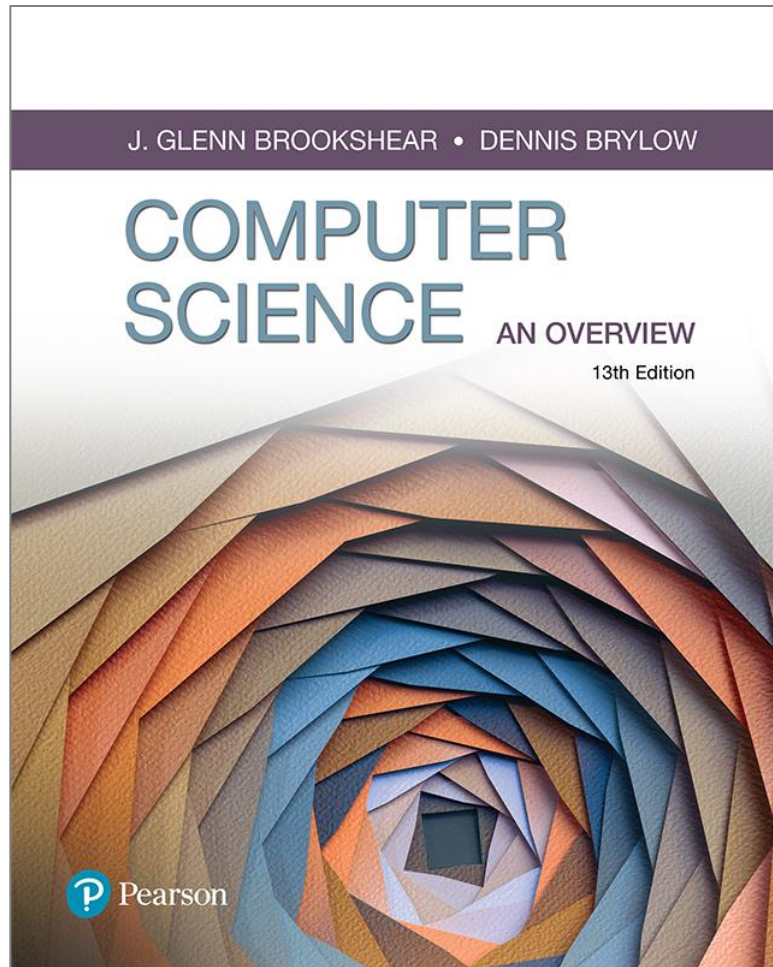


Computer Science An Overview

13th Edition



Chapter 7

Software Engineering

Chapter 7: Software Engineering

- 7.1 The Software Engineering Discipline
- 7.2 The Software Life Cycle
- 7.3 Software Engineering Methodologies
- 7.4 Modularity
- 7.5 Tools of the Trade
- 7.6 Quality Assurance
- 7.7 Documentation
- 7.8 The Human-Machine Interface
- 7.9 Software Ownership and Liability

7.1 The Software Engineering Discipline

- Distinct from other engineering fields
 - Lack of prefabricated components
 - Lack of metrics
- Practitioners versus Theoreticians
- Professional Organizations: ACM, IEEE, etc.
 - Codes of professional ethics
 - Standards

Computer Aided Software Engineering (CASE) tools

- Project planning
- Project management
- Documentation
- Prototyping and simulation
- Interface design
- Programming

7.2 The Software Life Cycle

- Effort put into the development of software can make a tremendous difference when modifications are required
- Unlike manufactured products, maintenance consists of correcting and updating
- Ongoing maintenance requires that someone can understand the program and its documentation

