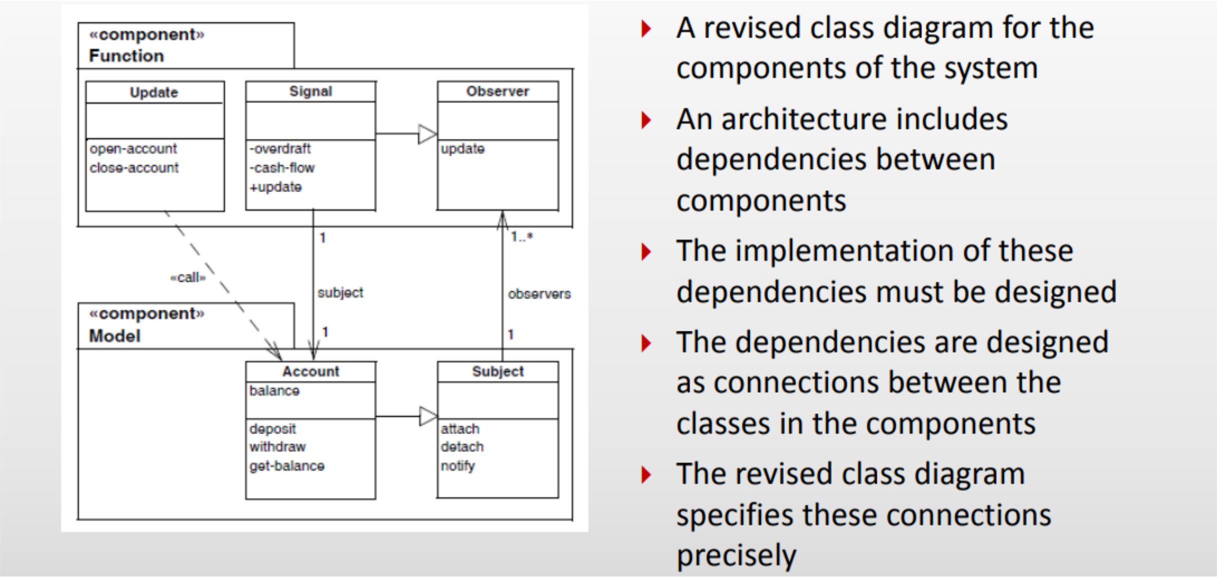
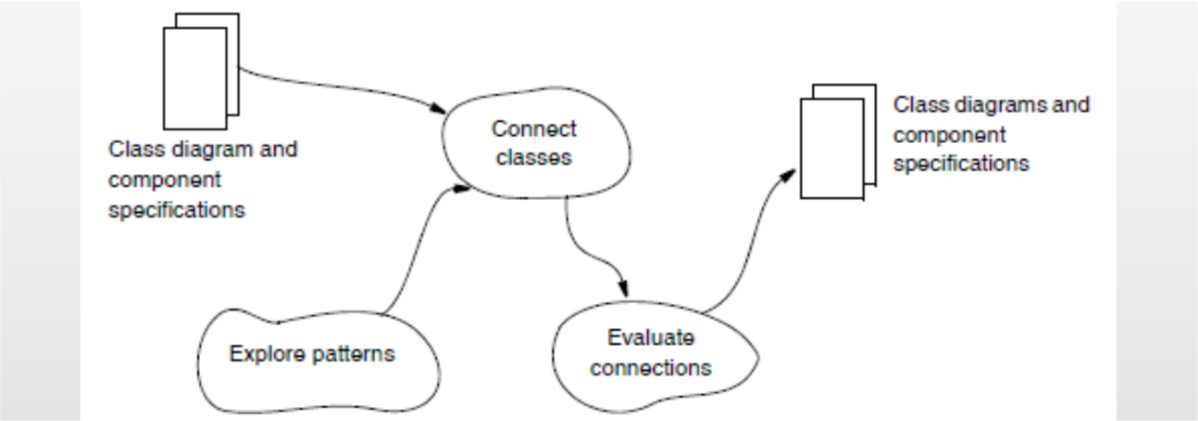
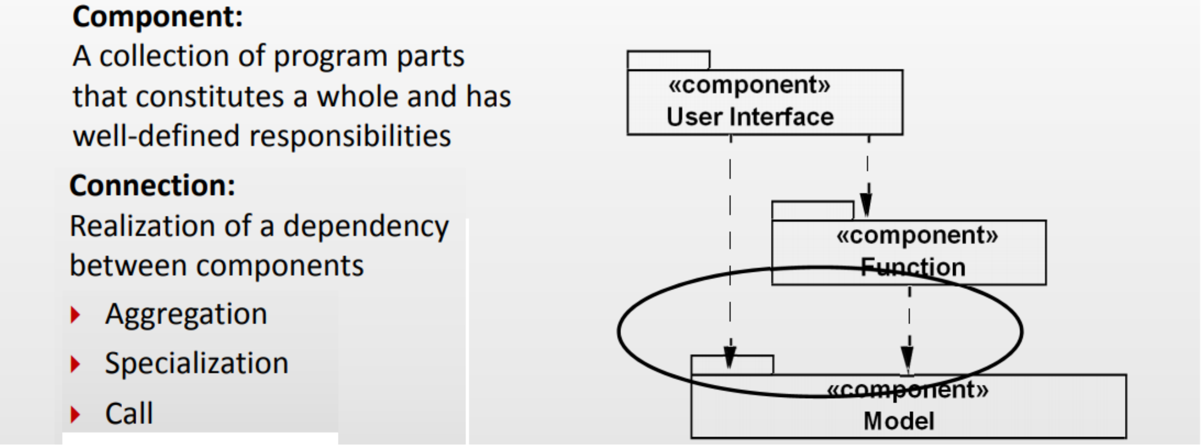
14: Connection of components: Evaluation of class diagrams.

