Designing Architectures

Instructor: Yongjie Zheng

CS 5553: Software Architecture and Design

How do we design?

- Creativity
  - This requires extensive experience and broad training.
  - May be nurtured, but may not be taught.
- Principles, process, and methods
  - Goals, activities, and principles
  - Process
  - Design methods: object-oriented design, function-oriented design, and quality-driven design.
- Reuse of existing experiences and results
  - Horizontal reuse: architecture patterns and styles
  - Vertical reuse: domain-specific software architecture

Goals (Considerations) of Architecture Design

Conceptual Integrity

The fact that a software product presents to each of its users <u>a coherent mental model</u> of the application, of strategies for doing the application, and of the user-interface tactics to be used in specifying actions and parameters. The conceptual integrity of the product is the most important fact in ease of use. - [Fred Brooks]

Conceptual integrity implies that the similar or same design decisions are made to solve a collection of similar problems for the same goal.

Goals of Architecture Design, cont.

- Basic requirement: the designed architecture should meet all the functional requirements of the system.
- Additional requirements: completeness, consistency, compatibility, and correctness.
- More advanced quality requirements
  - Deployment qualities: maintainability, reusability, flexibility, demonstrability, ...
  - Operational qualities: performance, reliability, robustness, fault-tolerance, ...
  - It is usually hard to achieve all these qualities, and tradeoff has to be made at some point.

Activities of Architecture Design

- Analyzing and refining requirements
- Decomposing the system into components
- Selecting protocols for communication,synchronization, and data access
- Developing global structures
- Designing component internal structures
- Selecting among design alternatives (often needs to consider non-functional properties)
- Dealing with deployment issues
- Making stakeholder related decisions

Design Principles

- **Abstraction**: we concentrate on the essential features and ignore, abstract from, details that are not relevant at the level we are currently working.
- **Modularity**: the degree (cohesion, coupling) to which a system is partitioned into components (or modules).
  - Cohesion: a measure of the mutual affinity of the elements of a component. E.g. functional cohesion, data cohesion.
  - Coupling: a measure of the strength of the inter-component connections. E.g. control coupling, data coupling.
  - High-cohesion and loose-coupling are generally preferred.

Design Principles, cont.

- **Information hiding** (separation of concerns): one begins with a list of difficult design decisions or design decisions which are likely to change. Every module (component) is designed to hide such a decision from all others. Its interface or definition was chosen to reveal as little as possible about its inner workings. [Parnas]

Engineering Design Process (from the textbook)

- Feasibility stage: identifying a set of feasible concepts for the design as a whole.
- Preliminary design stage: selection and development of the best concept.
- Detailed design stage: development of engineering descriptions of the concept.
- Planning stage: evaluating and altering the concept to suit the requirements of production, distribution,consumption, and product retirement.

Potential Problems

- If the designer is unable to produce a set of feasible concepts, progress stops.
- As problems and products increase in size and complexity, the probability that any one individual can successfully perform the first two steps decreases.
- The standard approach does not directly address the situation where system design is at stake, i.e. when relationship between a set of products is at issue.

Alternative Design Strategies

- Cyclic
  - Process can revert to an earlier stage.
- Parallel
  - Independent alternatives are explored in parallel.
- Adaptive (“lay tracks as you go”)
  - The next design strategy of the design activity is decided at the end of a given stage.
- Incremental
  - Each stage of development is treated as a task of incrementally improving the existing design.

A Rational Design Process [Parnas]

- Establish and document requirements.
- Divide the system into modules.
  - The principle of "information hiding".
- Design and document the module interfaces.
  - The interface specifications should not reveal the design decisions that the module encapsulates.
- Design and document the use hierarchy.
  - Avoid "loops" in the created structure.
- Design and document the module internal structures.

Another Design Process

- First, start with the big picture showing your system and the **context** that it interfaces with.
- Second, draw simple block diagrams showing all of the **containers** that make up the system.
- Third, zoom in and decompose each container into **components** and their interactions.
- (Optional) Fourth, Further decompose the major components into **classes** using object-oriented design principles.

Adapted from "Software Architecture for Developers" by Simon Brown.

Design Methods

- Object-oriented design
  - Views a system as a group of objects that interact to satisfy system requirements.
  - Combines data and methods into a cohesive entity.
  - The state information is distributed among system objects.
- Function-oriented design
  - Views a system as a transformation function that transforms specified input to specified output.
  - Separates data and procedures, and models them separately.
  - The state information is usually in centralized data stores.

