Basic Design Principles

- ***The Open-Closed Principle (OCP)***: open for extension but closed for modification
- ***The Liskov Substitution Principle (LSP)***: subclasses should be substitutable for their base classes
- ***Dependency Inversion Principle (DIP)***: Depend of abstractions. Do not depend on concretions
- ***The Interface Segregation Principle (ISP)***: Many client-specific interfaces are better than one general purpose interface

Packaging Principles

- ***The Release Reuse Equivalent Principle (REP)***: The granule of reuse is the granule of release
- ***The Common Closure Principle (CCP)***: Classes that change together belong together
- ***The Common Reuse Principle (CRP)***: Classes that aren't reused together should not be grouped together

Component-Level Design Guidelines

- Components
- Interfaces
- Dependencies and inheritance

Cohesion

- Functional
- Layer
- Communicational
- Sequential
- Procedural
- Temporal
- Utility

Coupling

- Content
- Common
- Control
- Stamp
- Data
- Routine call
- Type use
- Inclusion or import
- External

Data: If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent of each other and communicate through data. Module communications don't contain tramp data. Example-customer billing system.

Component-Level Design 1

1. Identify the application domain classes
2. Identify the infrastructure domain classes
3. Refine the design classes
   a. Specify the details of the collaborations
   b. Specify the interfaces of each component
   c. Refine the attributes, data types, and data structures
   d. Describe processing flow

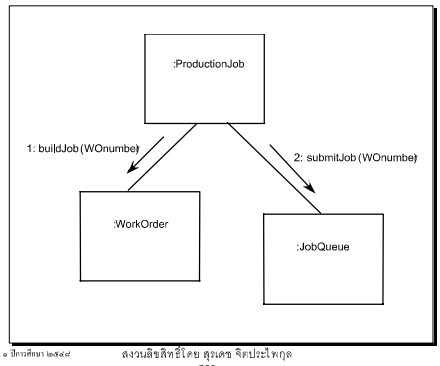
Component-Level Design 2

4. Specify persistent data sources and related classes
5. Refine the behavioral of a class or component
6. Add the implementation detail
7. Refactor

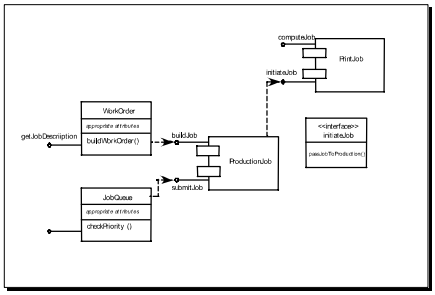
Component Level Design-I

- Step 1. Identify all design classes that correspond to the problem domain.
- Step 2. Identify all design classes that correspond to the infrastructure domain.
- Step 3. Elaborate all design classes that are not acquired as reusable components.
- Step 3a. Specify message details when classes or component collaborate.
- Step 3b. Identify appropriate interfaces for each component.

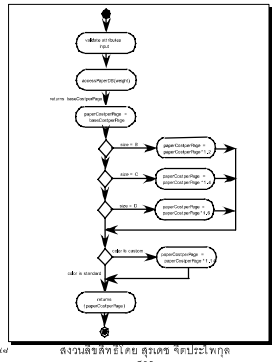
Collaboration Diagram



Refactoring



Activity Diagram



Statechart