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

Behavioral Modeling

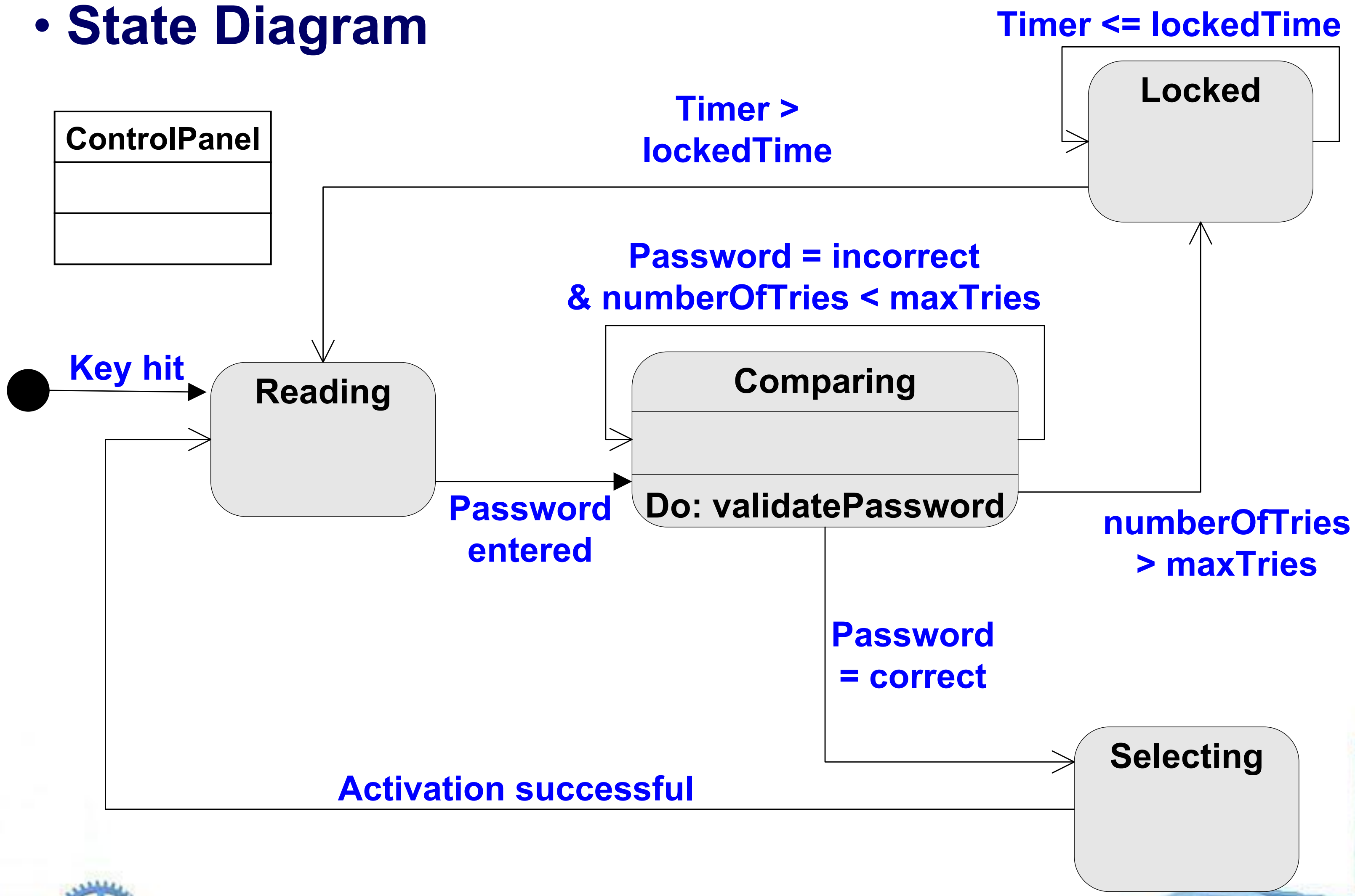
- The behavioral model indicates how software will respond to external events or stimuli. To create the model, the analyst must perform the following steps:
  1. Evaluate all **use-cases** to fully understand the sequence of interaction within the system.
  2. Identify **events** that drive the interaction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