Figure 7.1 The software life cycle

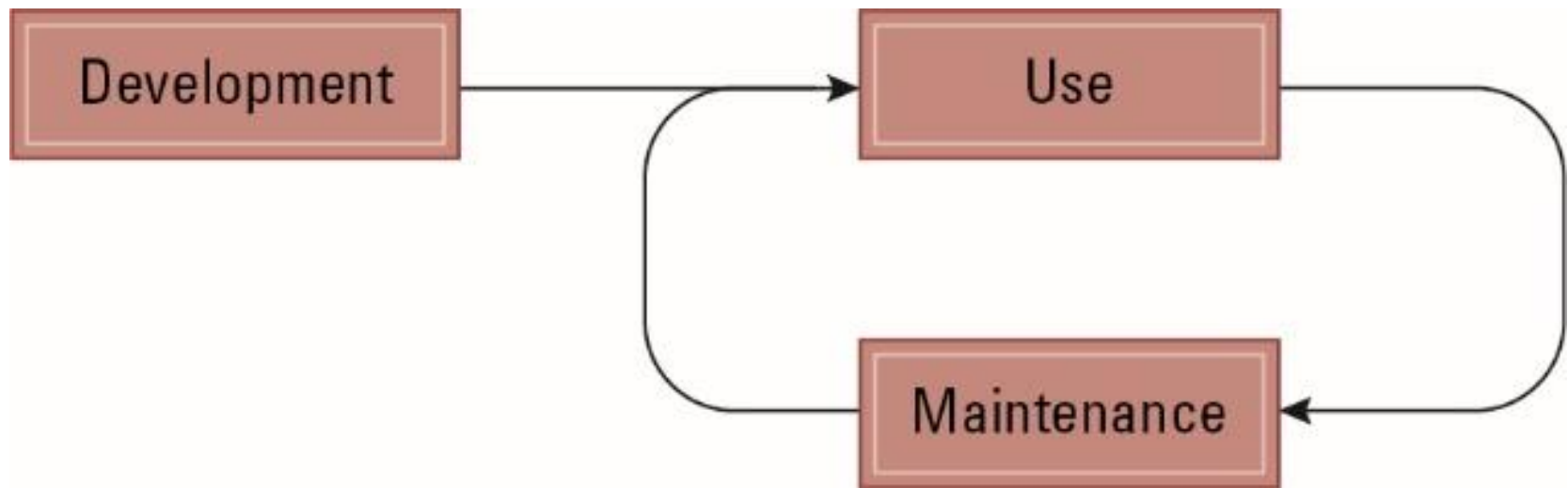
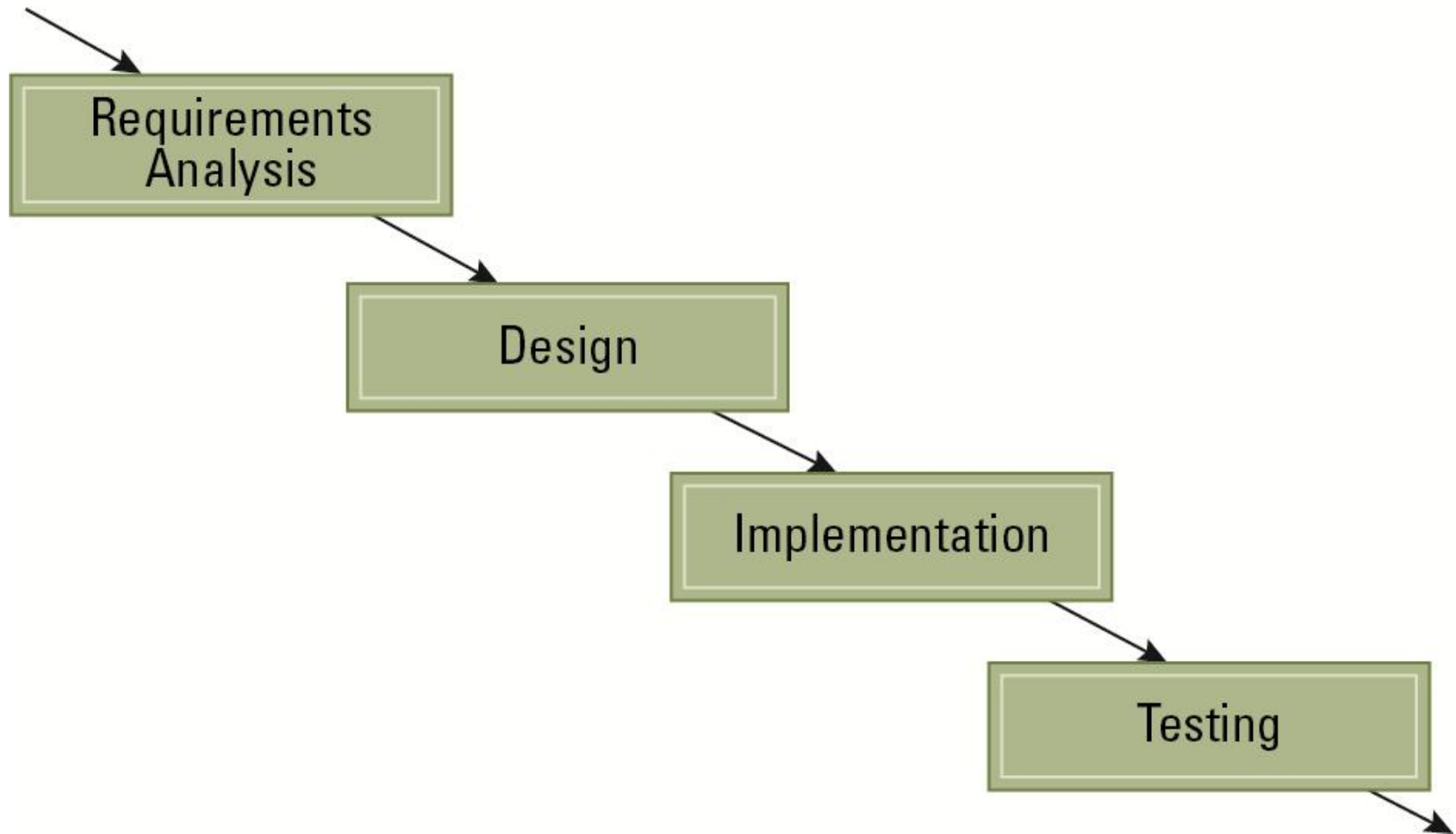


Figure 7.2 The development phase of the software life cycle



Requirement Analysis Stage

- Requirements
 - Application oriented
- Specifications
 - Technically oriented
- Software requirements document

Design Stage

- Methodologies and tools (discussed later)
- Human interface (psychology and ergonomics)

Implementation Stage

- Create system from design
 - Write programs
 - Create data files
 - Develop databases
- Role of “software analyst” versus “programmer”

Testing Stage

- Validation testing
 - Confirm that system meets specifications
- Defect testing
 - Find bugs

7.3 Software Engineering Methodologies

- Waterfall Model
- Incremental Model
 - Prototyping (Evolutionary vs. Throwaway)
- Open-source Development
- Agile Methods

7.4 Modularity

- Functions – Imperative paradigm
 - Structure charts
- Objects – Object-oriented paradigm
 - Collaboration diagrams
- Components – Component architecture

