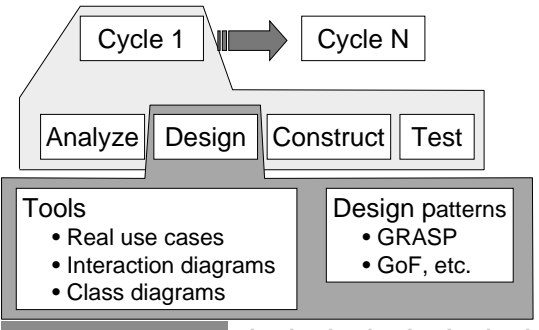


Design Details




---

---

---

---

---

---

---

---

Class Responsibilities

- **Know**
  - Private data
  - Related objects
  - Derived or calculated information
- **Do**
  - Do something itself
  - Initiate action in other objects
  - Control and coordinate actions of other objects

Typically, methods (e.g., C++ member functions) are used to implement responsibilities.

---

---

---

---

---

---

---

---

Patterns 

“Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

Christopher Alexander

---

---

---

---

---

---

---

---

Pattern Elements

- **Pattern name**
- **Problem**
  - When to apply the pattern
- **Solution**
  - Design elements
- **Consequences**
  - Results and trade-offs

From *Design Patterns*  
by Erich Gamma, Richard Helm,  
Ralph Johnson, and John Vlissides  
[the "Gang of Four" (GoF)]

---

---

---

---

---

---

---

---

---

---

Initial GRASP Patterns

- **Expert**
  - **Creator**
  - **High Cohesion**
  - **Low Coupling**
  - **Controller**
- Used to assign class responsibilities
- Pretty simple to be called "patterns", but still useful concepts

---

---

---

---

---

---

---

---

---

---

Expert

- **State responsibility**
  - "Know" or "Do"
- **Assign to the "expert" class**
  - That has information needed
  - To fulfill the responsibility

---

---

---

---

---

---

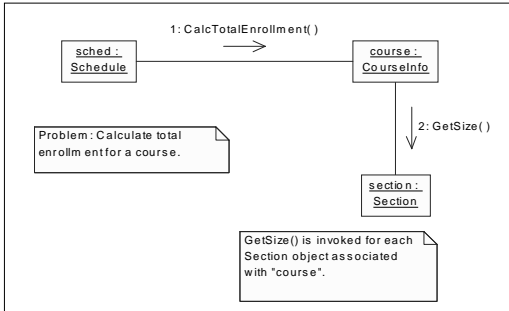
---

---

---

---

### Expert Example




---

---

---

---

---

---

---

---

### Creator

- **Responsibility**
  - Creating an instance of class A
- **Assign to class B where**
  - B aggregates/contains A objects
  - B records instances of A objects
  - B closely uses A objects
  - B has initializing data for A objects
    - Variation of Expert pattern

---

---

---

---

---

---

---

---

### Low Coupling



- **Assign responsibility**
  - Any specific responsibility
- **To maintain low coupling**
- **What is coupling?**
  - "Connectedness" to other classes
  - Knowledge of other classes
  - Reliance upon other classes

---

---

---

---

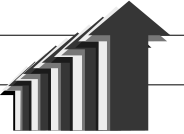
---

---

---

---

High Cohesion



- **Assign responsibility**
  - Again, any specific responsibility
- **To maintain high cohesion**
- **What is (functional) cohesion?**
  - Strong relationship between class responsibilities
  - Elements work together to provide some well-bounded behavior

---

---

---

---

---

---

---

---

Controller



- **Assign responsibility**
  - For handling a system event
- **To a class that represents:**
  - The overall system or organization
  - Some real-world active “thing”
  - An artificial handler for all system events in a use case
- **Danger: easy to do it wrong!**

---

---

---

---

---

---

---

---