Result



- A revised class diagram for the components of the system
- An architecture includes dependencies between components
- The implementation of these dependencies must be designed
- The dependencies are designed as connections between the classes in the components
- The revised class diagram specifies these connections precisely

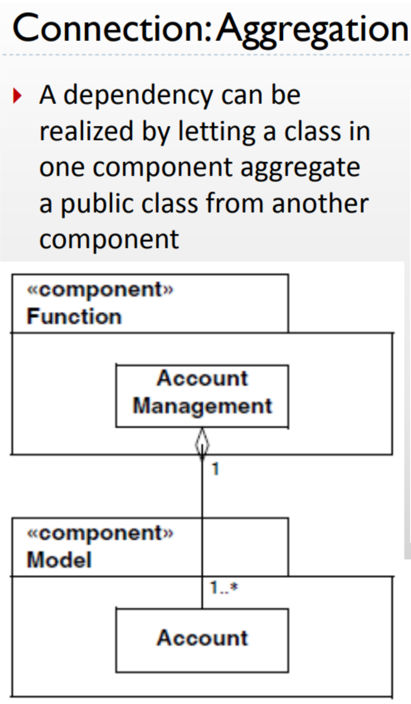

Class diagram and component specifications
Connect classes
Class diagrams and component specifications
Explore patterns
Evaluate connections



Component:
A collection of program parts that constitutes a whole and has well-defined responsibilities

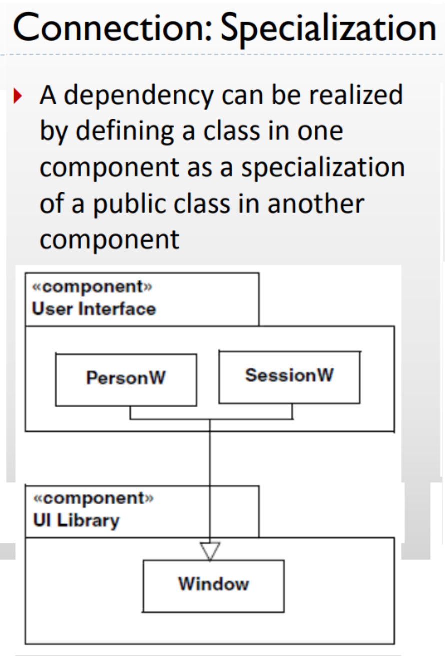
Connection:
Realization of a dependency between components

