Applying UML & Patterns (3rd ed.)

Chapter 1

Introduction

Dr. Glenn L. Ray

School of Information Sciences

University of Pittsburgh

gray@sis.pitt.edu 412-624-9470

Course Design

- Best practices in SW development
  - Requirements management is paramount
    - Apply techniques learned in IS 2510
    - Maintain traceability throughout SWDLC
  - OOAD paradigm trumps SAD paradigm
    - Closer correspondence to problem domain
    - More robust architecture
    - Reusability
    - Maintainability

Course Design

- Best practices in SW development
  - Blend OO with iterative SW processes
    - Apply right technique at the right time
  - Prepare UML-compliant models
    - The standard OO modeling notation
  - Learn from other good designs
    - Examine 2 case studies in class
    - Apply techniques learned to SASID project

Course Design

- Additional important practices
  - Follow good OO design practices
    - Skillful assignment of responsibilities to objects
      - The ‘critical’ design skill
    - GRASP patterns guide responsibility assignment
  - Use Patterns
    - Codify reusable design idioms
    - Bolsters architectural integrity

Course Design

- Additional important practices
  - Map OOD artifacts to OO code
    - Maintains traceability from requirements to code
  - Write tests prior to writing code
    - Use cases drive testing
    - Write unit tests prior to writing code
    - Maintain library of reusable unit tests