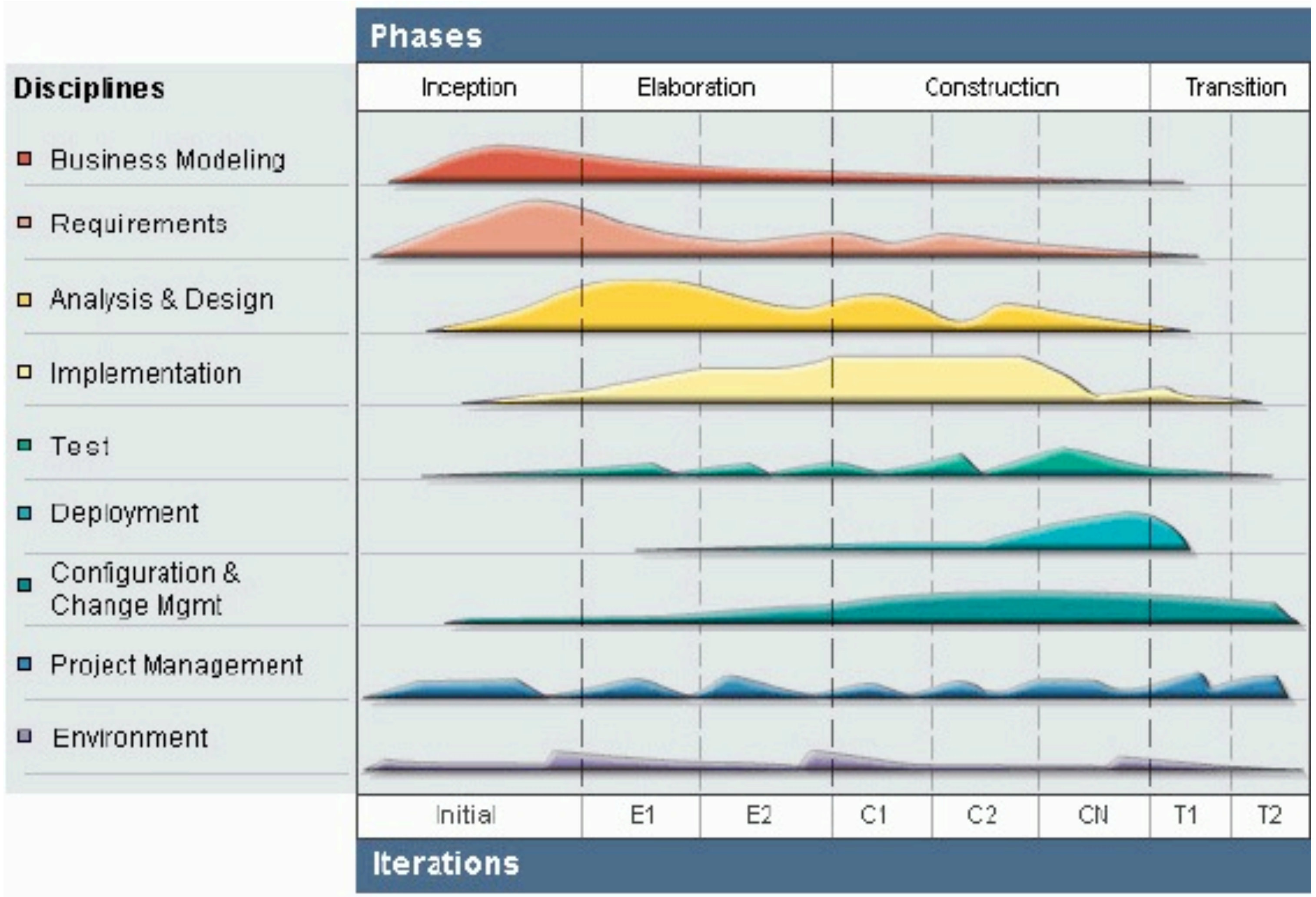
Object-Oriented Design

- Objects
  - An object is an entity that has a state and a defined set of operations that operate on that state.
  - Objects are created according to an object class definition.
- Design Process
  - Design the system architecture.
  - Identify objects in the system.
  - Describe the design using different object models.
  - Document the object interfaces.