Figure 7.3 A simple structure chart

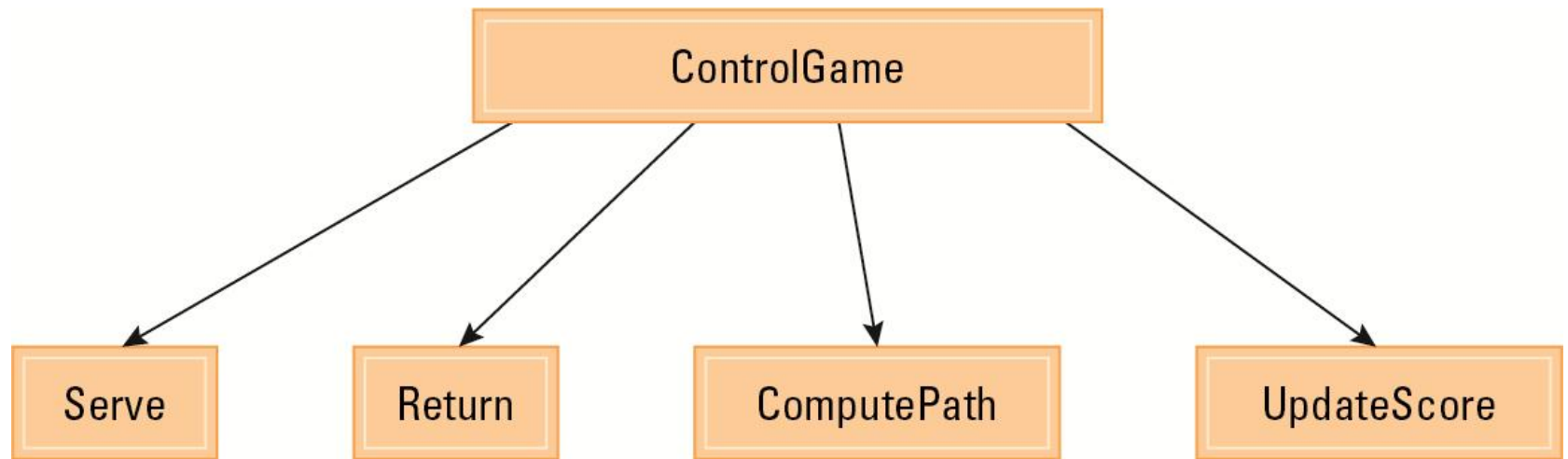


Figure 7.4 The structure of PlayerClass and its instances

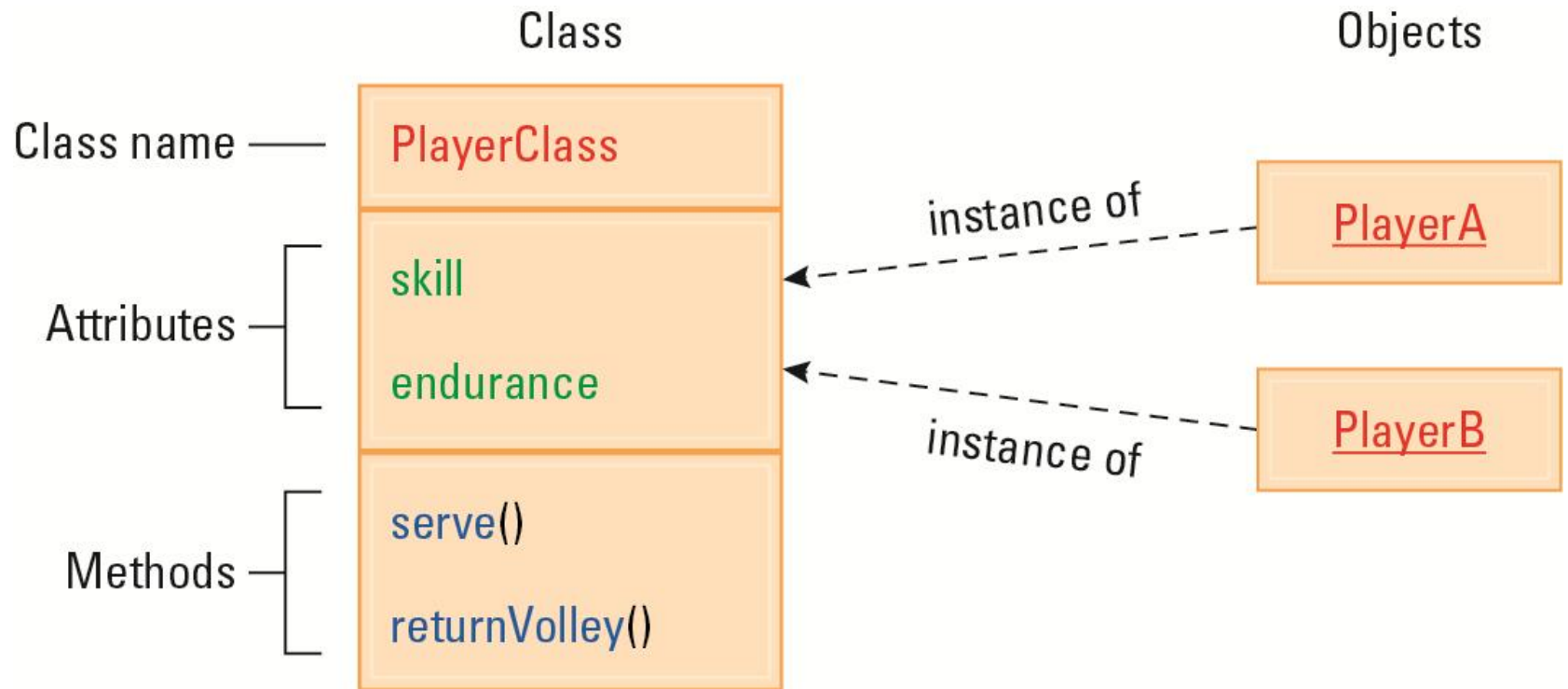