Behavioral Modeling

- 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 and
  - ***the state of the system*** as observed from the outside as the system performs its function

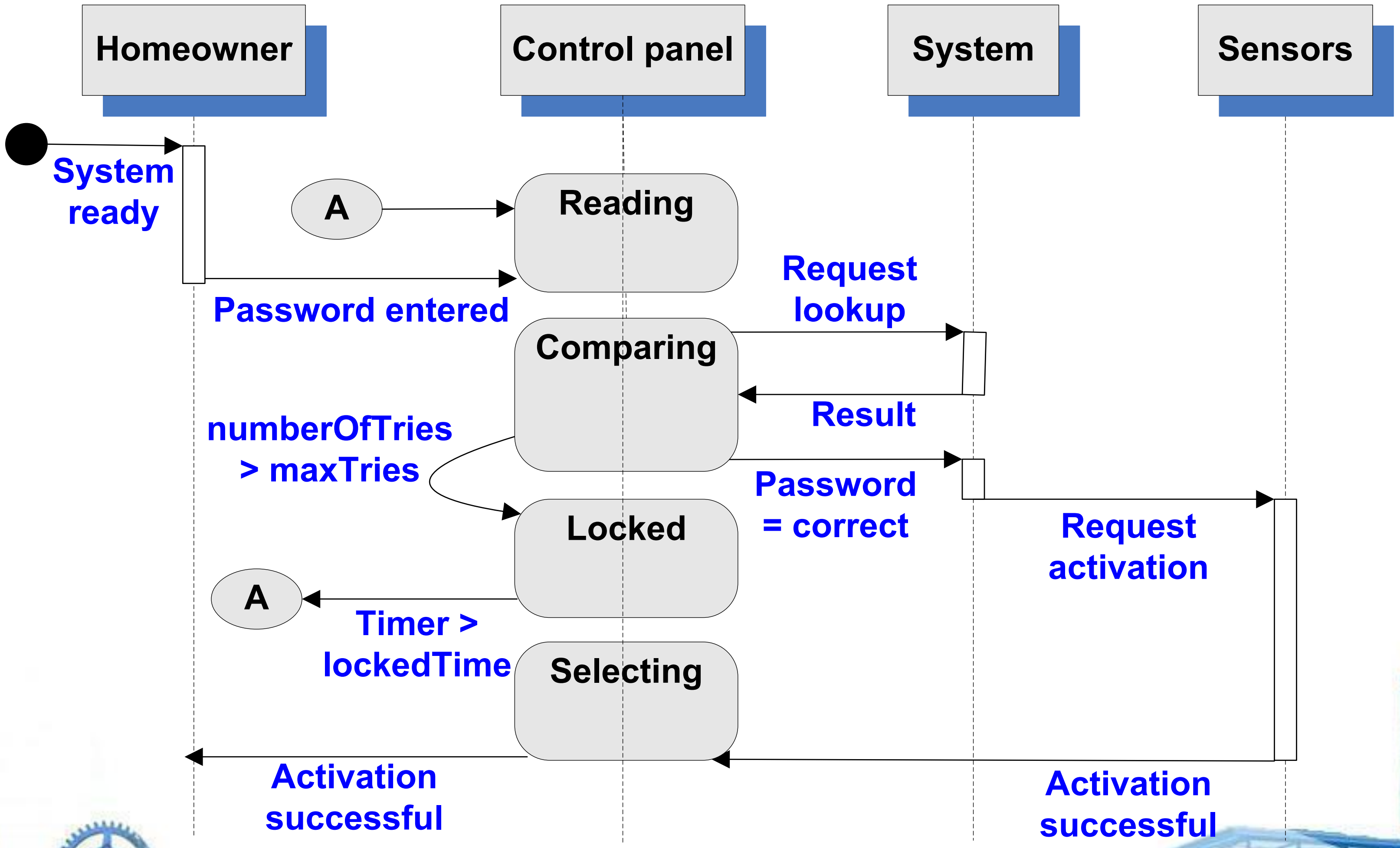
Behavioral Modeling

- **State Diagram**

