Appendix B
Notation Summary

This chapter contains a brief visual summary of notation. The major notational elements are included, but not every variation or option is shown. For full details, see the encyclopedia entry for each element.

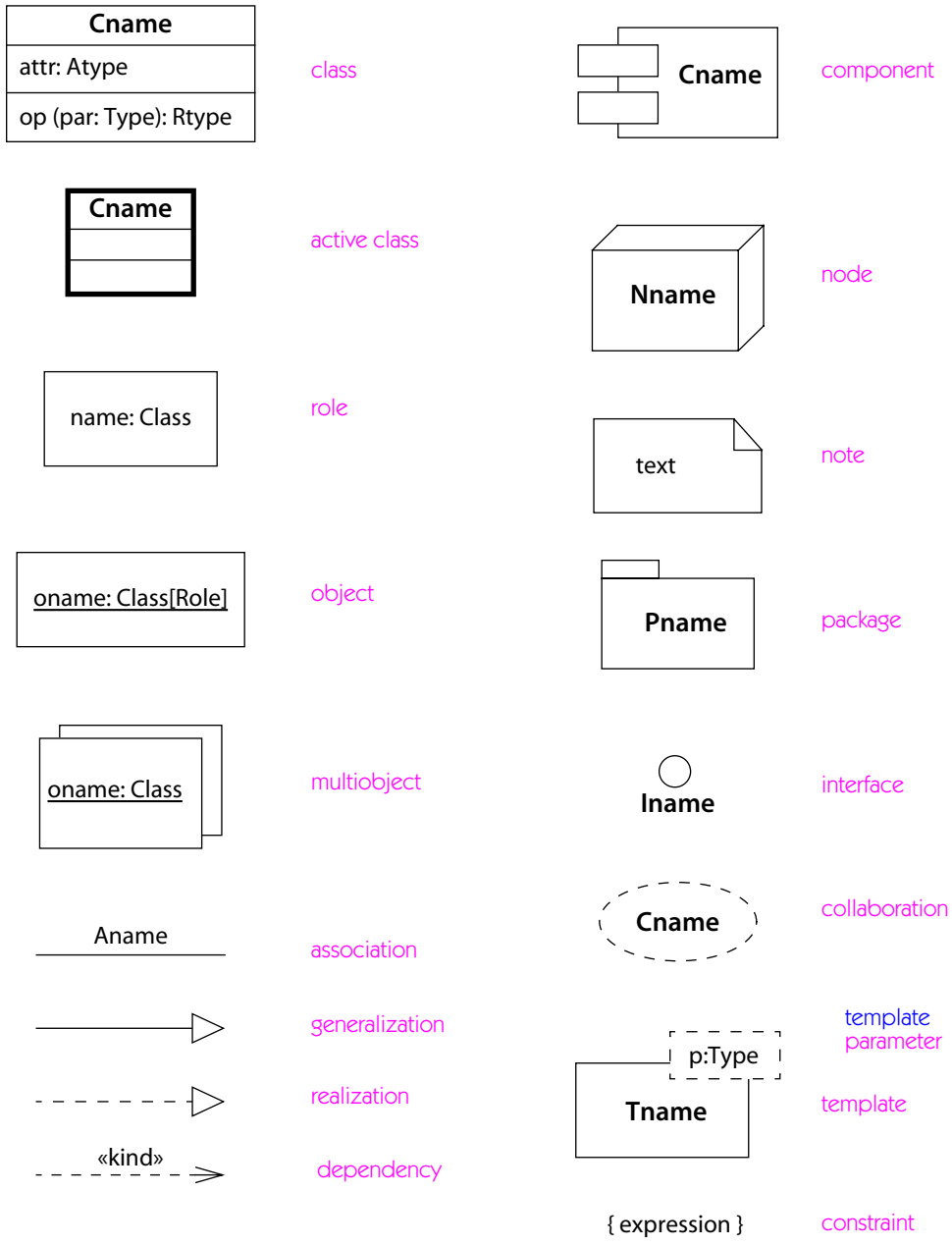


Figure B-1. *Icons on class, component, deployment, and collaboration diagrams*