Figure 7.5 The interaction between objects resulting from PlayerA's serve

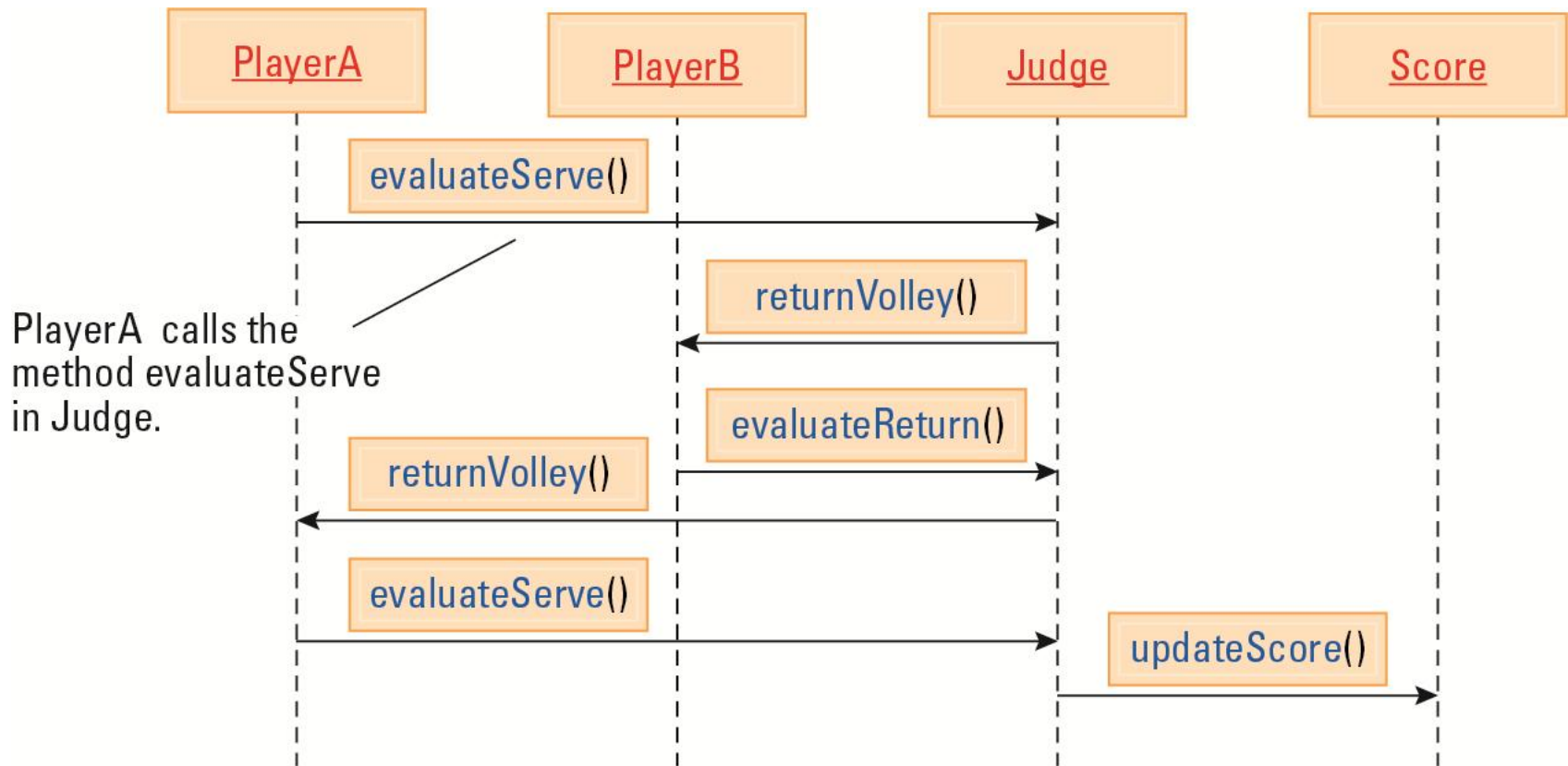
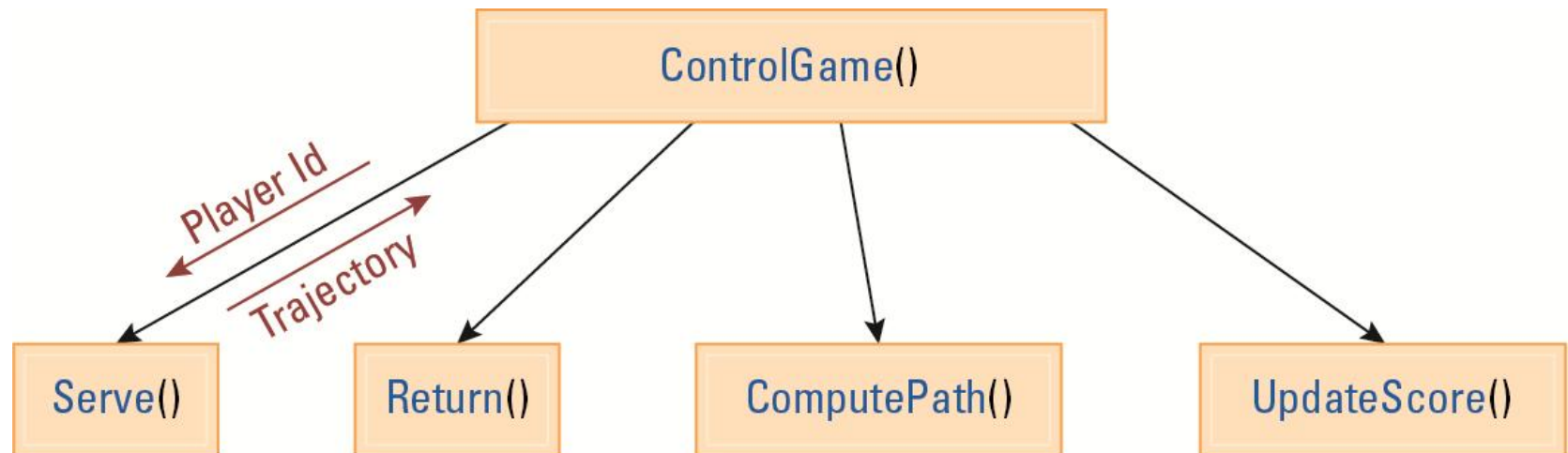


