Chapter 7
Software Engineering

The Software Engineering Discipline
- Distinct from other engineering fields
  - Prefabricated components
  - Metrics
- Practitioners versus Theoreticians
- Professional Organizations: ACM, IEEE, etc.
  - Codes of professional ethics
  - Standards

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 Testing
- 7.7 Documentation
- 7.8 Software Ownership and Liability

Computer Aided Software Engineering (CASE) tools
- Project planning
- Project management
- Documentation
- Prototyping and simulation
- Interface design
- Programming
Figure 7.1 The software life cycle

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

Analysis Stage

Figure 7.2 The development phase of the software life cycle

- 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”

Software Engineering Methodologies

- Waterfall Model
- Incremental Model
  - Prototyping (Evolutionary vs. Throwaway)
- Open-source Development
- Extreme Programming

Testing Stage

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

Modularity

- Procedures -- Imperative paradigm
  - Structure charts
- Objects -- Object-oriented paradigm
  - Collaboration diagrams
- Components -- Component architecture
Figure 7.3 A simple structure chart

ControlGame

Serve Return ComputePath UpdateScore

Figure 7.4 The structure of PlayerClass and its instances

Class
PlayerClass Skill Endurance

Attributes

Methods
serve return

Objects
PlayerA
PlayerB

instance of

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

PlayerA calls the method evaluateServe in Judge.

PlayerA

Judge

Score

evaluateServe
returnVolley

evaluateReturn
updateScore

Figure 7.6 A structure chart including data coupling

ControlGame

Serve Return ComputePath UpdateScore

Player Id

Trajectory
Coupling versus Cohesion

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

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 7.7 Logical and functional cohesion within an object

Figure 7.8 A simple dataflow 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

- **One-to-one**
  - Entities of type x
  - Entities of type y

- **One-to-many**
  - Entities of type x
  - Entities of type y

- **Many-to-many**
  - Entities of type x
  - Entities of type y

---

**Figure 7.9** A simple use case diagram

**Figure 7.10** A simple class diagram
Structured Walkthroughs

- “Theatrical” experiment
- Class-responsibility-collaboration cards

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
Software Testing Strategies

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

Software Ownership

• Copyright
  – The “substantial similarity” test
  – Filtration criteria: what is not copyrightable
    • Features covered by standards
    • Characteristics dictated by software purpose
    • Components in the public domain
  – The “look and feel” argument

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.

Software Ownership (continued)

• Patents
  – “Natural laws” are traditionally not patentable
• Trade secrets
  – Non-disclosure agreements are legally enforceable