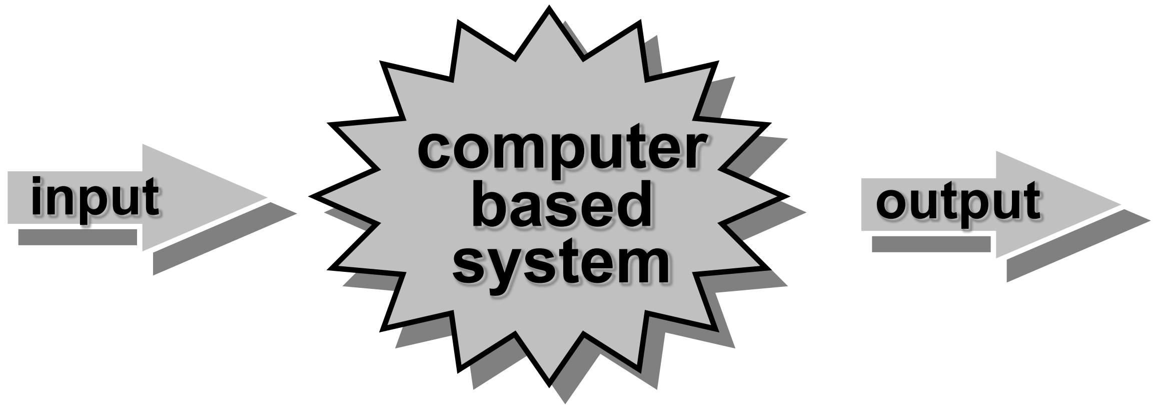


Behavioral Modeling

- **Sequence Diagram**

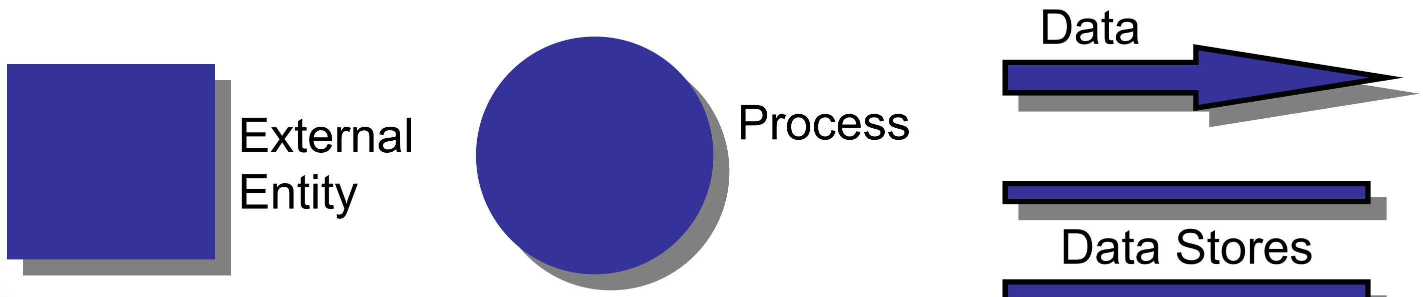


Flow-Oriented Modeling



System = data + function

- Data Flow Diagram



Flow-Oriented Modeling

- **Example:** [From《*Fundamentals of Software Engineering*》]