Object-Oriented Design: Techniques

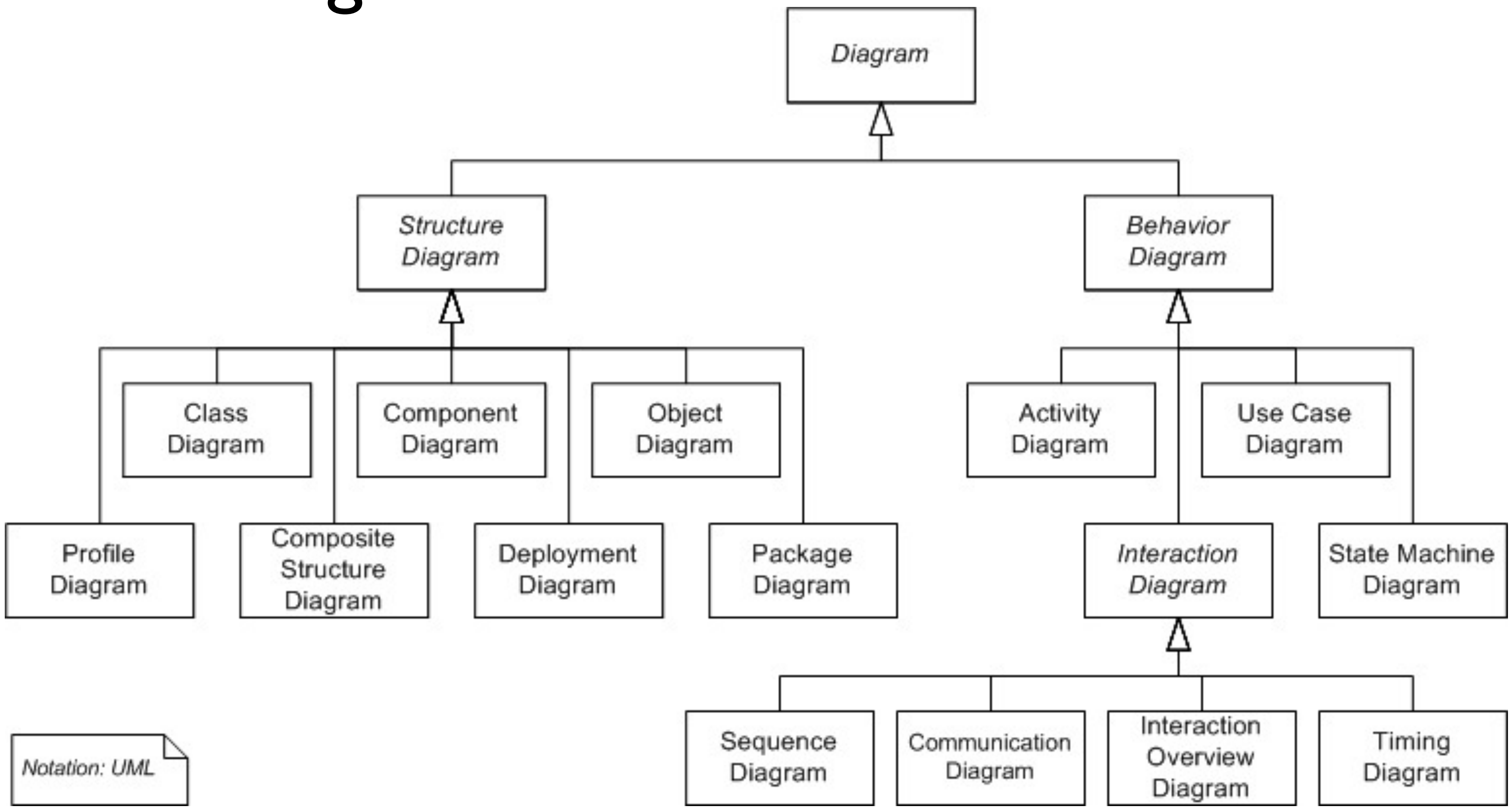
- Rational Unified Process (RUP)
  - An iterative software development process that is mostly used to guide object-oriented analysis and design.
- Unified Modeling Language (UML)
  - Provides a range of notations that can be used to document an object-oriented design.
- Design Patterns
  - Reuses solutions, rather than solves every problem from first principles.

Rational Unified Process (RUP)



- Four phases, nine workflows (activities).
- All of RUP workflows may be active at all phases of the process.
- Each phase and the whole set of phases are enacted in an iterative way.

UML Diagrams



Will be covered in the lecture of architecture modeling.

