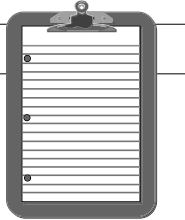


### System Design Issues

- Three-tier architecture
- UML packages
- Physical design
- Interface patterns
- Indirect communication
- Application coordinators
- Storage and persistence




---

---

---

---

---

---

---

---

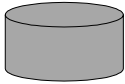
### Three-Tier Architecture



Presentation



Application logic (business rules)



Storage (database)

---

---

---

---

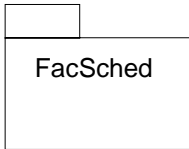
---

---

---

---

### UML Packages



Represents a group of elements or subsystems

- Application logic
- GUI
- Etc.

In Rose, used in both logical and component views:

- Logical package (class category)
- Component package (subsystem)

Often named the same

---

---

---

---

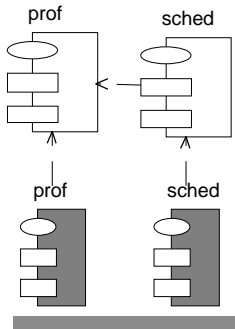
---

---

---

---

### Physical Design



- Assign classes to files, directories  
Rose component diagram
- Package
  - Module specification
    - “.h” file
  - Module body
    - “.cpp” file
  - Dependencies

---

---

---

---

---

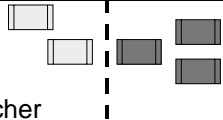
---

---

---

### Interface Patterns

- **Facade**
  - Single interface class
  - Acts as broker/dispatcher
  - Multiple (hidden) objects in package
- **Model-view separation**
  - Domain classes (model)
  - Presentation classes (view)
    - Thin layer over windows, etc.




---

---

---

---

---

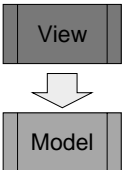
---

---

---

### Indirect Communication

- **View can see model**
  - Invoke operations
  - Retrieve data for display
- **Can model see view?**
  - Generally, no; low coupling
  - What if model must notify view?
    - E.g., updated data available
- **Need indirect communication**




---

---

---

---

---

---

---

---

Publish-Subscribe Pattern



- **Separate event manager class**
  - Clearing house for “events”
- **Recipient “subscribes” to specified event(s)**
- **Originator “publishes” event(s)**
  - Doesn’t know subscriber!
- **Notification passed to recipient**
  - How exactly does this work?

---

---

---

---

---

---

---

---

Application Coordinator

- **Intermediary class**
  - Mediates between model and view
    - E.g., MFC “document” class
- **Responsibilities**
  - Map information
  - Respond to events
  - Manage windows (e.g., MFC “view”)

---

---

---

---

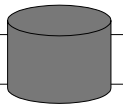
---

---

---

---

Storage and Persistence



- **Normal objects are transient**
  - Disappear when program exits
- **How to make them persistent?**
  - Relational/object database
  - Application data file
    - MFC “serialization” mechanism
    - File open/save
    - What’s the trick on “open”?

---

---

---

---

---

---

---

---