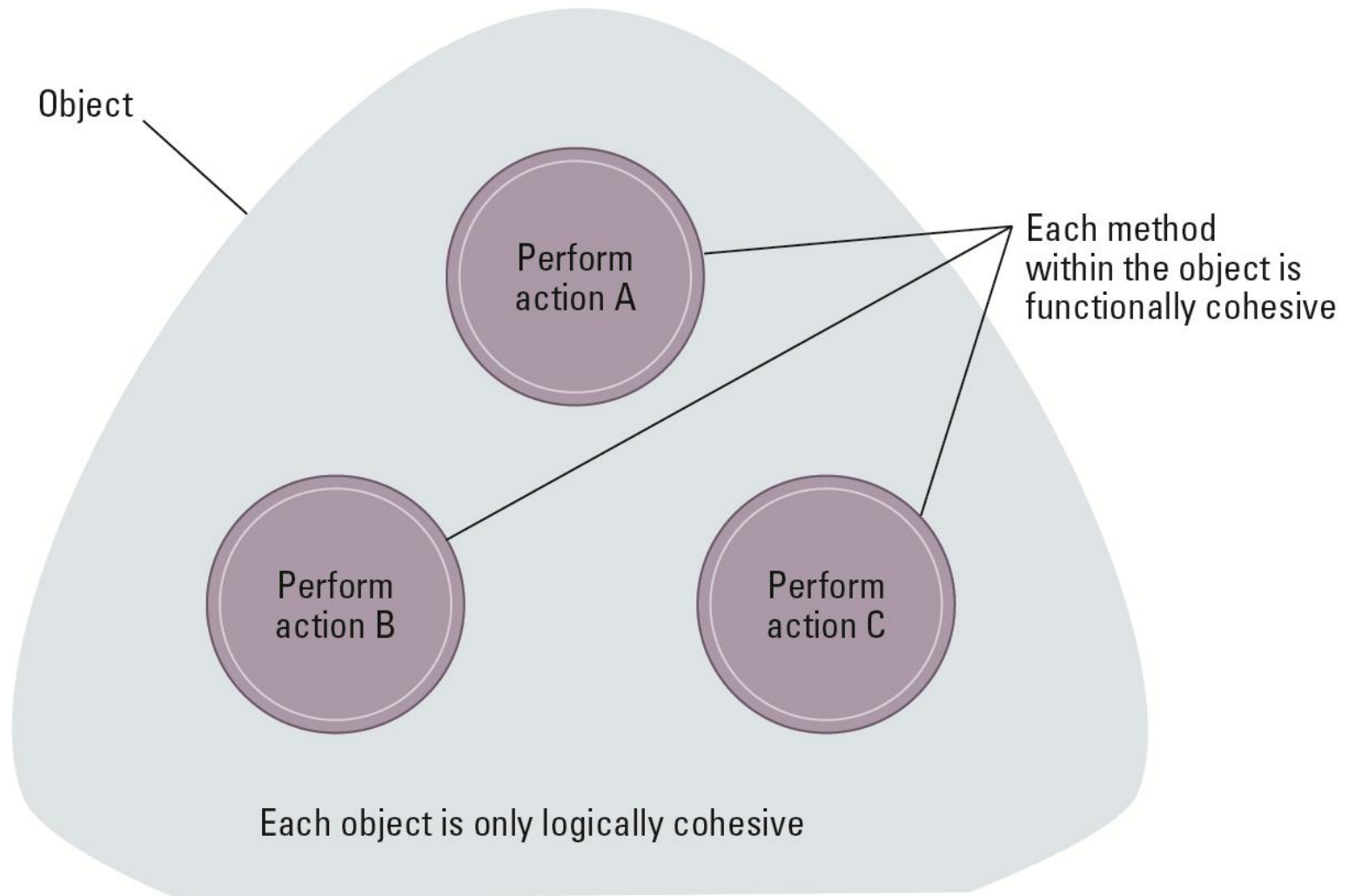
Figure 7.6 A structure chart including data coupling



Coupling versus Cohesion

- Coupling
 - Control coupling
 - Data coupling
- Cohesion
 - Logical cohesion
 - Functional cohesion

Figure 7.7 Logical and functional cohesion within an object



7.5 Tools of the Trade

- Data Flow Diagram
- Entity-Relationship Diagram
 - One-to-one relation
 - One-to-many relation
 - Many-to-many relation
- Data Dictionary