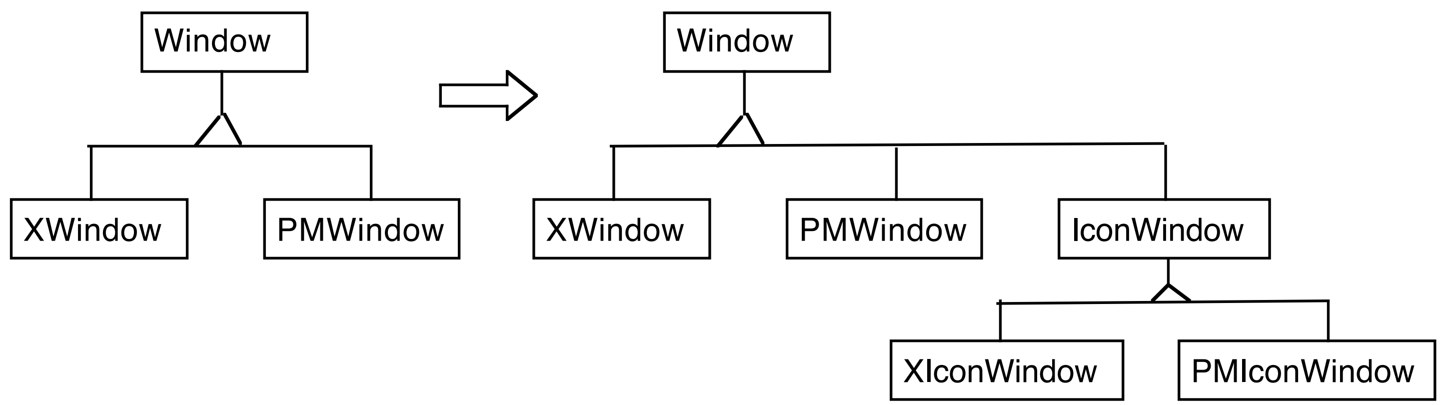
Design Patterns

- First codified by the Gang of Four in 1995
  - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
- Definition of Design Pattern
  - Descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.
- Essence of Design Pattern
  - Records recurring design in object-oriented systems.
  - Identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities.

Basic Principles of Design Patterns

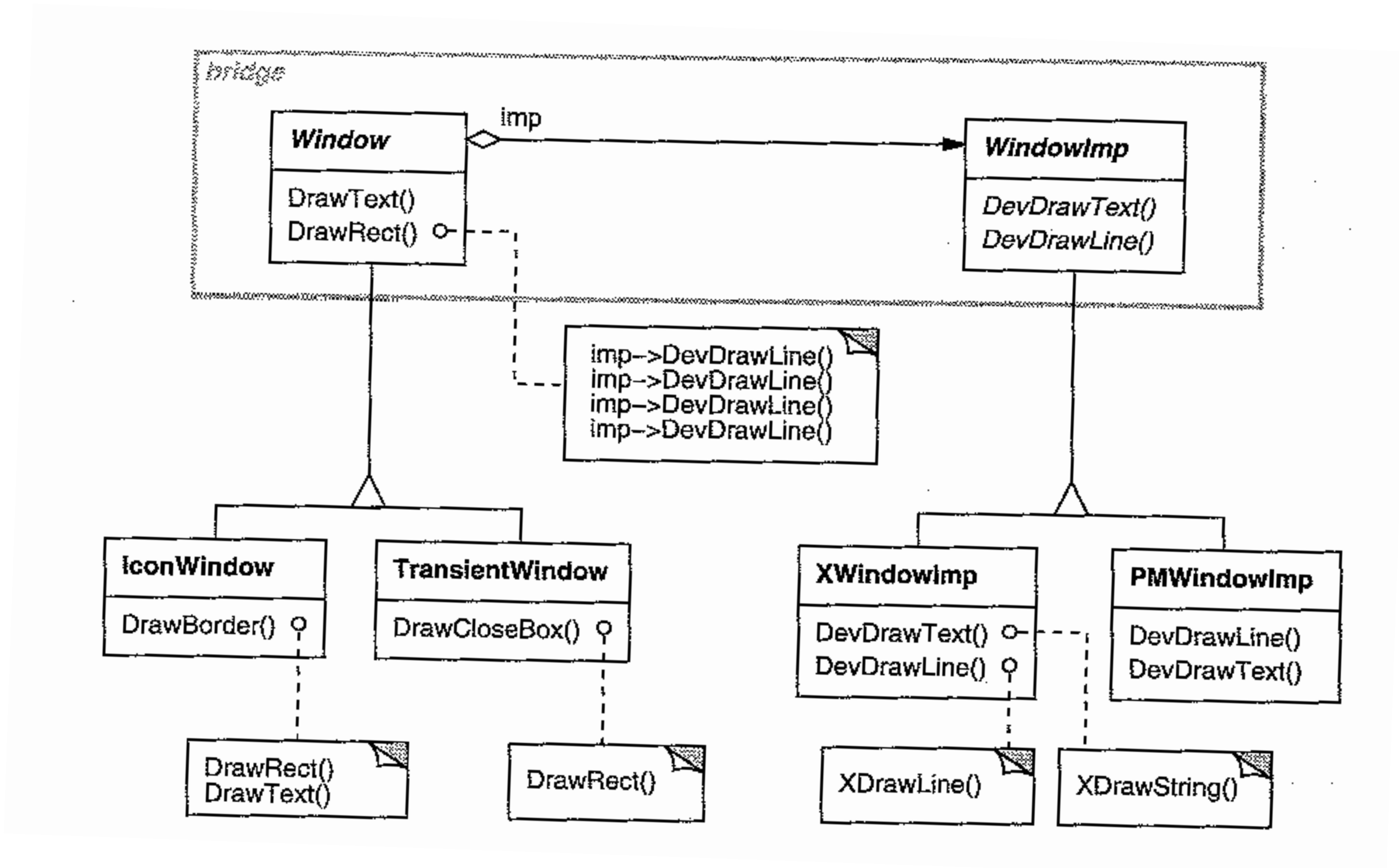
- Commit only to an interface defined by an abstract class
  - Program to an interface, not an implementation.
  - Do not declare variables to be instances of concrete classes.
- Favor object composition over class inheritance
  - Inheritance binds an implementation to the abstraction permanently.
  - Inheritance breaks encapsulation: subclass sees parent's implementations.
  - Inheritance only represents extensions along one dimension.