Information System of a Public Library

if { user requests a book (title, author, user’s name) }

{ **Get a book** }

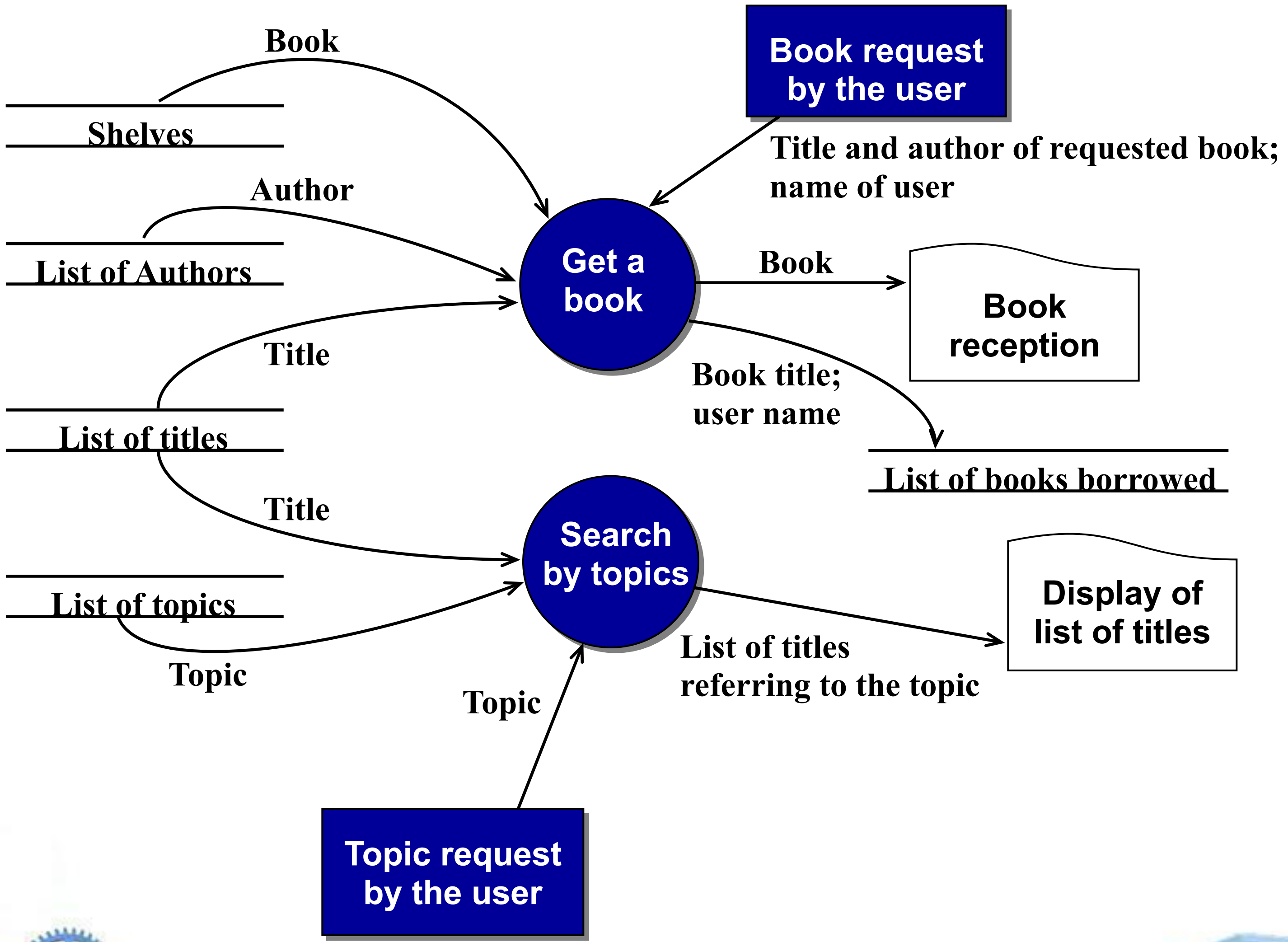
→ book, and user’s list of books borrowed;

if { user searches a book by topics }

{ **Search by topics** }

→ list of book titles referring to the topic.