- Aggregation
- Specialization
- Call



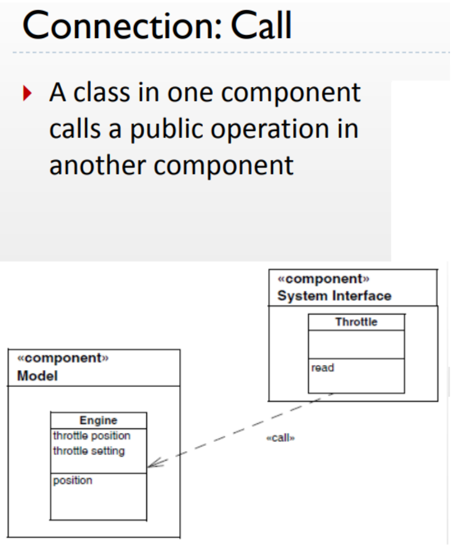
Connection: Aggregation

- A dependency can be realized by letting a class in one component aggregate a public class from another component



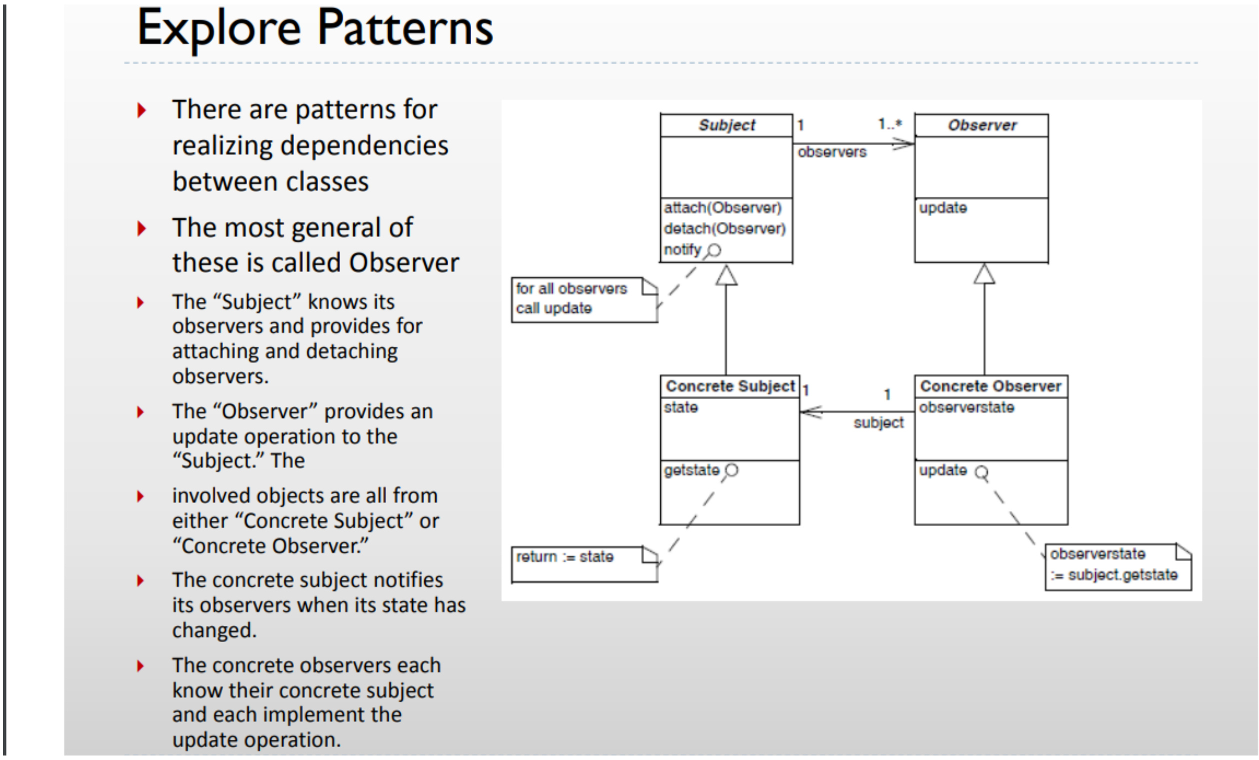