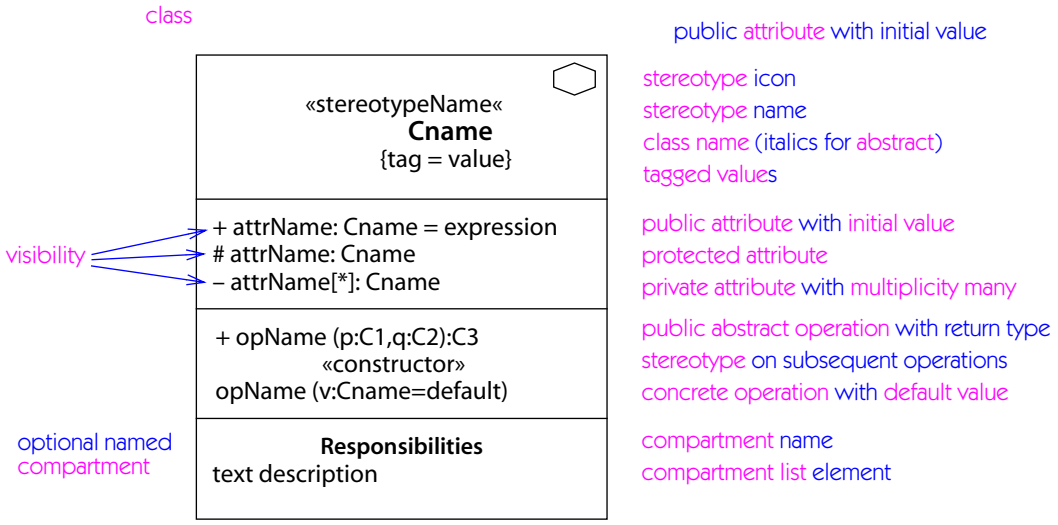


Figure B-2. *Class contents*

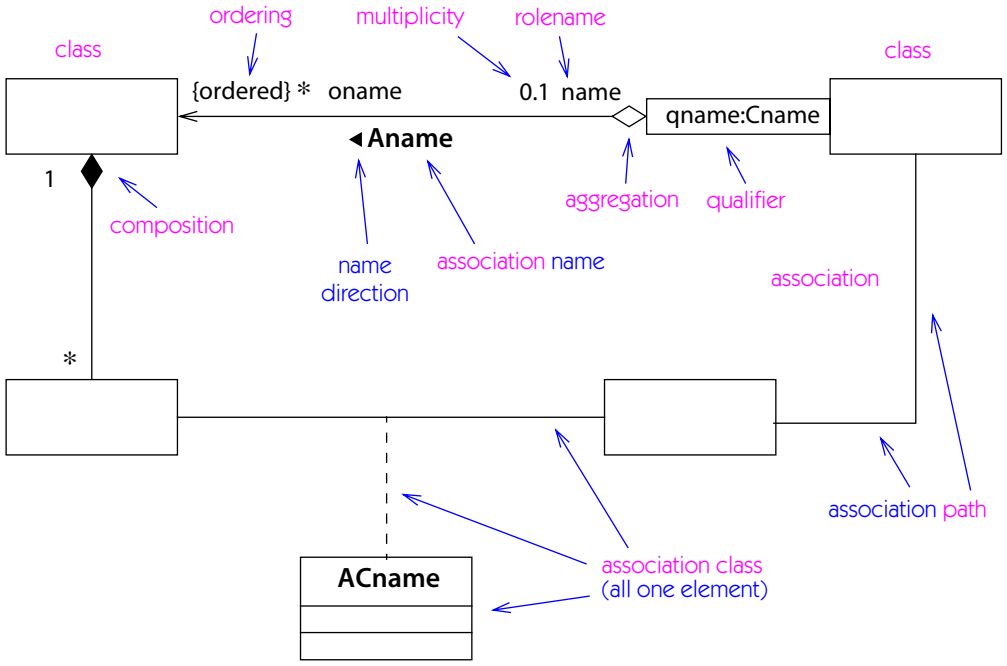


Figure B-3. *Association adornments within a class diagram*

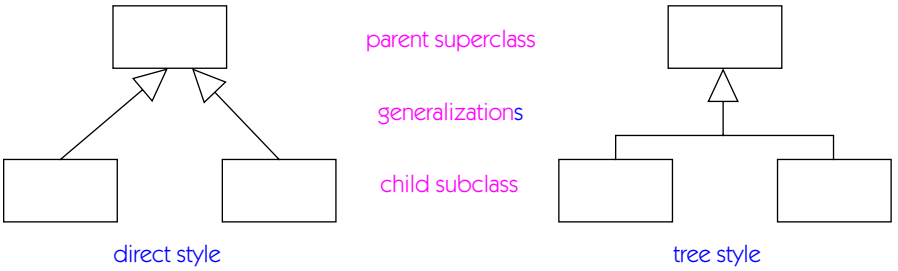


Figure B-4. *Generalization*

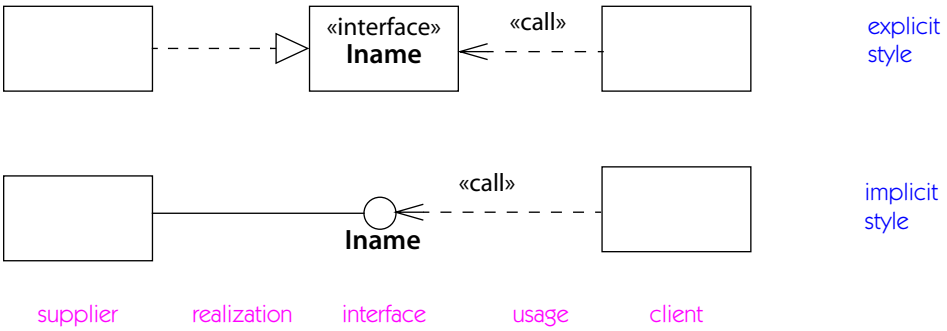


Figure B-5. *Realization of an interface*