Figure 7.8 A simple dataflow diagram

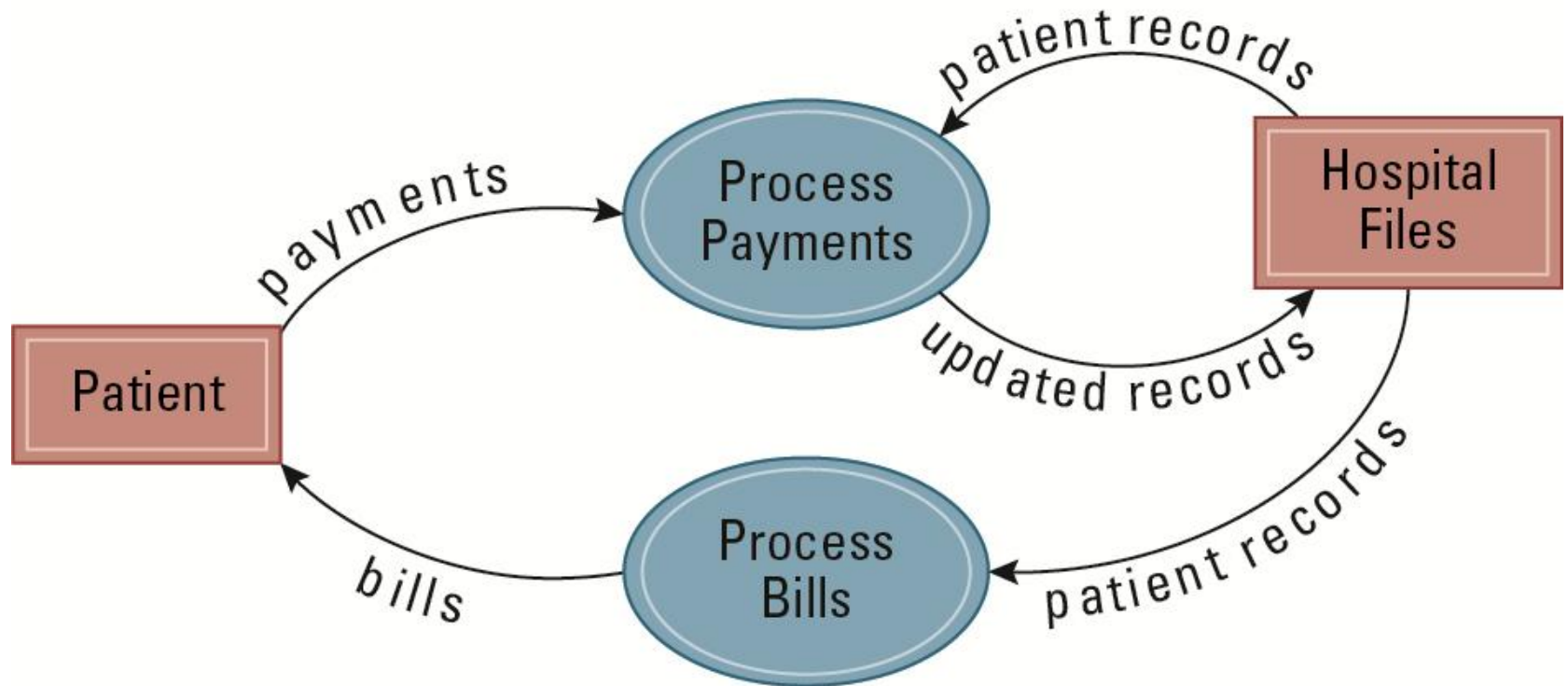


Figure 7.9 A simple use case diagram

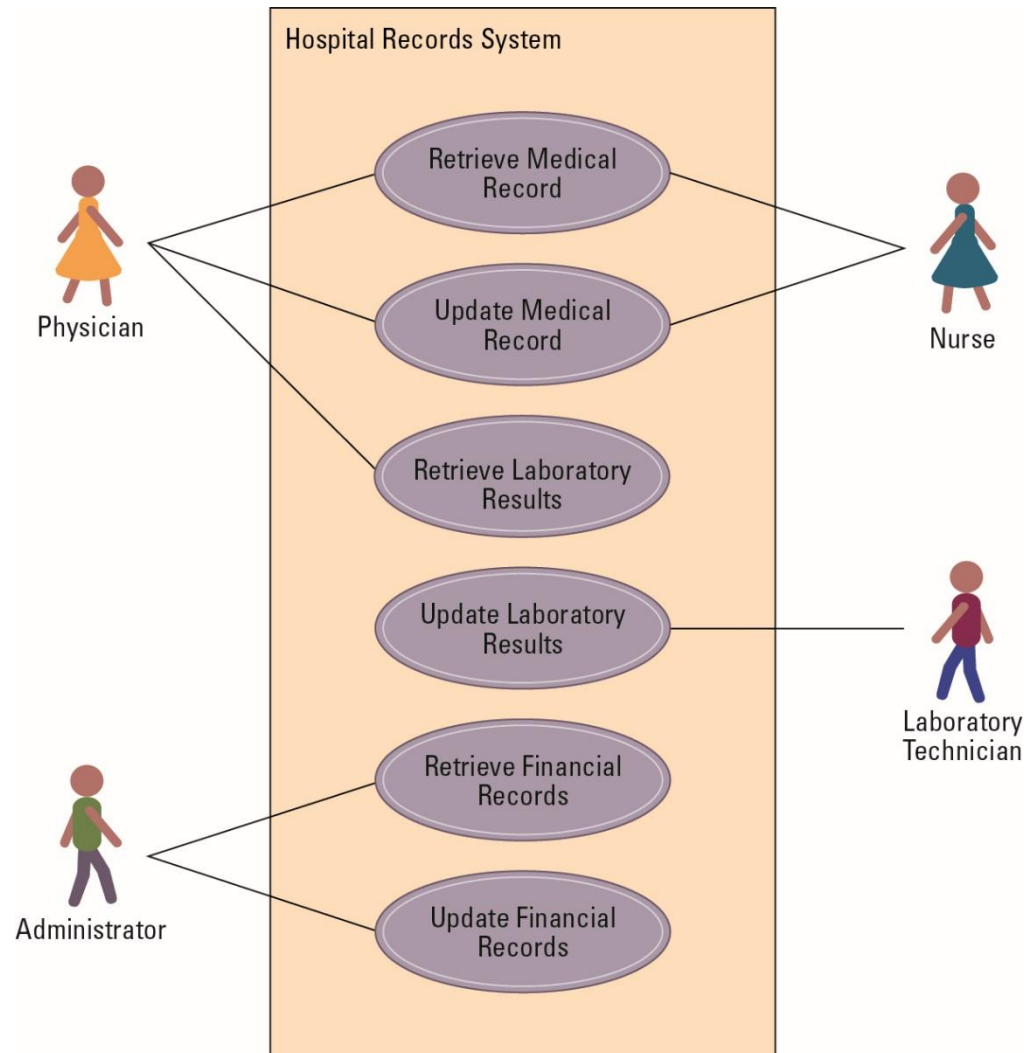


Figure 7.10 A simple class diagram



Unified Modeling Language

- Use Case Diagram
 - Use cases
 - Actors
- Class Diagram

Figure 7.11 One-to-one, one-to-many, and many-to-many relationships between entities of types X and Y

