Chapter 3
Chapter 3 Service-Oriented Computing and SOA
Lecture Note


1. Design Fundamentals
   • What is Analysis and Design
   • The Meaning of Design
   • Design Related Terms
     o Design Characteristics
     o Design Principle
     o Design Paradigm
     o Design Pattern
     o Design Pattern Language
     o Design Standard
     o Best Practice
   • A Fundamental Design Framework

What is Analysis and Design? [1]
   • “Analysis emphasizes an investigation of problem and requirements, rather than a solution. …”
   • “During object-oriented analysis, there is an emphasis on findings and describing the objects—or concepts—in the problem domain. ..”
   • “Design emphasizes a conceptual solution that fulfills the requirements, rather than its implementation. ..”
   • “During object-oriented design, there is an emphasis on defining software objects and how they collaborate to fulfills the requirements…”

How about Service-Oriented Analysis and Design?

The Meaning of Design [2]
“In every engineering discipline, design encompasses the disciplined approach we use to invent a solution for some problem, thus providing a path from requirements to implementation. In the context of software engineering, Mostow suggests that the purpose of design is to construct a system that:
   • Satisfies a given (perhaps informal) functional specification
   • Conforms to limitations of the target medium
   • Meets implicit or explicit requirements on performance and resource usage
   • Satisfies implicit or explicit design criteria on the form of the artifact
   • Satisfies restrictions on the design process itself, such as its length or cost, or the tools available for doing the design” [3]
How about the meaning of Service-Oriented Design?
Design Related Terms [4]

- Design Characteristics
- Design Principle
- Design Paradigm
- Design Pattern
- Design Pattern Language
- Design Standard
- Best Practice

Design Characteristics [4]

- A characteristic of something is simply an attribute or quality.
- The desired design characteristic: a specific attribute or quality of a body of solution logic that we document in a design specification and plan to realize in development
- Example: Figure 3.1
  - Design A:
  - Design B
  - Design C

![Diagram of Design A, B, and C](image)

- componentized
- tightly coupled
- shared database
- moderately stateful

- componentized
- tightly coupled
- dedicated database
- highly stateful

- componentized and distributed
- loosely coupled
- targeted reuse
- dedicated database
- minimally stateful (state deferral via shared external state database)
Figure 3.1 In this simple example, three distinct application designs (A, B, C) are established, each with its own list of design characteristics. We will continue to reference these applications in the upcoming sections. (Note that the small squares represent units of solution logic, solid arrows represent reuse or shared access, and dashed arrows represent state data transfer.)

- **Design Characteristics of Service-Orientation**
  - Common design characteristic
    - => Spread consistency
    - => Increase commonality increased degree of consistency, making different kind of solution logic more alike
  - Predictable design
    - => Predictable behavior
  - Reliability

**Design Principle [4]**
- A principle is a generalized, accepted industry practice
- Design Principle
  - Represents a highly recommended guideline for shaping solution logic in a certain way and with certain goals in mind
- Eight Design Principles
- Example – Figure 3.2
  - Design Principle
    - Increases the amount of common design characteristics
    - Increases the level of decoupling or loose coupling
Design Paradigm

- A design paradigm within the context of business automation is generally considered a governing approach to designing solution logic.
- It consists of a set of complementary rules or principles that collectively define the overarching approach represented by the paradigm.
- The service-oriented design paradigm
  - Apply a right balance of loose coupling and reuse to distributed solution logic
- Example – Figure 3-3
Figure 3.3 Because a design paradigm represents a collection of design principles, it further increases the degree of commonality across different bodies of solution logic. In the example, the amount of reuse in A and B has increased.
Design Pattern
• A design pattern systematically names, explains, and evaluates an important and recurring design. Design patterns make it easier to reuse successful designs and architecture. [5]
• A design pattern describes a common problem and provides a corresponding solution (Figure 3.4) [4]

"Problem: 
Reusable solution logic that relies on a shared database executes with inconsistent response times."

"Solution: 
If solution logic is being reused, it should have a dedicated database to maximize autonomy."

Figure 3.4 Patterns provide recommended solutions for common design problems. In this simplified example, a pattern suggests we reduce external access to a database to increase application autonomy.

• Design pattern (Computer Science), http://en.wikipedia.org/wiki/Design_pattern_(computer_science) : “In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve problems that can be used in many different situations. …”
• A design pattern has four essential elements [3]:
  o The pattern name is a handle we can use to describe a design problem, its solution, and consequences in a word or two.
The problem describes when to apply the pattern.
The solution describes the elements that make up the design, their relationships, responsibilities, and collaborations.
The consequences are the results and trade-offs of applying the pattern.

- SOA Patterns, by Thomas Erl, http://www.soapatterns.org/
- SOA Design Patterns, http://zimmermann-software.de/soa-templates-project/patterns.html

Design Pattern Language
- Comprises of a chain of related design patterns that establish a configurable sequence in which the patterns can be applied
- Example - Figure 3.5

Figure 3.5 A sequence of related design patterns formalize the primary decision points of a design paradigm. In this example, the logic in application design B is decomposed as a result of one pattern, and then further decomposed as a result of another. Subsequent fundamental patterns continue to shape the logic.
Design Standards [4]

- Design standards are (usually mandatory) design conventions customized to consistently pre-determine solution design characteristics in support of organizational goals and optimized for specific enterprise environments.
- Example – Figure 3.6

"Due to specific security and privacy requirements, state data cannot be shared in a separate database."

Figure 3.6 In this case, a design standard requires that C’s original design be altered to remove access to a shared, external state database.
Best Practice

- A best practice is generally considered a technique or approach to solving or preventing certain problems
- Example – Figure 3.7

"Reusable solution logic should be owned and maintained by a separate custodian to ensure it is evolved properly to accommodate long-term enterprise requirements."

Figure 3.7 Best practices provide guidance in the form of general "lessons learned." In the example, it is suggested that the on-going maintenance of reusable solution logic units from all applications fall under a single custodian.
A Fundamental Design Framework

Figure 3.8 Fundamental design terms establish a basic taxonomy used throughout the upcoming chapters. This diagram hints at how some parts of a basic design framework can relate to each other.
Design patterns provide additional intelligence that can enrich a design framework with a collection of proven solutions to common problems.

**Figure 3.9** Design patterns provide additional intelligence that can enrich a design framework with a collection of proven solutions to common problems.
Figure 3.10 The purpose of applying a design paradigm is the achievement of certain goals. It is important to emphasize how design standards, design patterns, and best practices can all support the successful application of a design paradigm and, as a result, the attainment of its goals.
References
[5] *Design Patterns: Elements of Reusable Object-Oriented Software*, by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, from Addison-Wesley, 1995.