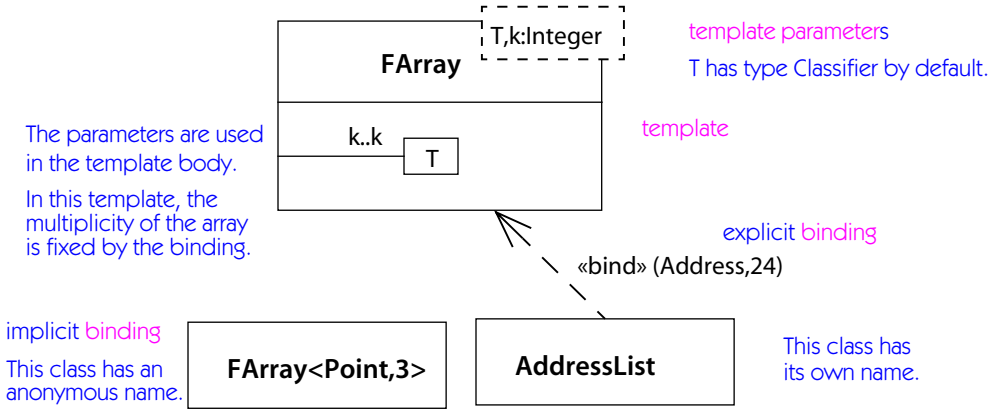


Figure B-6. *Template*

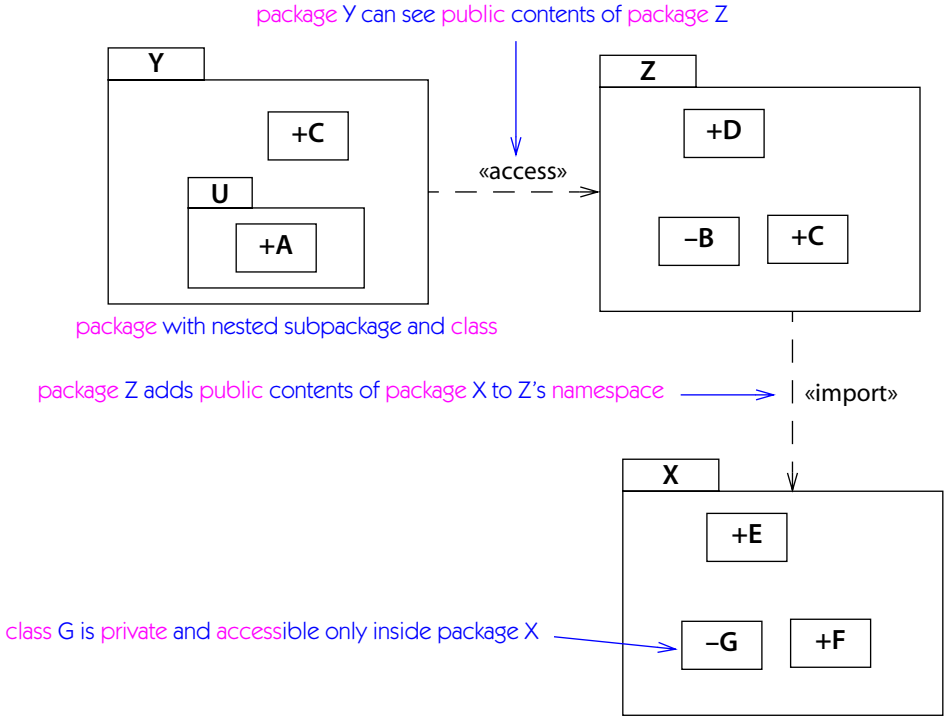


Figure B-7. *Package notation*

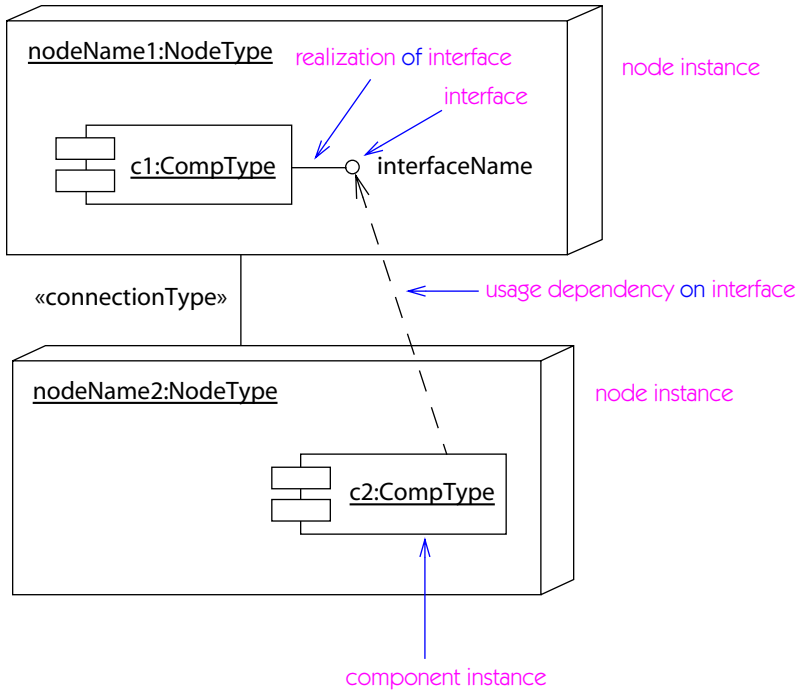


Figure B-8. *Component and node notation*

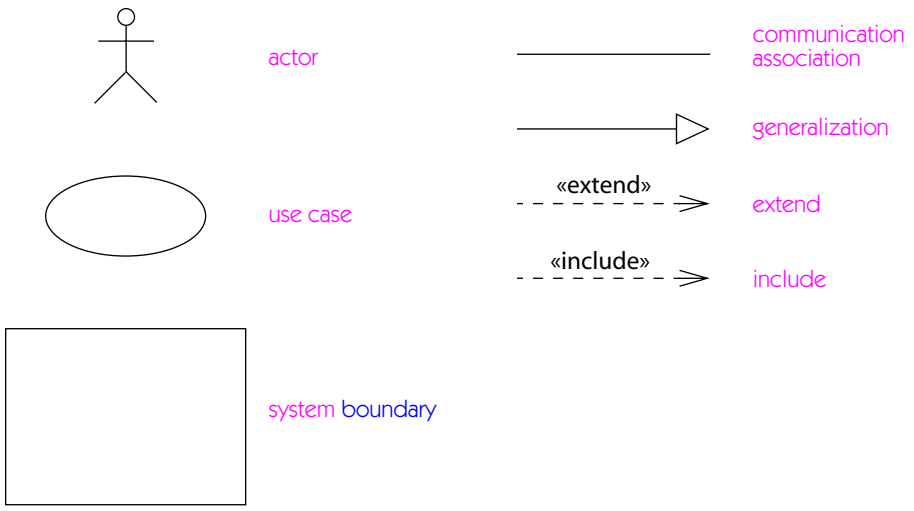


Figure B-9. *Icons on use case diagrams*

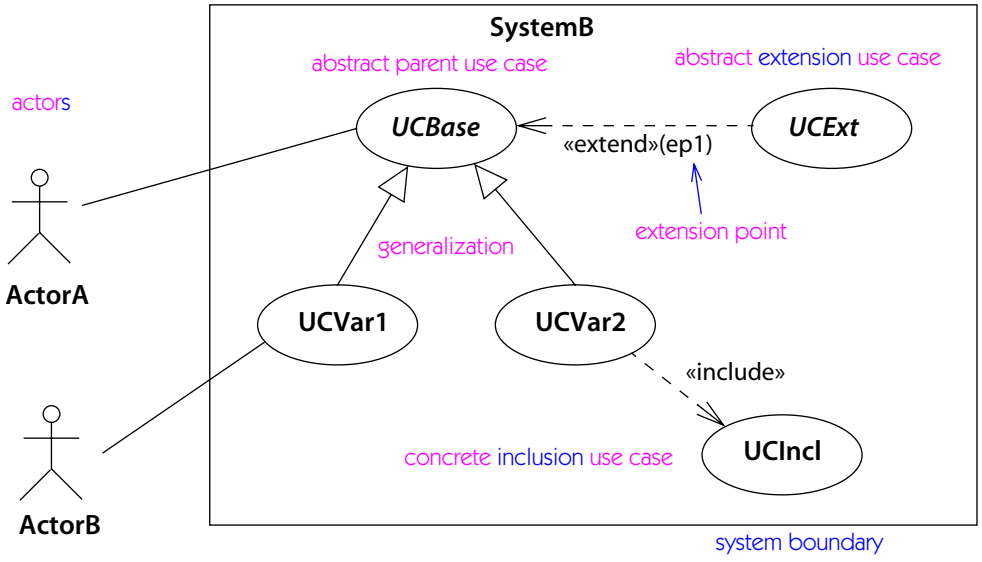