An example of design patterns



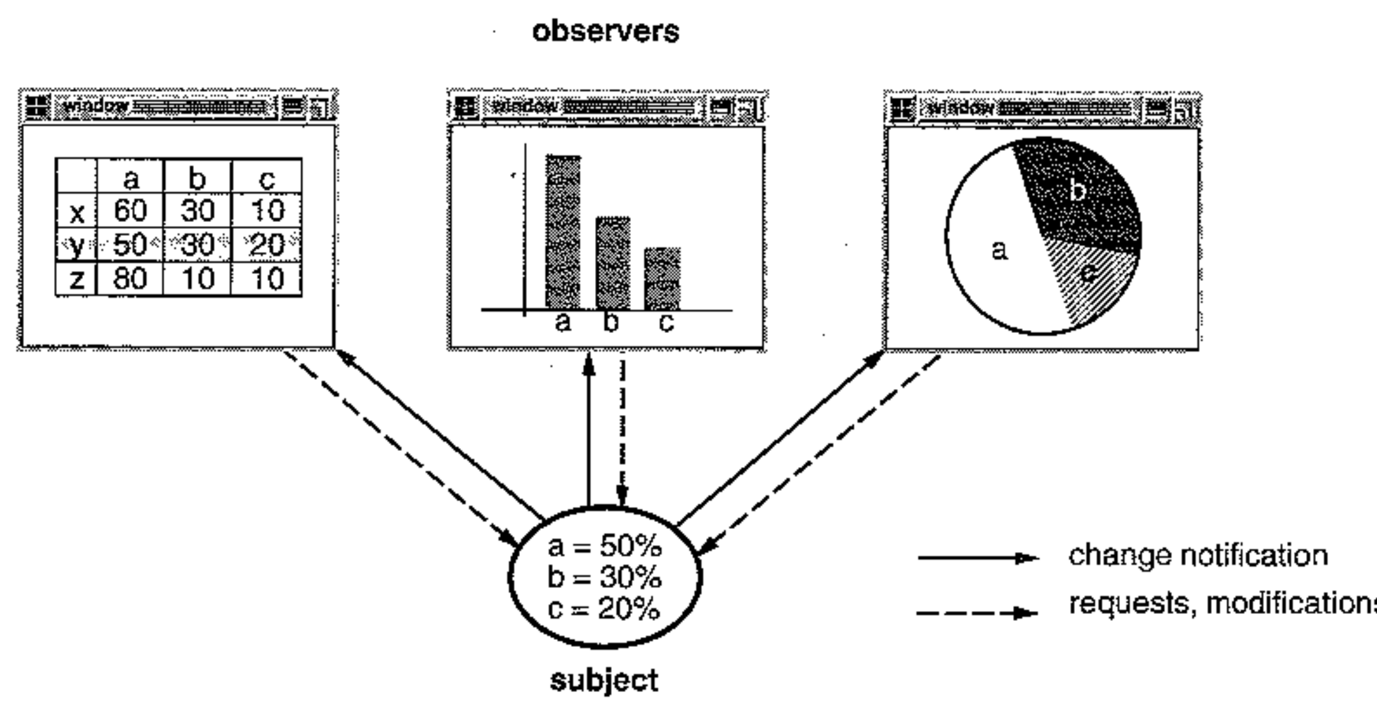
What is wrong with the above design?

