History of UML
Unified Modelling Language

- UML is a graphical language for visualizing, specifying, constructing, and documenting software artifacts.
- UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as PL statements, DB schemas, or reusable components.
- UML is a set of notations, not a methodology and not a process
  - Version 2.2 is the latest standard (Feb 2009)
  - There are now 14 kinds of diagrams

http://www.uml.org/

History of UML
Unified Modelling Language

- UML does have an official standard
  - Backed by OMG (Object Management Group)
  - OMG is a not-for-profit industry specifications consortium
  - OMG members define and maintain the UML spec
  - Software providers build tools to conform to these specs
- Rational (now owned by IBM) is the big mover behind UML,
- but they don't "own" UML
- Tremendous history and politics behind UML
- Many expensive tools, seminars, books, hype, … but
  - UML is just a set of notations
  - UML doesn't solve the problems, it gives a way of writing them down

http://www.omg.org/

Domains Covered by UML Notations and Semantics

- User Interaction or Use Case Model
  - Describes the boundary and interaction between the system and users
  - Corresponds in some respects to a requirements model
- Interaction or Communication Model
  - Describes how objects in the system will interact with each other to get work done
- State or Dynamic Model
  - State charts describe the states or conditions that classes assume over time.
  - Activity graphs describe the workflows the system will implement
- Logical or Class Model
  - Describes the classes and objects that will make up the system
- Physical Component Model
  - Describes the software and hardware components that make up the system
- Physical Deployment Model
  - Describes the physical architecture and the deployment of components

History of Analysis and Design Notations

1970s

- Procedural languages
  - COBOL, FORTRAN, PL/I, C, Pascal
- Systems are structured as TDFD
  - TDFD == top-down functional decomposition
- Data is mostly global and passive
- Notations and tools
  - Entity Relationship (ER) diagrams
    - Originally for DB design
  - Data-flow diagrams (DFD)
  - Control-flow diagrams (CFG)
  - Flowcharts
  - State transition diagrams STD
    - STDs (for state-oriented engineering applications)
  - Data dictionaries
- Methodologies
  - Structured analysis

History of Analysis and Design Notations

1980s

- Some OO languages emerge
  - Simula-67, C++, Objective-C, Objective Pascal, OO-Fortran, OO-Cobol
- Systems structured as modules, use info-hiding & interfaces
- Data is encapsulated; must use interfaces
- Notations and tools
  - Class/object diagrams (ER++) for analysis modelling
  - Statecharts (formal STDs for engineering applications)
  - Message sequence charts (MSC)
  - Use cases (Ivar Jacobson)
- Methodologies
  - Object Modeling Technique (OMT) (Jim Rumbaugh)
  - Object-Oriented Analysis (OOA) and Object-Oriented Design (OOD)
  - (OOA/D) (Grady Booch)
  - Computer-Aided Software Engineering (CASE) tools
  - Many others

History of Analysis and Design Notations

1990s

- Most of the software industry is tired of tool/notation wars
  - An agreement on a notation without religion
- The three amigos gather at Rational
  - Grady Booch, Jim Rumbaugh and Ivar Jacobson
  - They announce war is over (if you want it)
  - → UML
- UML takes a kitchen-sink approach to diagram design
  - Contains many kinds of diagrams
  - Makes few restrictions on how to use them
  - Model various views
    - Requirements
    - Architecture
    - Design
    - Implementation
    - Dynamic or run-time