Figure B-10. *Use case diagram notation*

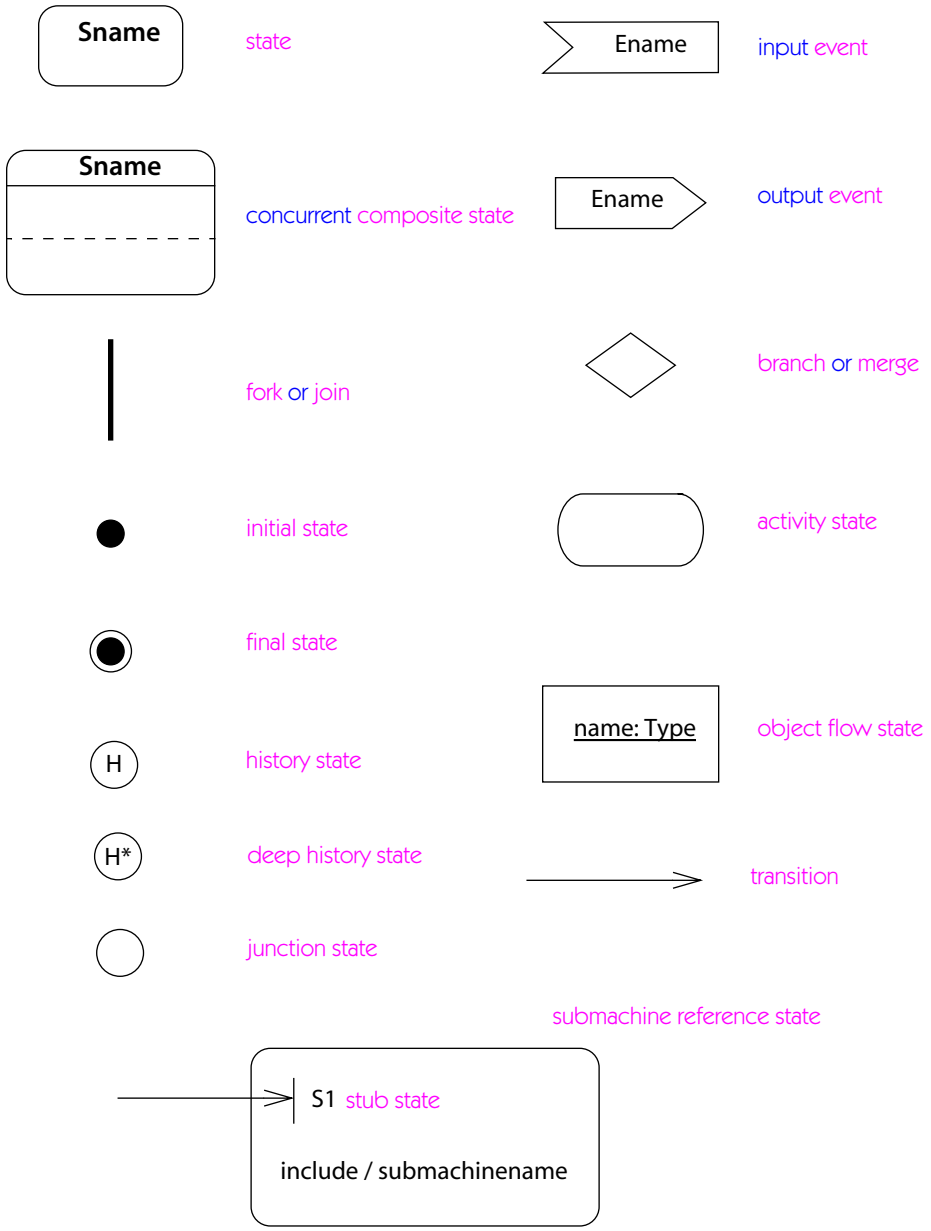


Figure B-11. *Icons on statechart and activity diagrams*

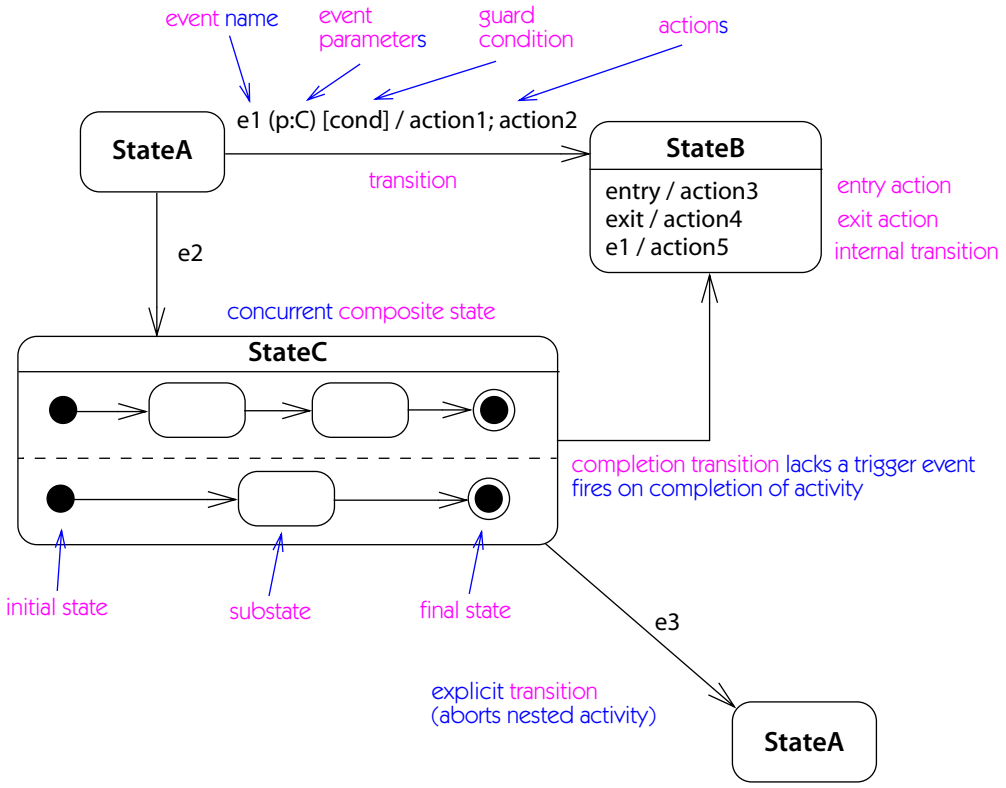


Figure B-12. *Statechart notation*

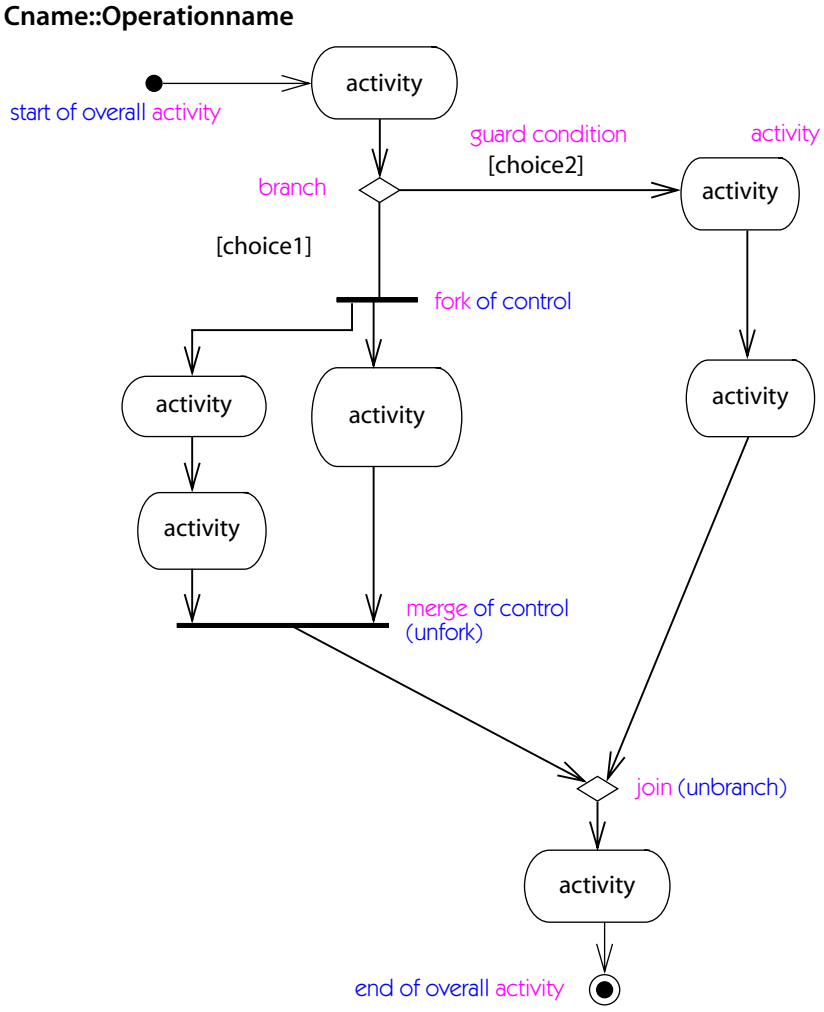


Figure B-13. *Activity diagram notation*

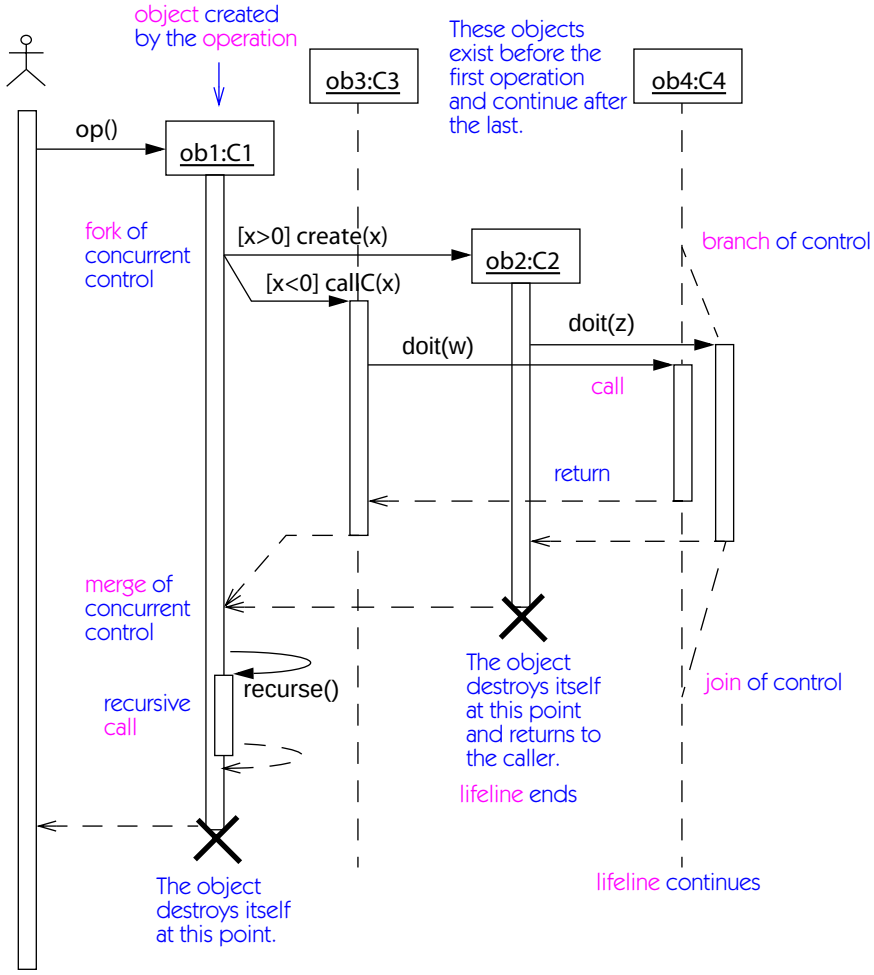