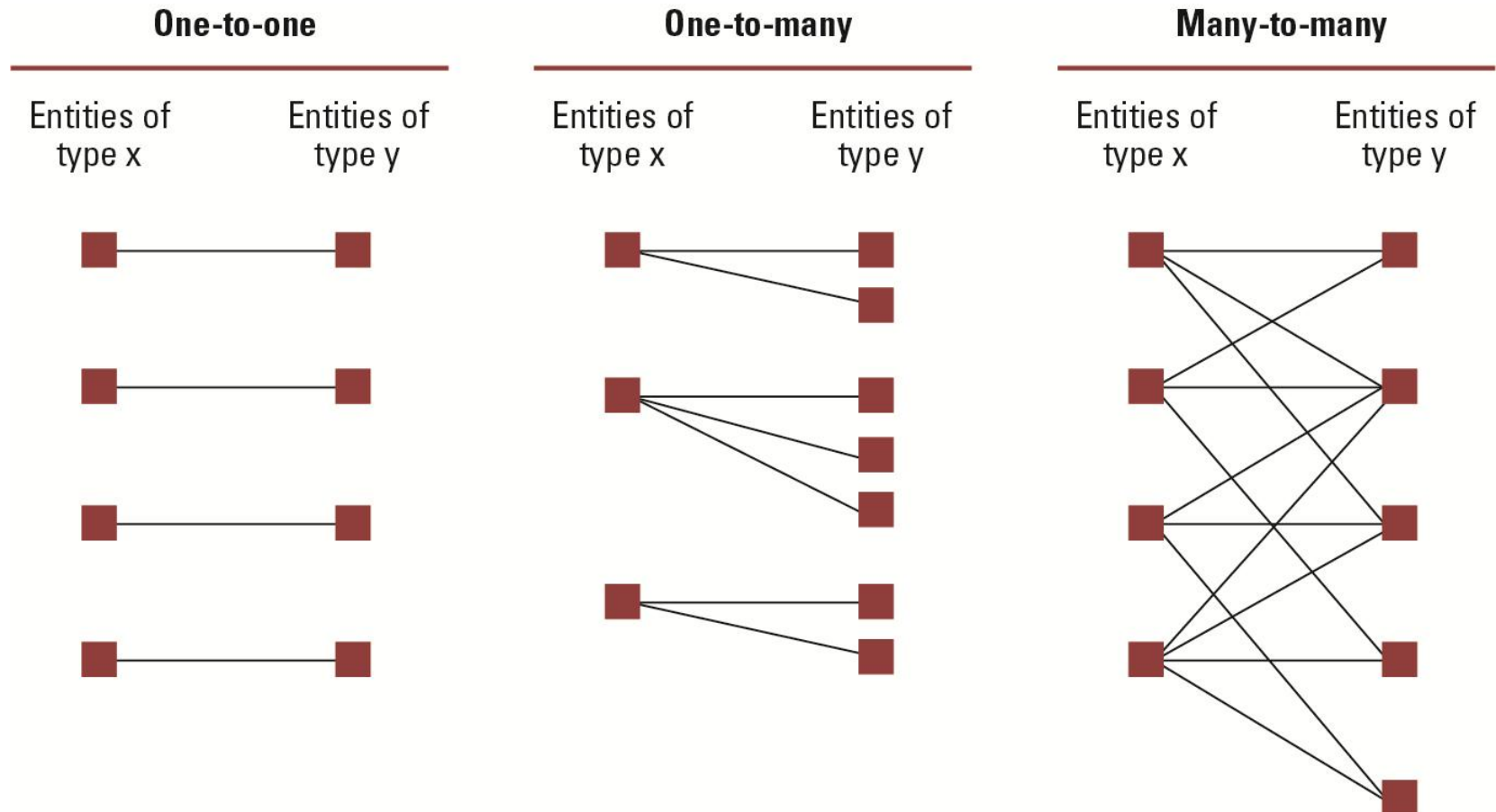


Figure 7.12 A class diagram depicting generalizations

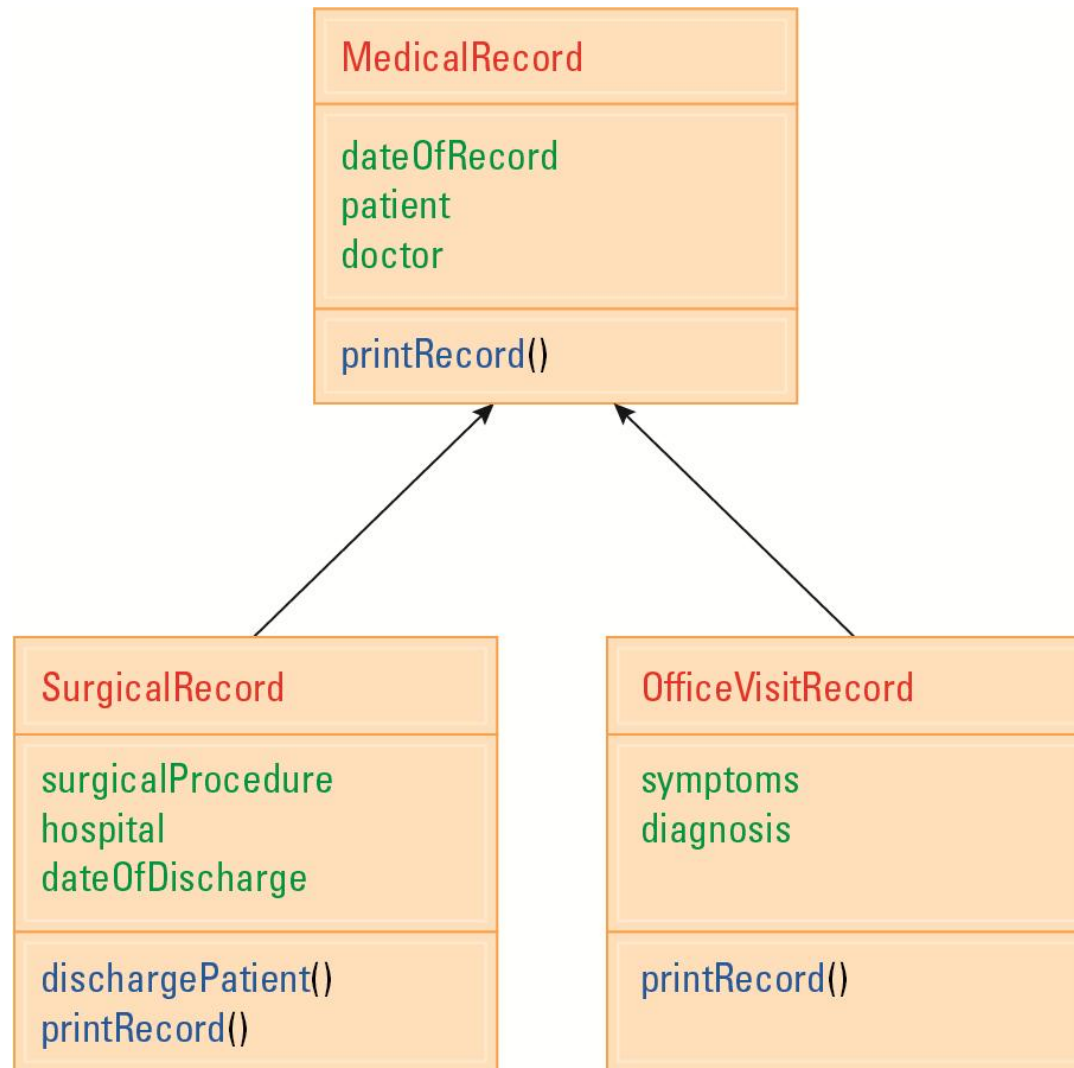
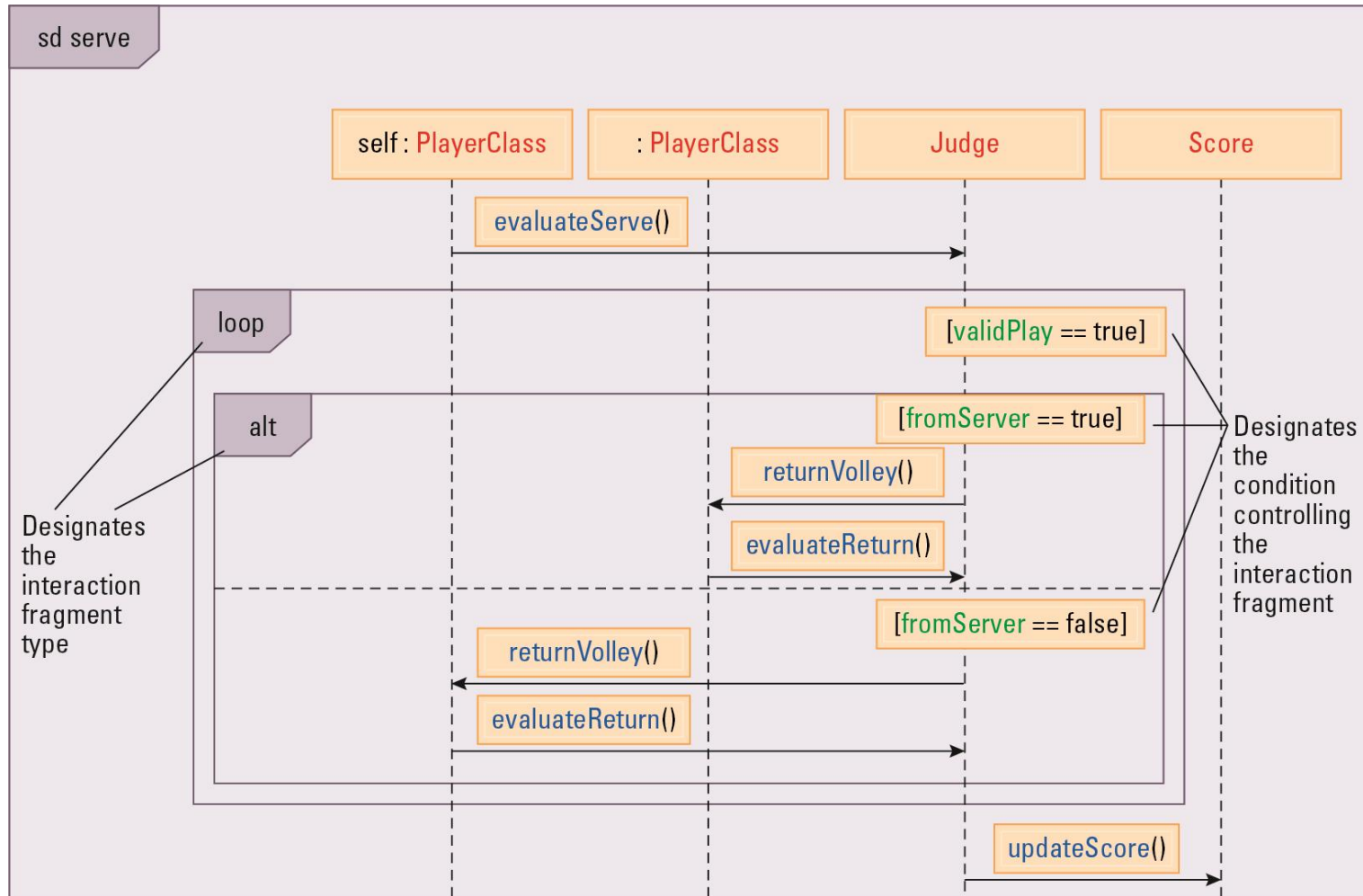


Figure 7.13 A sequence diagram depicting a generic volley



Design Patterns

- Well designed “templates” for solving recurring problems
- Examples:
 - Adapter pattern: Used to adapter a module’s interface to current needs
 - Decorator pattern: Used to control the complexity involved when many different combinations of the same activities are required
- Inspired by the work of Christopher Alexander in architecture

Simulating a System

- Structured Walkthroughs
- Each member of the design team is given a CRC card (Class-responsibility-collaboration)
- “Theatrical” experiment – role playing

7.6 Quality Assurance

- Glass-box testing
 - Pareto principle
 - Basis path testing
- Black-box testing
 - Boundary value analysis
 - Redundancy testing
 - Beta testing

7.7 Documentation

- User Documentation
 - Printed book for all customers
 - On-line help modules
- System Documentation
 - Source code
 - Design documents
- Technical Documentation
 - For installing, customizing, updating, etc.

7.8 The Human-Machine Interface

- Idea that a software system is a tool which is designed for the convenience of the human using it
- Ergonomics – physical abilities of humans
- Cognetics – mental abilities of humans
 - Habits
 - Attention

7.9 Software Ownership and Liability

- Copyright
 - Allow a product to be released while retaining ownership of intellectual property
 - Asserted in all works:
 - Specifications
 - Source code
 - Final product

Intellectual Property

- Software License
 - A legal agreement that grants the user certain permissions without transferring ownership
- Patents
 - Must demonstrate that it is new, usable, and not obvious to others with similar backgrounds
 - Process is expensive and time-consuming

Copyright



This work is protected by United States copyright laws and is provided solely for the use of instructions in teaching their courses and assessing student learning. dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.