The Bridge Design Pattern



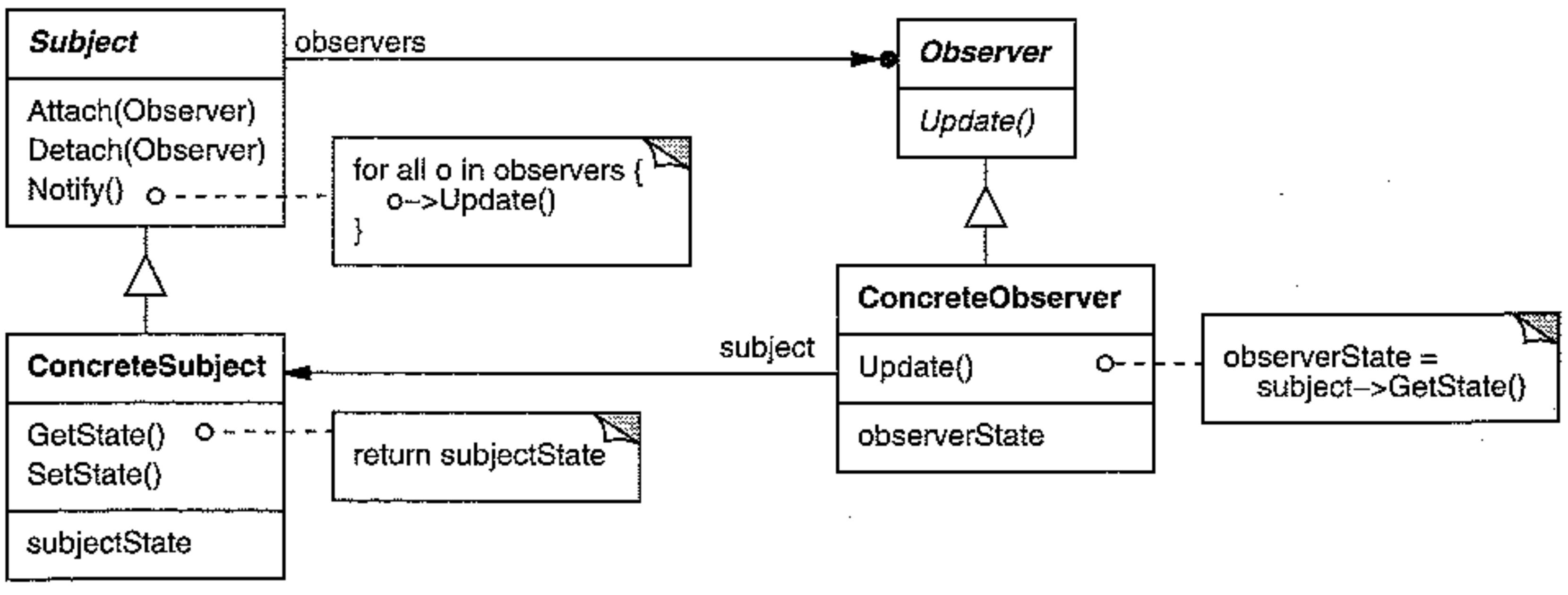
Decouples an abstraction from its implementation so that the two can vary independently.

Another example



A one-to-many dependency (publish-subscribe) between objects: when one object changes state, all its dependents are notified and updated automatically.

The Observer Pattern



- Subjects and observers are loosely coupled.
- Add observers without modifying the subject or other observers.

Object-Oriented Design: Benefits & Limitation

- Benefits
  - Easy to evolve software: changing the internal details of an object is unlikely to affect any other objects.
  - Reusability (really?)
  - More natural: it fits the way we view the world around us.
- Limitation
  - Essentially, object-oriented design decomposes a system along only one dimension – objects. However, we may need to decompose a system along some other dimensions, such as functionalities.