Fig. 1.1

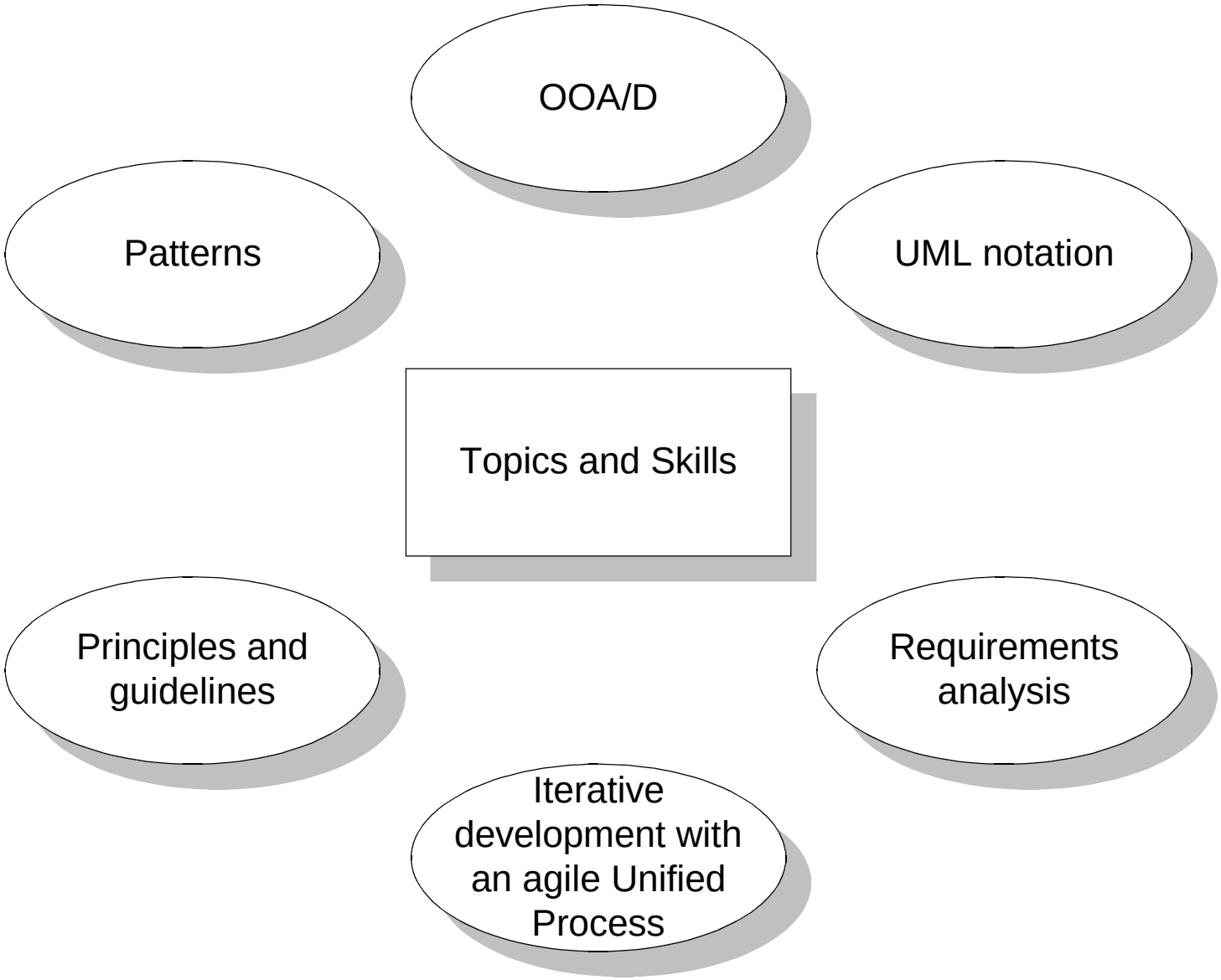


Fig. 1.2

Analysis concept

Design concept

domain concept

representation in an object-oriented programming language

Plane
tailNumber

visualization of domain concept

```
public class Plane
{
private String tailNumber;

public List getFlightHistory() {...}
}
```

Fig. 1.3

- Basic SW process
  - Write use cases
  - Prepare domain model
    - Conceptual class diagram
    - Depicts ‘static’ object relationships
  - Prepare interaction diagrams
    - Models execution of a use case scenario(s)
    - Depicts how objects collaborate over time
    - Prepare design class diagram(s)

Fig. 1.3

Domain model for a dice game

Fig. 1.4

Sequence diagram for Play Dice Game

(Where's the Player?)

