

CHAPTER 13

Component Diagrams

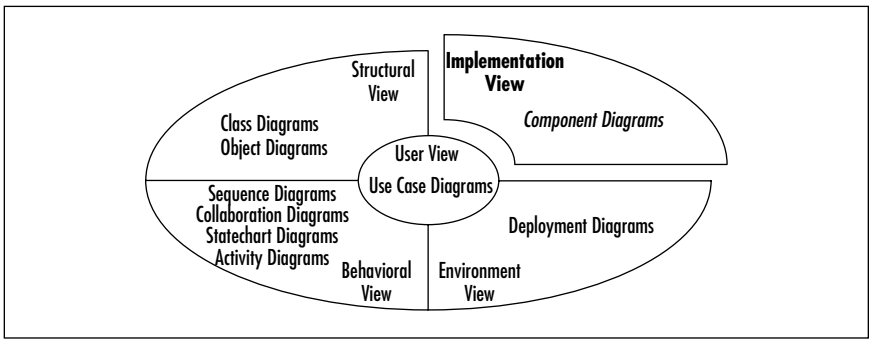


Figure 13-1: Implementation Model View—Component Diagrams

Component diagrams (Figure 13-1) render the implementation view of a system. These diagrams describe the organization of and dependencies among software implementation components. These diagrams only have a type form, not an instance form. These diagrams contain components and their relationships. They may be constructed using the following technique:

1. Model components.
2. Model development-time relationships between components.
3. Model calls relationships between components.
4. Refine and elaborate as required.

Component diagrams may have the various expressions (conditions, etc.) they utilize be expressed using pseudocode or another language.

Components

Components (Figure 13-2) are classes that define development-time and run-time physical objects. They are used to model distributable physical units containing

model elements and having identity and a well-defined interface. Units in software systems include software code in source, binary, or executable form. Units in human systems include humans, business documents, and so forth.



Figure 13-2: Components

Components

- Are denoted as rectangles with two small rectangles projected from the side.
- Have types that represent different kinds of software modules. Component types
  - Exist at development time, or the time that compilation of software modules occurs, and represent software units of storage and manipulation. Development time includes compile time and link time. During compile time, software is translated from source code to binary code. During link time, software is translated from binary code to executable code by resolving external references to other code units.
  - Are represented on component diagrams.
  - Must have a name or identifier string that represents the type of the component.
- Have instances that represent run-time manifestations of software module types. Component instances
  - Exist at run time, or during the time that execution of the system occurs, and represent executable software units.
  - Are represented on deployment diagrams (Chapter 14) (or degenerate deployment diagrams without nodes).
  - Must have a name or identifier string that represents the name of the component. The name must be underlined.
  - May have a colon followed by a type expression that indicates the type of the component.
- May have aggregation relationships with other components, objects, and processes (or active objects). The "location" tagged value may be used to specify this relationship (Chapter 15).
- May use other notation of classes and objects.
- May have a property string to indicate the form of the component (source, binary, executable) (Chapter 15).
- May be located on nodes.

Development-Time Relationships

Development-time relationships (Figure 13-3) are associations between components. They are used to model compile-time and link-time dependencies, that is, the dependency of one component on another component during development time.

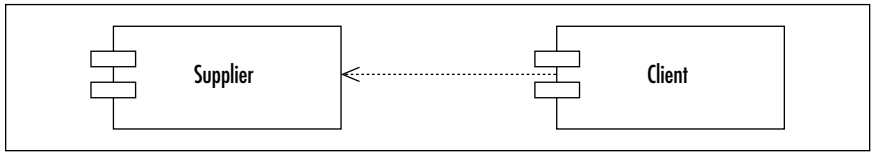


Figure 13-3: Development-Time Relationship

Development-time relationships

- Are denoted as dashed arrows.
- Have components with the following properties:
  - Client components are dependent on supplier components.
  - Supplier components exist at development time but need not exist at run time.
- May be stereotyped to depict an implementation-specific development-time dependency. When not stereotyped, the relationship represents a development-time communication or calls relationship.

Calls Relationships

Calls relationships (Figure 13-4) are associations between components. They are used to model calling or communication dependencies among components, that is, one component calling or using the services of another component. Calls relationships

- Are denoted as dashed arrows.
- Have components with the following properties:
  - Client components call or use the services (interfaces) of supplier components.
  - Supplier elements may be objects.
- May be stereotyped to depict an implementation-specific invocation.
- Exist at development time among development-time component types. They are represented on component diagrams.
- Exist at run time among run-time component instances. They are represented on deployment diagrams (Chapter 14) (or degenerate deployment diagrams without nodes).

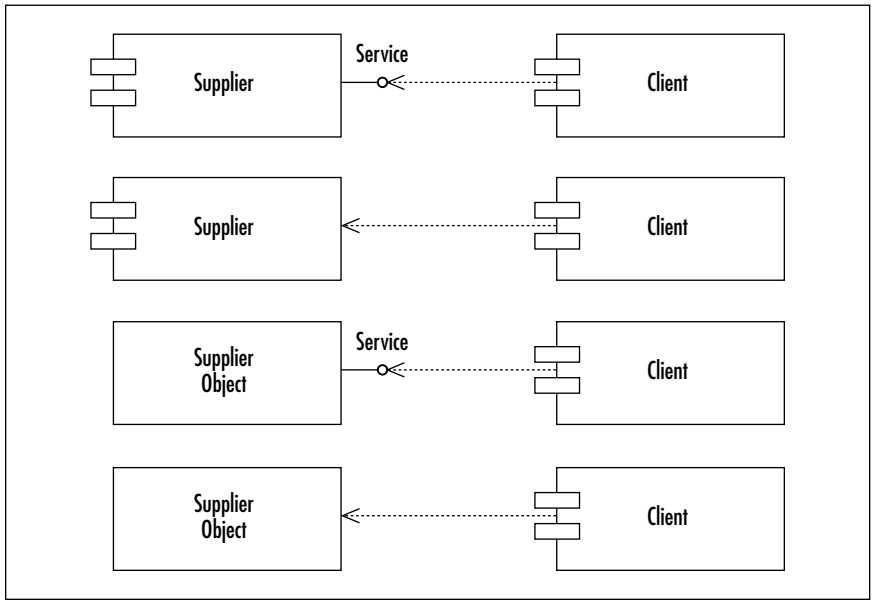


Figure 13-4: Calls Relationships