Figure B-14. *Sequence diagram notation*

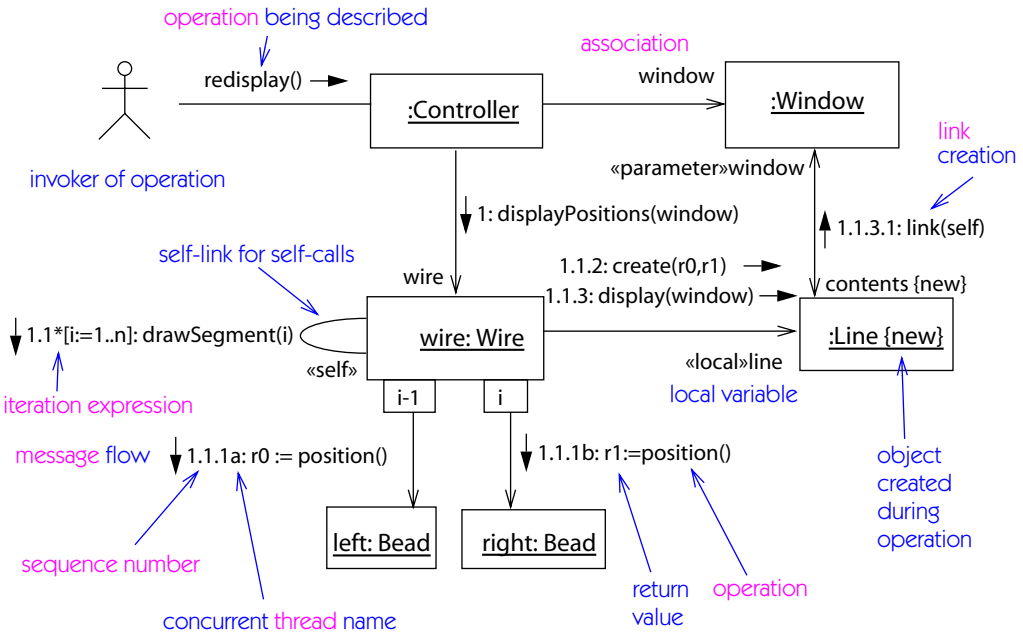


Figure B-15. *Collaboration diagram notation*

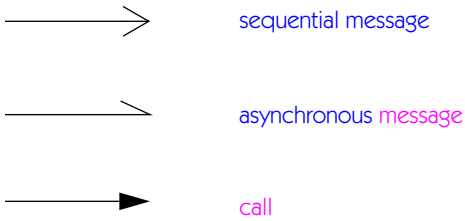


Figure B-16. *Message notation*

MORE UML NOTATION

38.1 General Notation

Stereotypes and Property Specifications with Tags

Stereotypes are used in the UML to classify an element (see Figure 38.1).