Connection: Specialization

- A dependency can be realized by defining a class in one component as a specialization of a public class in another component



Connection: Call

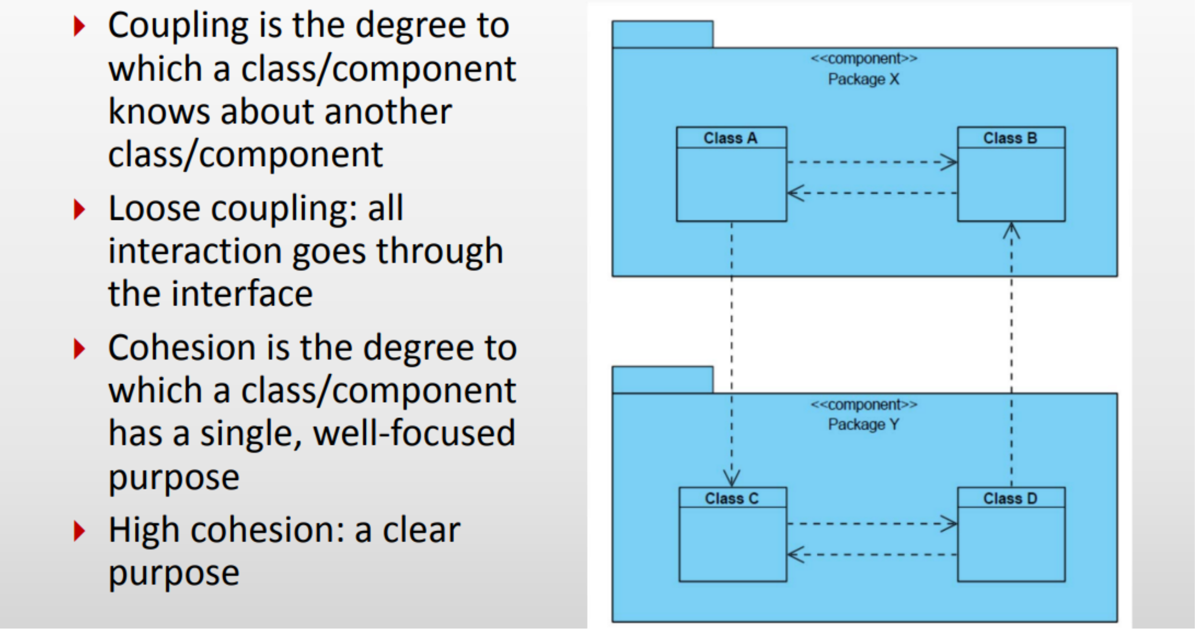
- A class in one component calls a public operation in another component