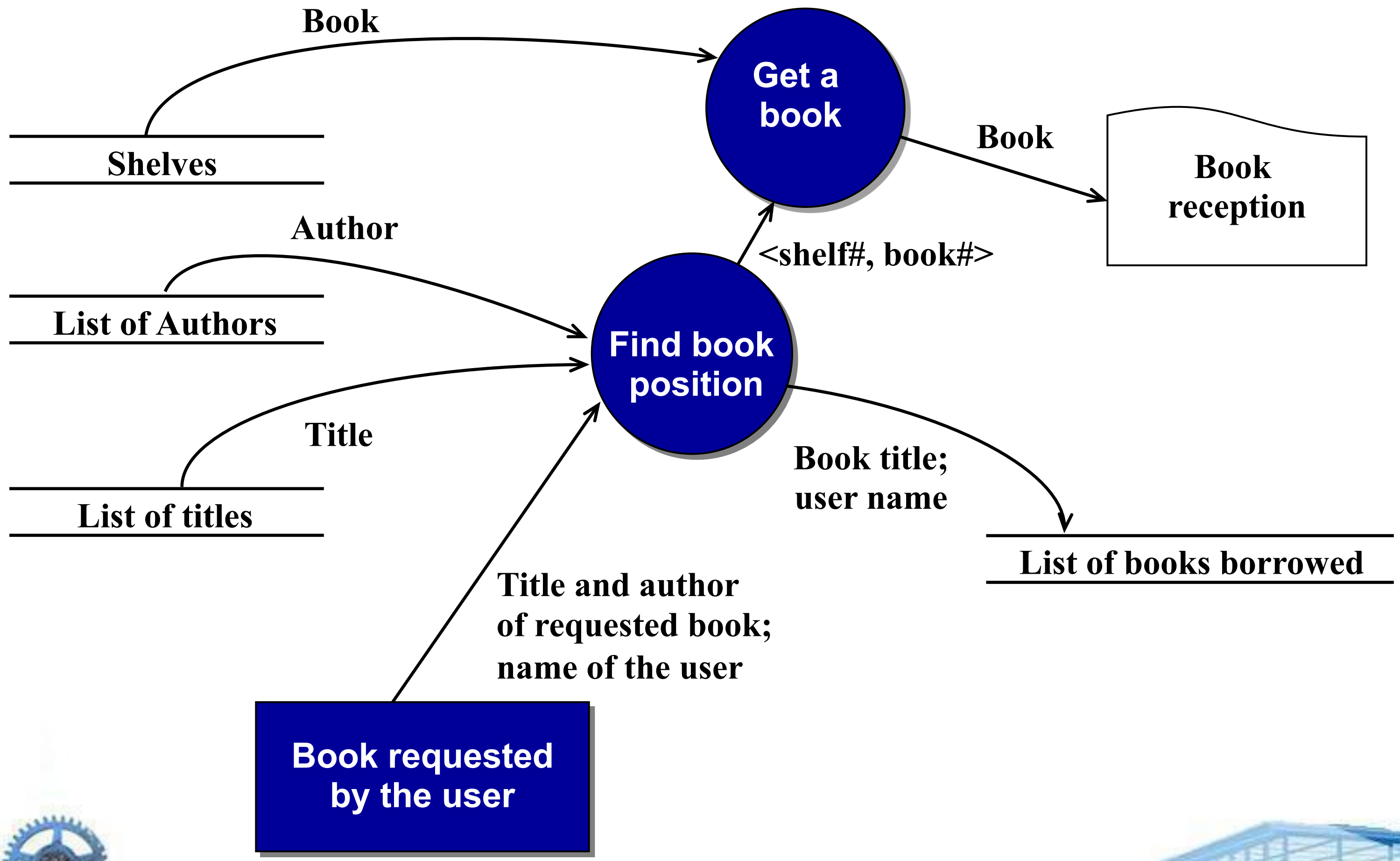
Flow-Oriented Modeling



Flow-Oriented Modeling

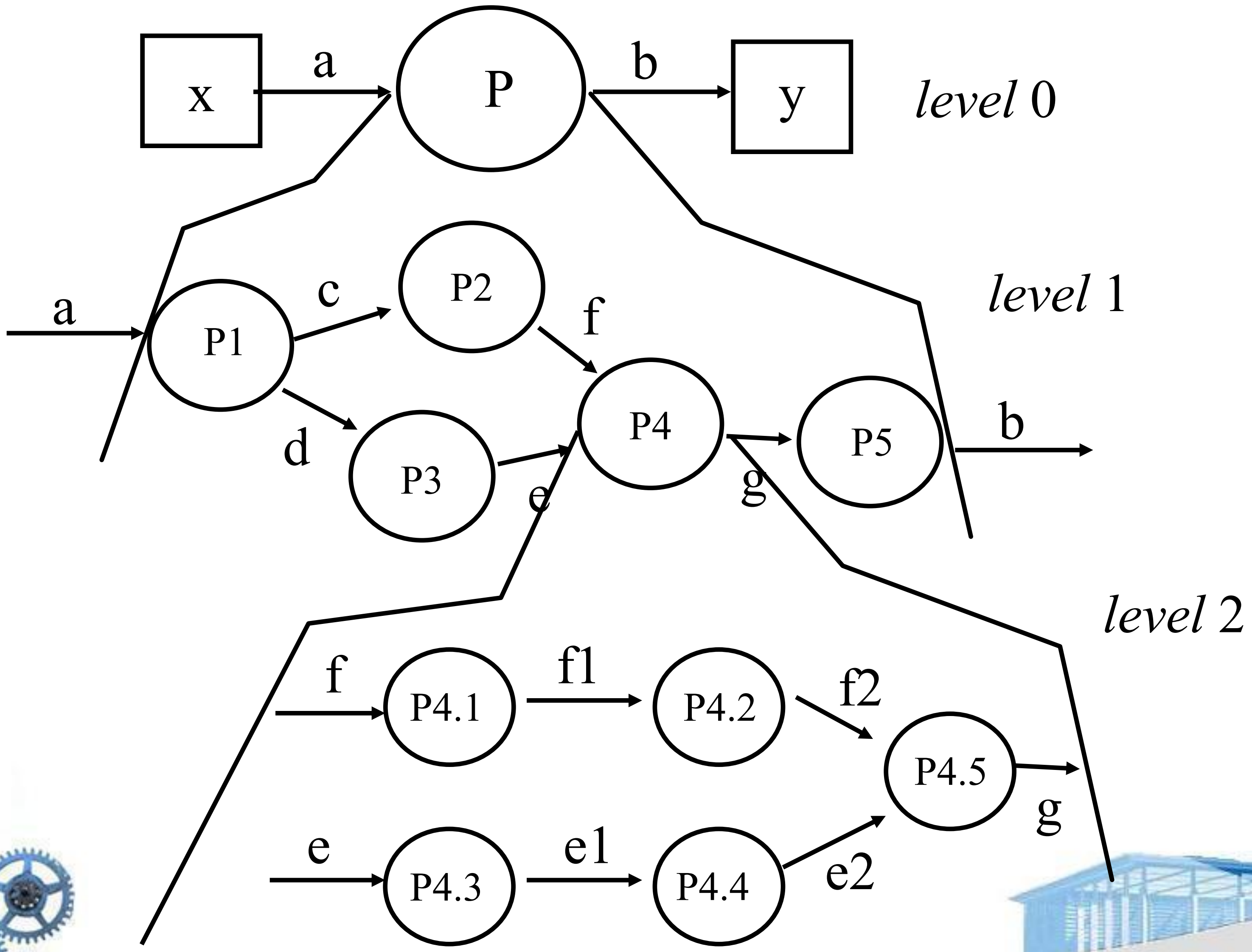
- **Refinement:**

Book request = Find book position + Get a book

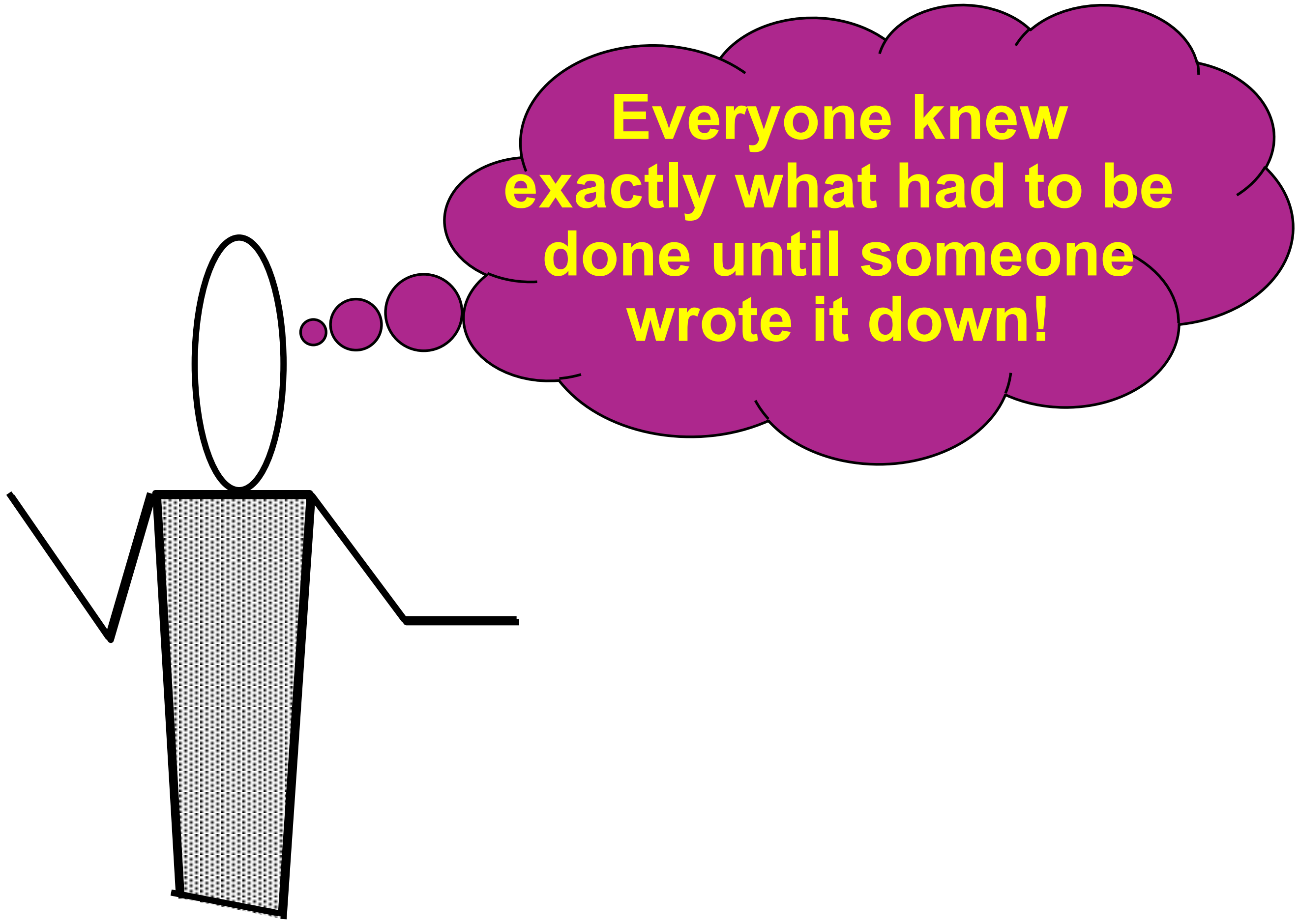


Flow-Oriented Modeling

- **The Data Flow Hierarchy**



Writing the Software Specification


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

Specification Guidelines

- use a layered format that provides increasing detail as the "layers" deepen
- use consistent graphical notation and apply textual terms consistently (stay away from aliases)
- be sure to define all acronyms
- be sure to include a table of contents; ideally, include an index and/or a glossary
- write in a simple, unambiguous style
- always put yourself in the reader's position, "Would I be able to understand this if I wasn't intimately familiar with the system?"

《Software Requirements Specification》

Due: 22:00 on April 19th, 2015

Minimum requirement of contents:

Introduction (2 points);
User Scenarios(8 points); Data Flow Diagram (7 points); State Diagrams(5 points); Class Diagrams(5 points) and CRC Cards (5 points);
Validation Criteria (15 points).

Concerned points:

The accuracy of the validation criteria: full marks can be obtained if more than 90% of the functions are covered. The acceptance testing of the subsystem version 1.0 will strictly go by the criteria. The language and style of the document must be uniformed (3 points).

Grading: The full mark = **50 points × number of participants**

Requirements Modeling for WebApps

- When do we perform analysis?
  - 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

Requirements Modeling for WebApps

- **Content Analysis** – describe
  - *text*
  - *graphics and images*
  - *video*
  - *audio*
- **Interaction Analysis** – use-cases
- **Functional Analysis** – use-cases that define
  - the operations that will be applied to WebApp content
  - imply other processing functions
- **Configuration Analysis** – environment and infrastructure
- **Navigation Analysis** – focus on overall requirements

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?