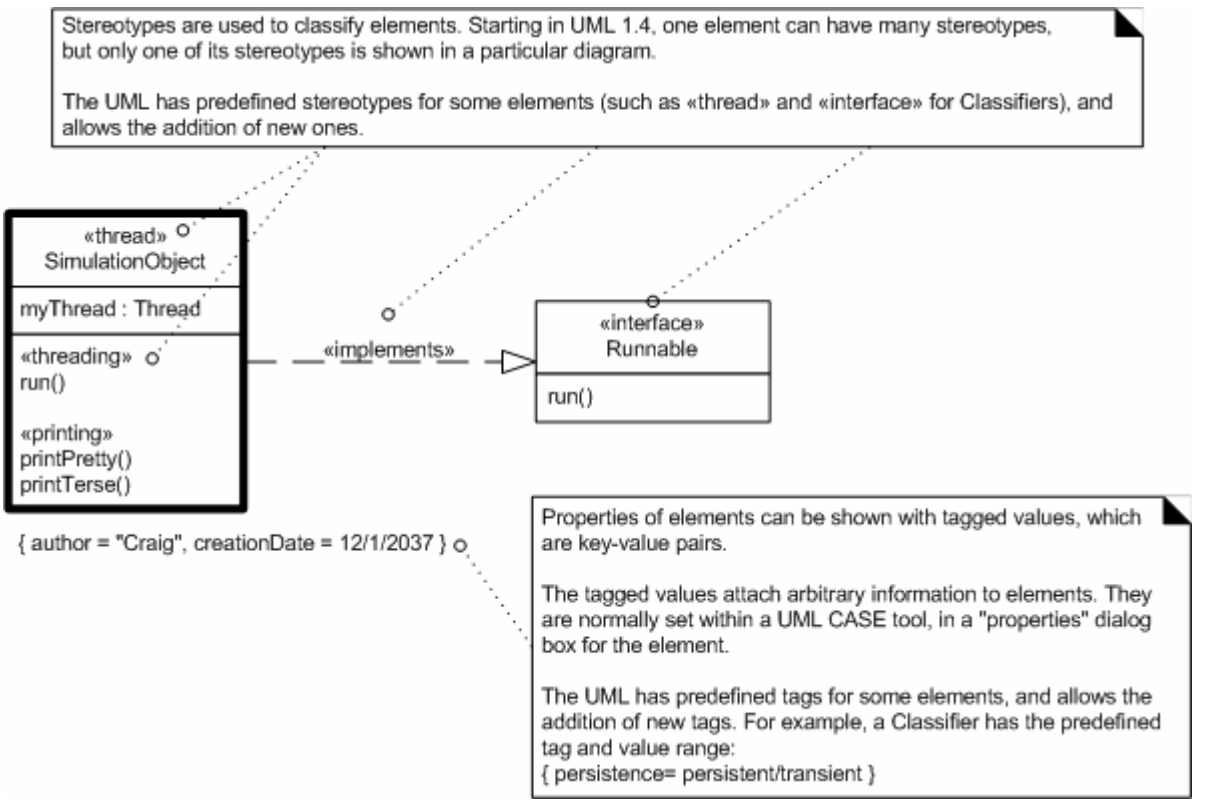
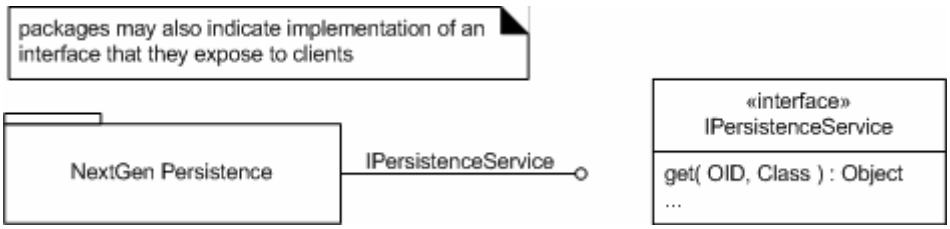


Figure 38.1 Stereotypes and properties.

Figure 38.2 Interface of a package.

Package Interfaces

A package can he illustrated as implementing an interface (see Figure 38.2).



Dependency

Dependencies can exist between any elements, but they are probably most often used in UML package diagrams to illustrate package dependencies (see Figure 38.3)

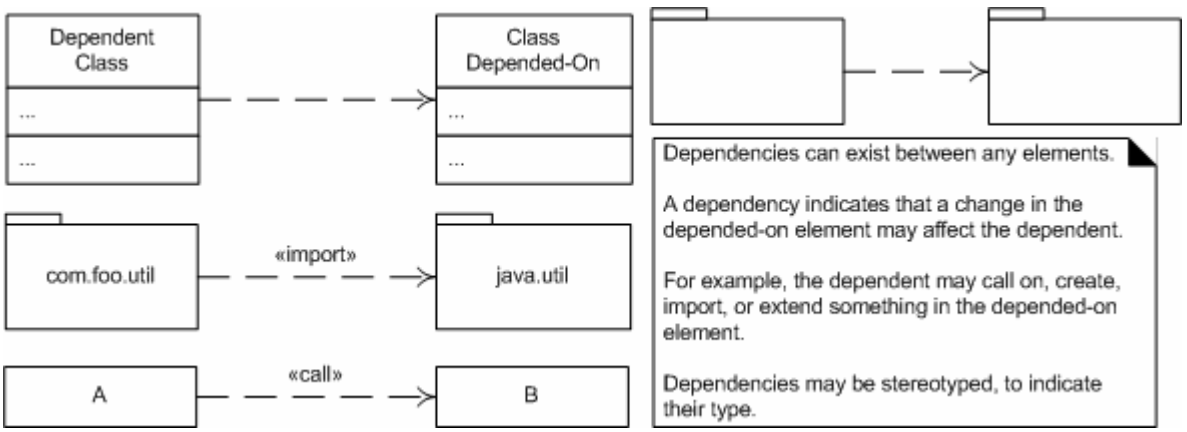


Figure 38.3 Dependencies.