Explore Patterns

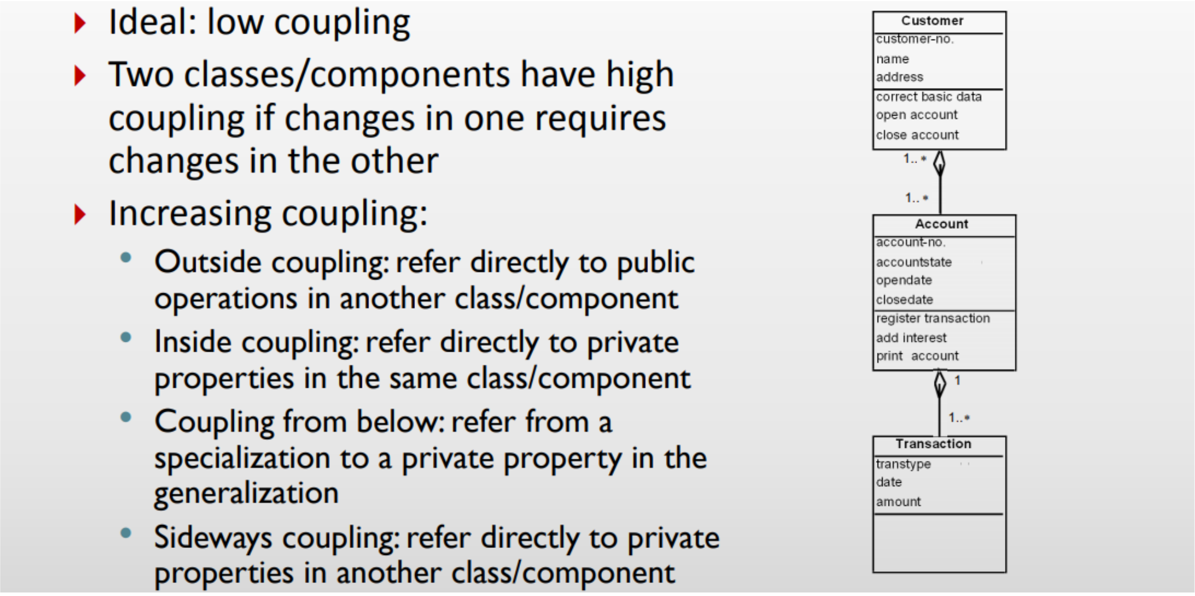
- There are patterns for realizing dependencies between classes
- The most general of these is called Observer
- The "Subject" knows its observers and provides for attaching and detaching observers.
- The "Observer" provides an update operation to the "Subject." The
- involved objects are all from either "Concrete Subject" or "Concrete Observer."
- The concrete subject notifies its observers when its state has changed.
- The concrete observers each know their concrete subject and each implement the update operation.

Evaluate Connections: Coupling and Cohesion



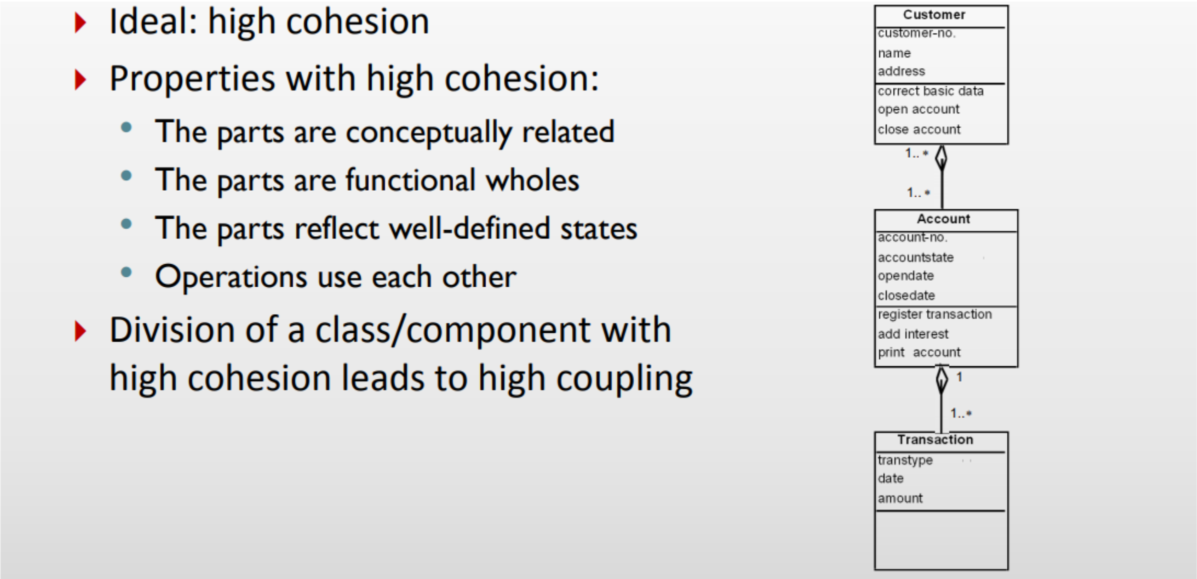
- Coupling is the degree to which a class/component knows about another class/component
- Loose coupling: all interaction goes through the interface
- Cohesion is the degree to which a class/component has a single, well-focused purpose
- High cohesion: a clear purpose