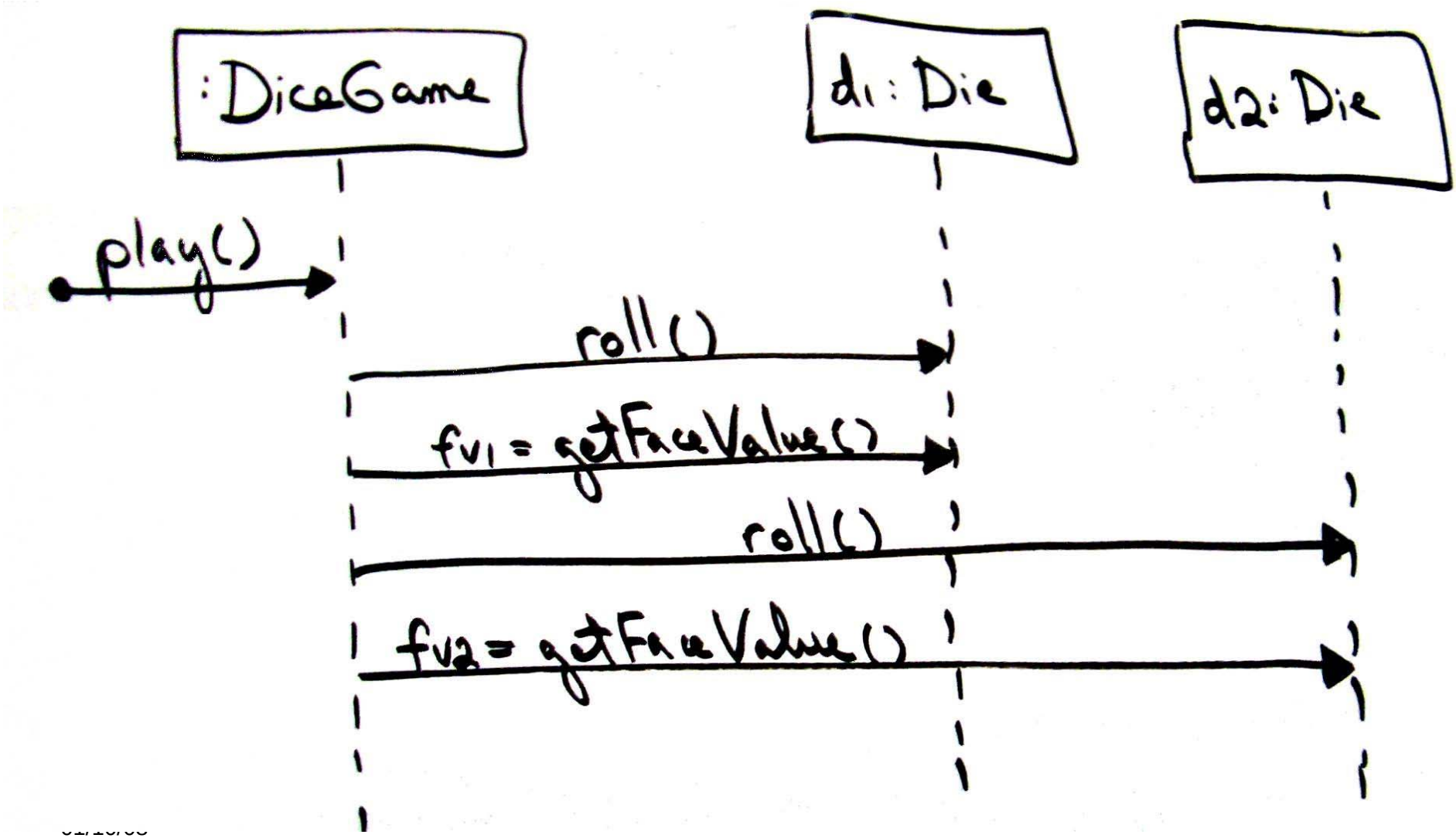


Fig. 1.5

Design class diagram for dice game

(How does it differ from domain model?)

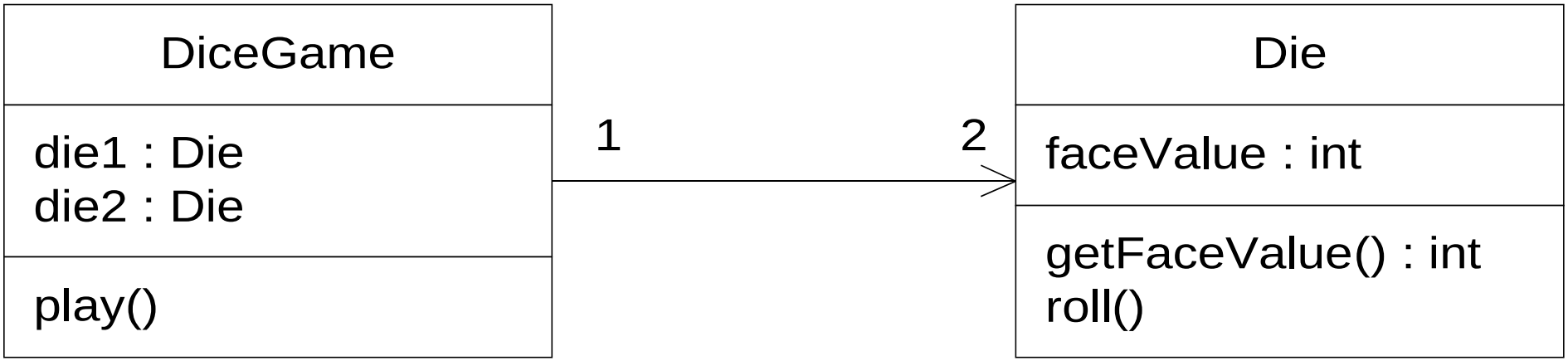


Fig. 1.6

- 3 ways to use UML
  - Sketch
    - Very quick, requires no fancy tools
    - Preferred by Agile modelers
    - Often good enough
  - Blueprint
    - Detailed model drawn by tool
    - Use to generate code
    - Reverse engineer from code

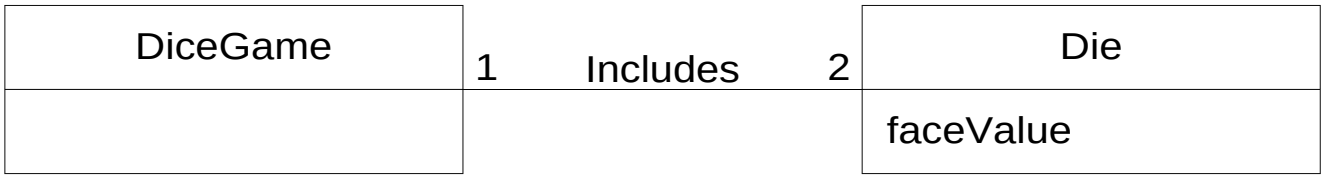
Fig. 1.6

- 3 ways to use UML
  - Executable programming language
    - Prepare complete suite of models for system
    - Compile models directly into executable code
    - Eliminates costly coding activity
    - Not ready for ‘primetime’

Fig. 1.6

3 perspectives of UML modeling

Conceptual class



Conceptual Perspective (domain model)

Raw UML class diagram notation used to visualize real-world concepts.

Specification or Implementation Perspective (design class diagram)

Raw UML class diagram notation used to visualize software elements.

SW or Implementation class

Implementation perspective uses

Language-specific syntax