38.2 Implementation Diagrams

The UML defines several diagrams that can be used to illustrate implementation details. The most commonly used is a deployment diagram, to illustrate the deployment of components and processes to processing nodes.

Component Diagrams

To quote: A **component** represents a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces [OMG01]. It may, for example, be source code, binary, or executable. Examples include executables such as a browser or HTTP server, a database, a DLL, or a JAR file (such as for an Enterprise Java Bean). UML components are usually shown within deployment diagrams, rather than on their own. Figure 38.4 illustrates some common notation.

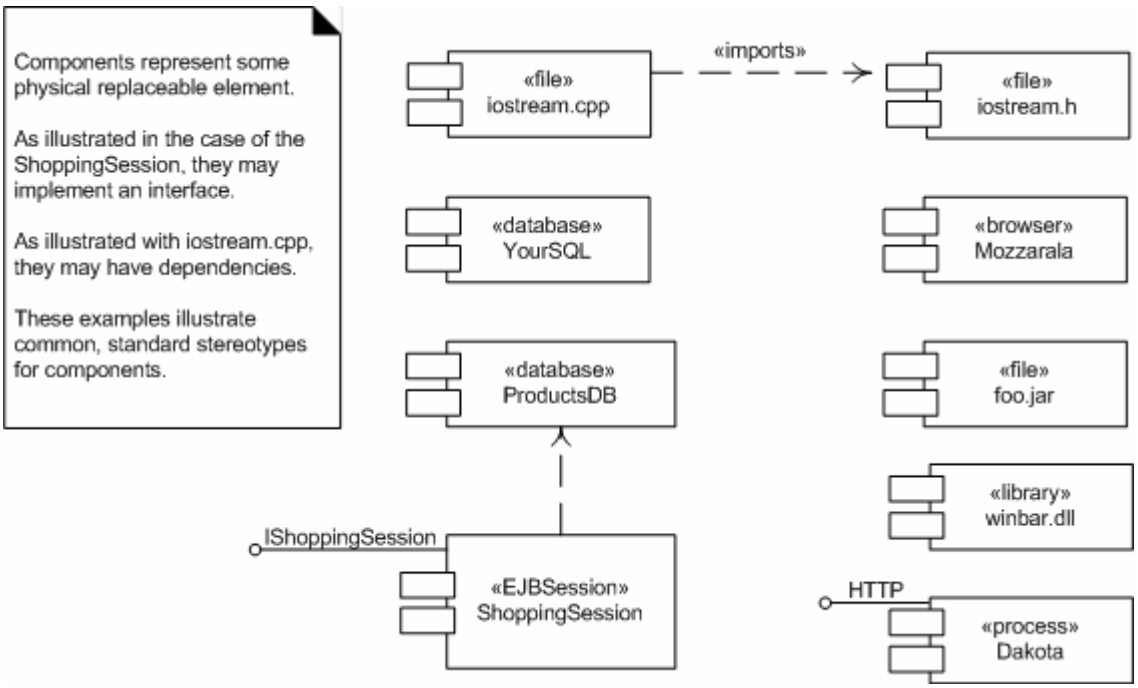


Figure 38.4 UML components.

Deployment Diagrams

A deployment diagram shows how instances of components and processes are configured for run-time execution on instances of processing **nodes** (something with memory and processing services; see Figure 38.5).

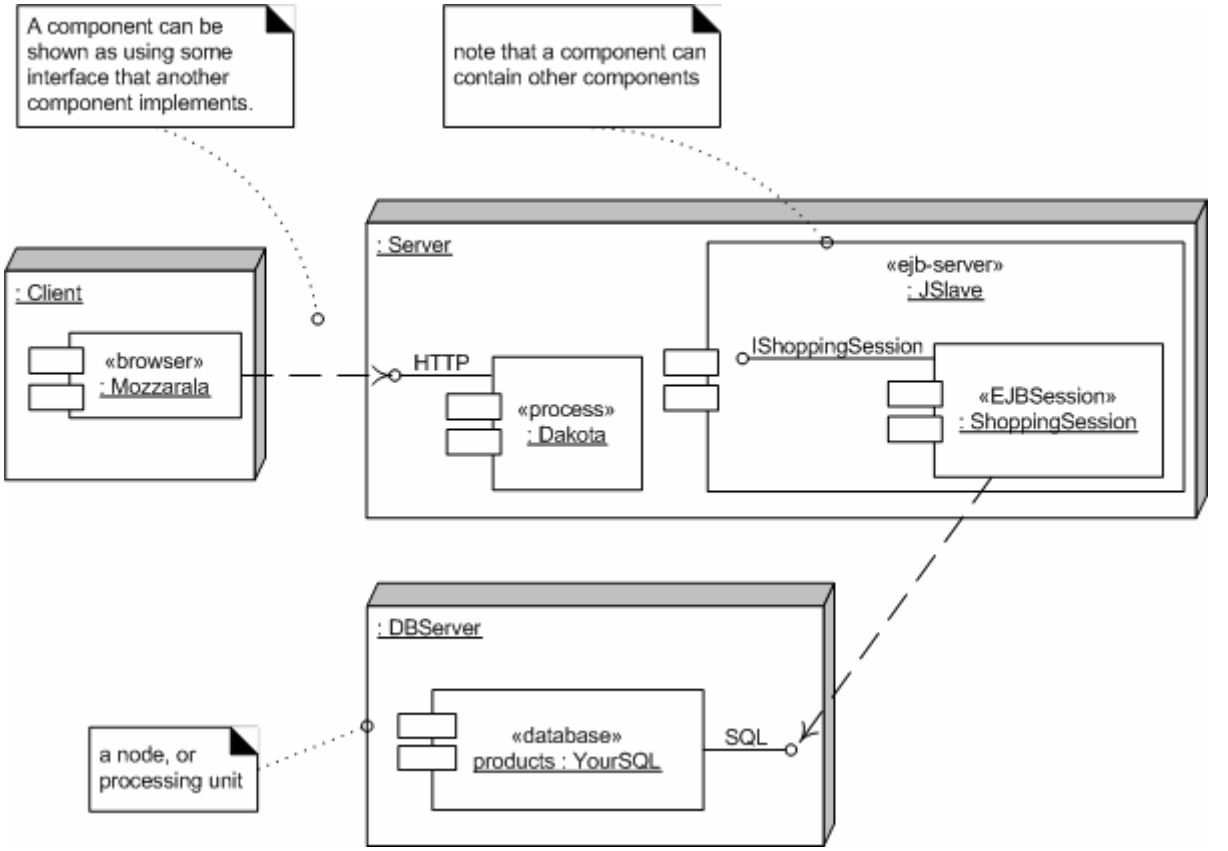


Figure 38.5 A deployment diagram.