Evaluate Connections: Coupling



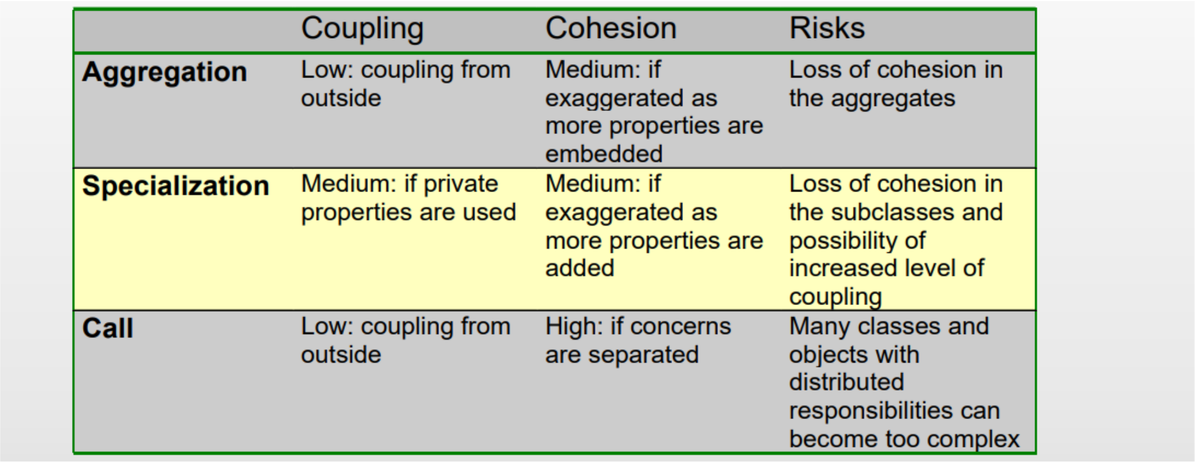
- Ideal: low coupling
- Two classes/components have high coupling if changes in one requires changes in the other
- Increasing coupling:
  - Outside coupling: refer directly to public operations in another class/component
  - Inside coupling: refer directly to private properties in the same class/component
  - Coupling from below: refer from a specialization to a private property in the generalization
  - Sideways coupling: refer directly to private properties in another class/component

Evaluate Connections: Cohesion



- Ideal: high cohesion
- Properties with high cohesion:
  - The parts are conceptually related
  - The parts are functional wholes
  - The parts reflect well-defined states
  - Operations use each other
- Division of a class/component with high cohesion leads to high coupling

Evaluate Connections



	Coupling	Cohesion	Risks
Aggregation	Low: coupling from outside	Medium: if exaggerated as more properties are embedded	Loss of cohesion in the aggregates
Specialization	Medium: if private properties are used	Medium: if exaggerated as more properties are added	Loss of cohesion in the subclasses and possibility of increased level of coupling
Call	Low: coupling from outside	High: if concerns are separated	Many classes and objects with distributed responsibilities can become too complex