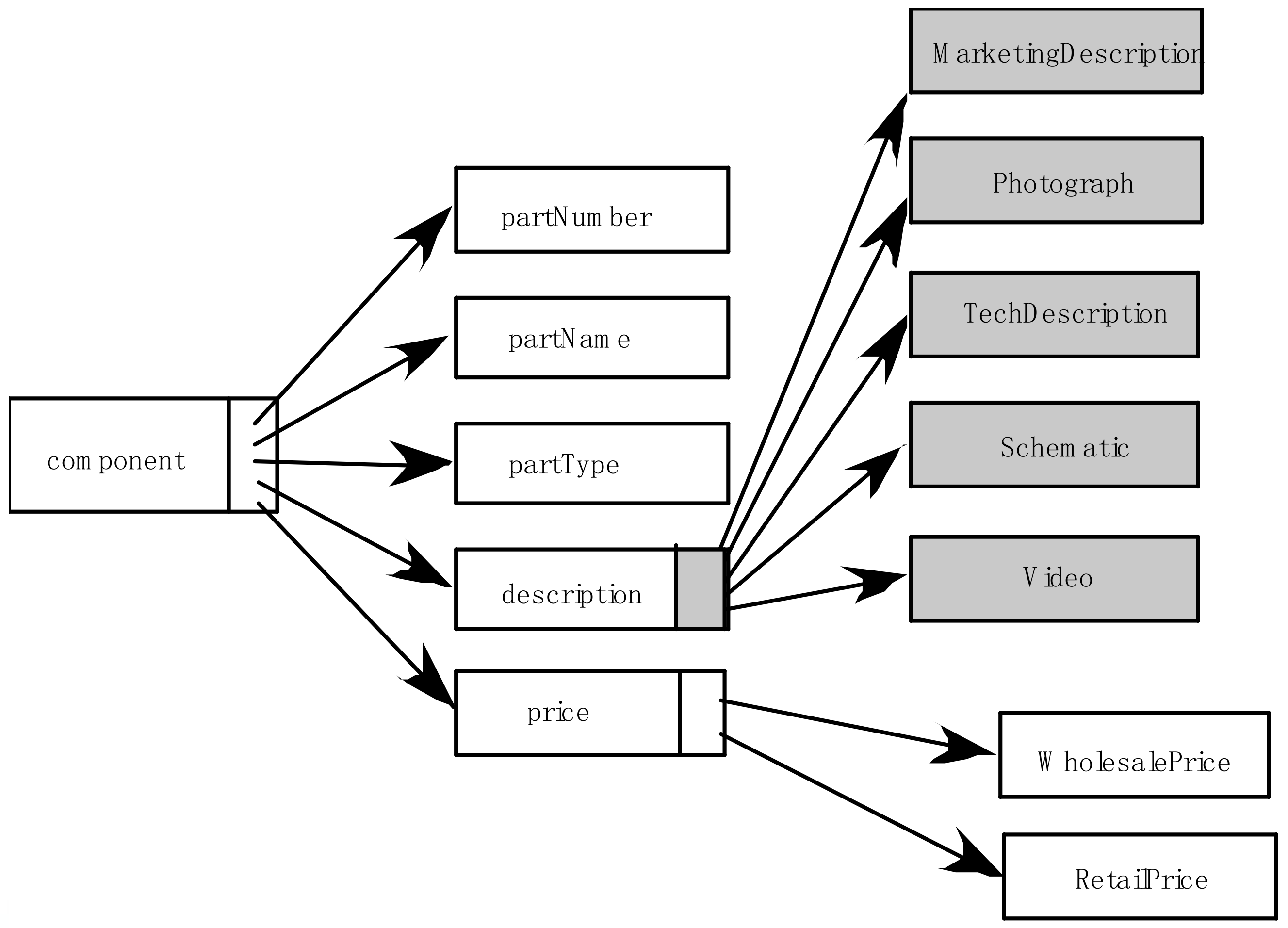
When Do We Perform Analysis?

- **In some Web/Mobile App situations, analysis and design emrge. However, an explicit analysis activity occurs when …**
  - the Web or Mobile App to be built is large and/or complex
  - the number of stakeholders is large
  - the number of developers is large
  - the development team members have not worked together before
  - the success of the app will have a strong bearing on the success of the business

The Content Model

- Content objects are extracted from use-cases
  - examine the scenario description for direct and indirect references to content
- Attributes of each content object are identified
- The relationships among content objects and/or the hierarchy of content maintained by a WebApp
  - Relationships—entity-relationship diagram or UML
  - Hierarchy—data tree or UML

Data Tree


component
partNumber
partName
partType
description
price
MarketingDescription
Photograph
TechDescription
Schematic
Video
WholesalePrice
RetailPrice

The Interaction Model

- Composed of four elements:
  - use-cases
  - sequence diagrams
  - state diagrams
  - a user interface prototype
- Each of these is an important UML notation and is described in Appendix I

The Functional Model

- **The functional model addresses two processing elements of the WebApp**
  - user observable functionality that is delivered by the WebApp to end-users
  - the operations contained within analysis classes that implement behaviors associated with the class.
- **An activity diagram can be used to represent processing flow**

The Configuration Model

- **Server-side**
  - Server hardware and operating system environment must be specified
  - Interoperability considerations on the server-side must be considered
  - Appropriate interfaces, communication protocols and related collaborative information must be specified
- **Client-side**
  - Browser configuration issues must be identified
  - Testing requirements should be defined

Navigation Modeling-I

- Should certain elements be easier to reach (require fewer navigation steps) than others? What is the priority for presentation?
- Should certain elements be emphasized to force users to navigate in their direction?
- How should navigation errors be handled?
- Should navigation to related groups of elements be given priority over navigation to a specific element.
- Should navigation be accomplished via links, via search-based access, or by some other means?
- Should certain elements be presented to users based on the context of previous navigation actions?
- Should a navigation log be maintained for users?

Navigation Modeling-II

- Should a full navigation map or menu (as opposed to a single "back" link or directed pointer) be available at every point in a user's interaction?
- Should navigation design be driven by the most commonly expected user behaviors or by the perceived importance of the defined WebApp elements?
- Can a user "store" his previous navigation through the WebApp to expedite future usage?
- For which user category should optimal navigation be designed?
- How should links external to the WebApp be handled? overlaying the existing browser window? as a new browser window? as a separate frame?

《Software Requirements Specification》+ 《Software Plan》

(2 Speaker: 10-minute presentation + 5-minute Q&A)

Note: Word doc (Chinese or English) can be used for Presentation, Ppt slide isn't required.

Due: on the head of May, 2016

Site：曹东502

Grading Policy: Each group will evaluate the other groups' performances and fill in the grading table.