38.3 Template (Parameterized, Generic) Class

Template classes and their instantation are shown in Figure 38.6.

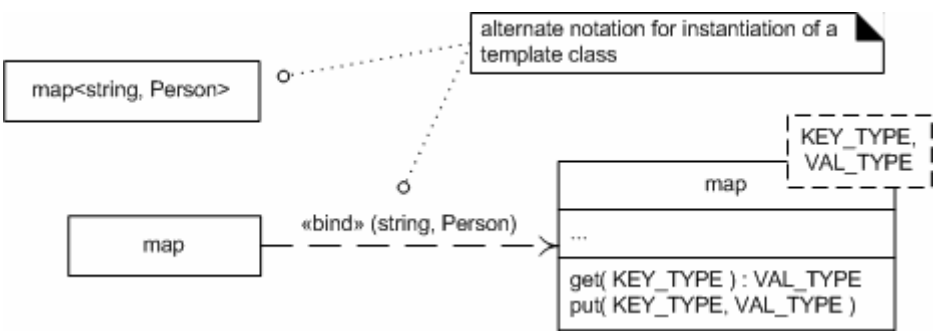


Figure 38.6 Template classes.

Some languages, such as C++, support templatized, generic, or parameterized classes. In addition, this feature will be added to the Java language. For example, in C++, *map<string, Person>* declares the instantiation of a template class with keys of type *string,* and values of type *Person.*

38.4 Activity Diagrams

A UML activity diagram offers rich notation to show a sequence of activities. It may be applied to any purpose (such as visualizing the steps of a computer algorithm), but is considered especially useful for visualizing business workflows and processes, or use cases. One of the UP workflows (disciplines) is **Business Modeling;** its purpose is to understand and communicate "the structure and the dynamics of the organization in which a system is to be deployed" [RUP]. A key artifact of the Business Modeling discipline is the **Business Object Model** (a superset of the UP Domain Model), which essentially visualizes how a business works, using UML class, sequence, and activity diagrams. Thus, activity diagrams are especially applicable within the Business Modeling discipline of the UP.

Some of the outstanding notation includes parallel activities, swimlanes, and action-object flow relationships, as illustrated in Figure 38.7 (adapted from [OMGOl, FS00]). Formally, an activity diagram is considered a special kind of UML statechart diagram in which the states are actions, and event transition is automatically triggered by action completion.

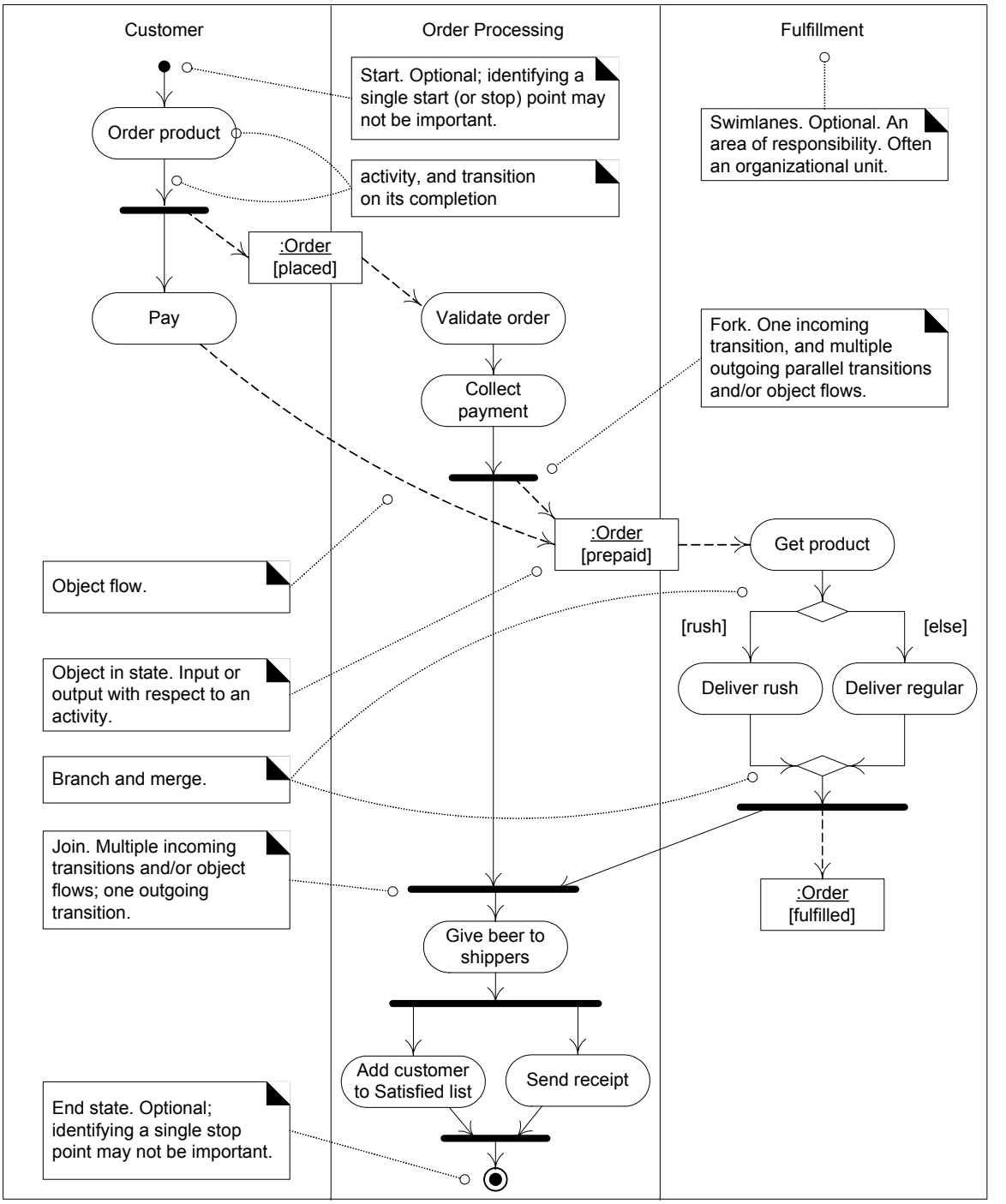


Figure 38.7 Activity diagram.