Overview of UML Diagrams

Structural

: element of spec. irrespective of time

- Class
- Component
- Deployment
- Object
- *Composite structure*
- *Package*

Behavioral

: behavioral features of a system / business process

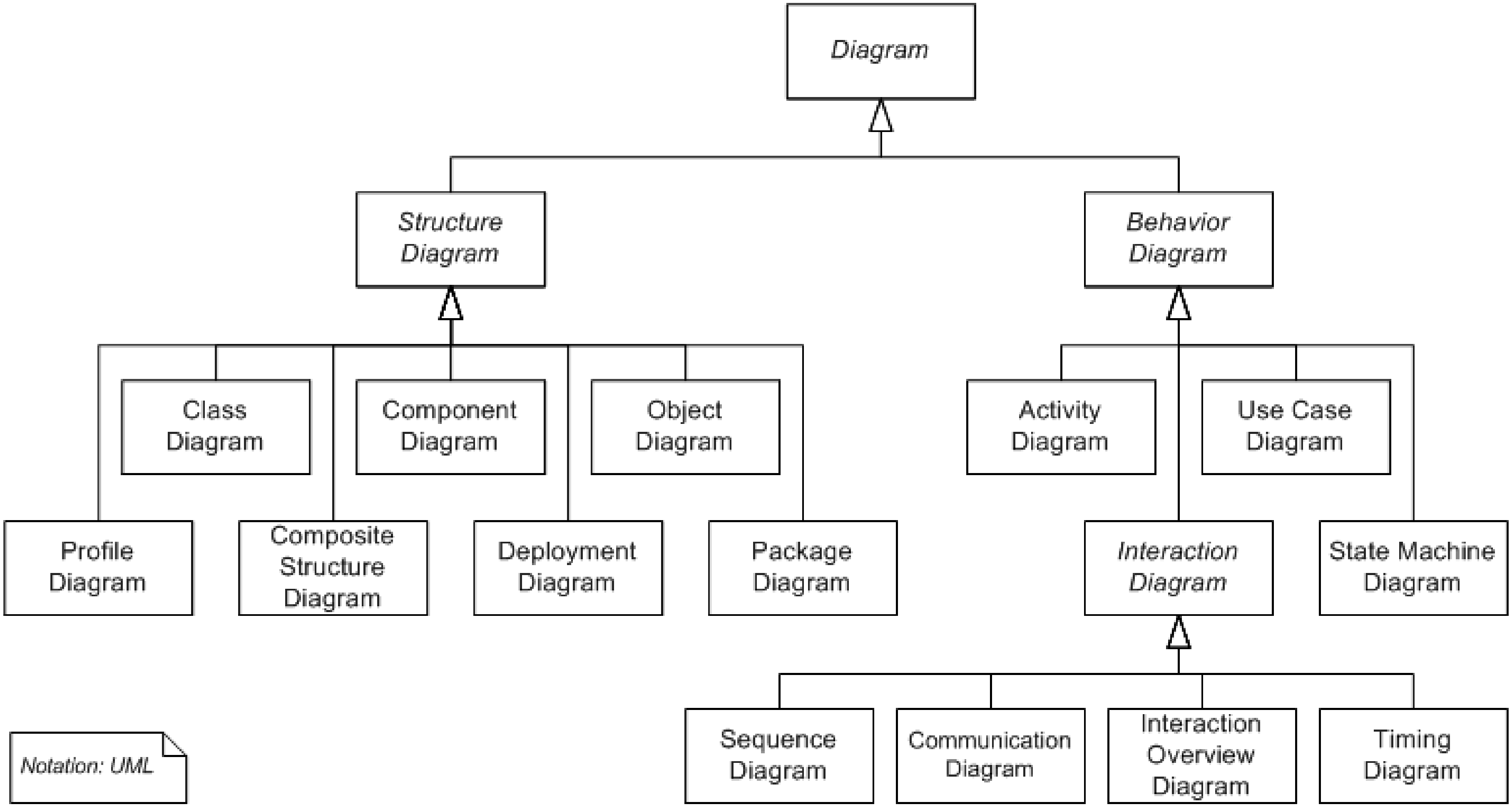
- Activity
- State machine
- Use case
- *Interaction*

Interaction

: emphasize object interaction

- Communication(collaberation)
- Sequence
- *Interaction overview*
- *Timing*

UML diagram hierarchy



Class diagram

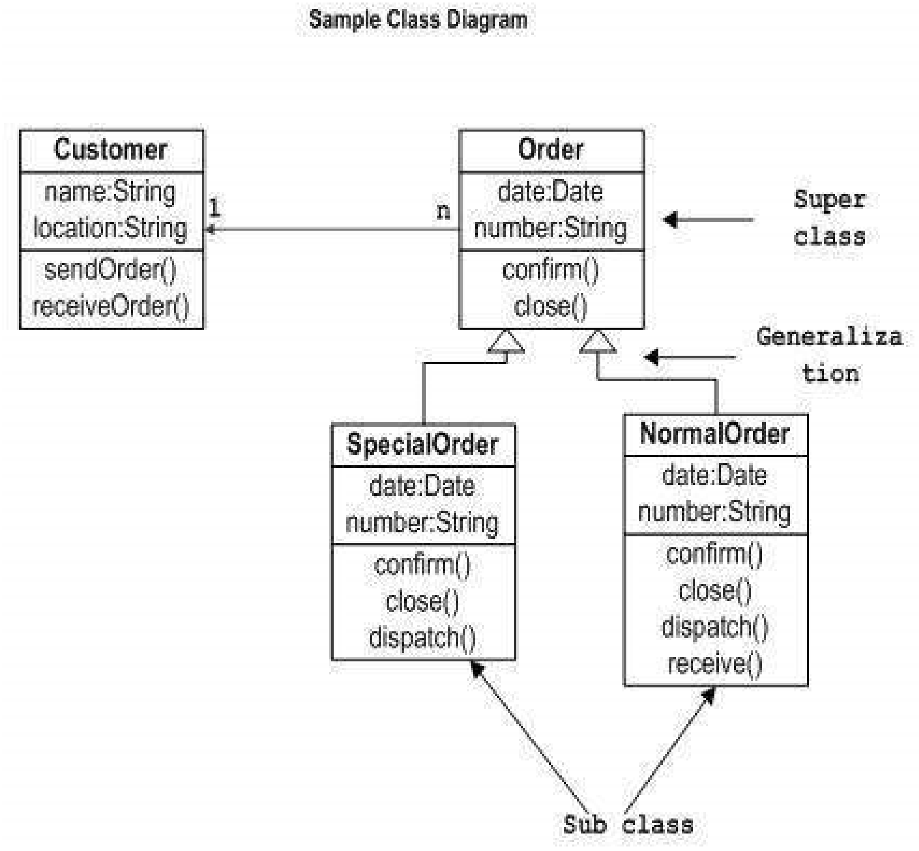
UML class diagrams show the classes of the system, their inter-relationships, and the operations and attributes of the classes

Explore domain concepts in the form of a domain model

Analyze requirements in the form of a conceptual/analysis model

Depict the detailed design of object-oriented or object-based software

Class diagram



Class diagram

So in a brief, class diagrams are used for:

•Describing the static view of the system.

•Showing the collaboration among the elements of the static view.

•Describing the functionalities performed by the system.

•Construction of software applications using object oriented languages.

Use case diagram

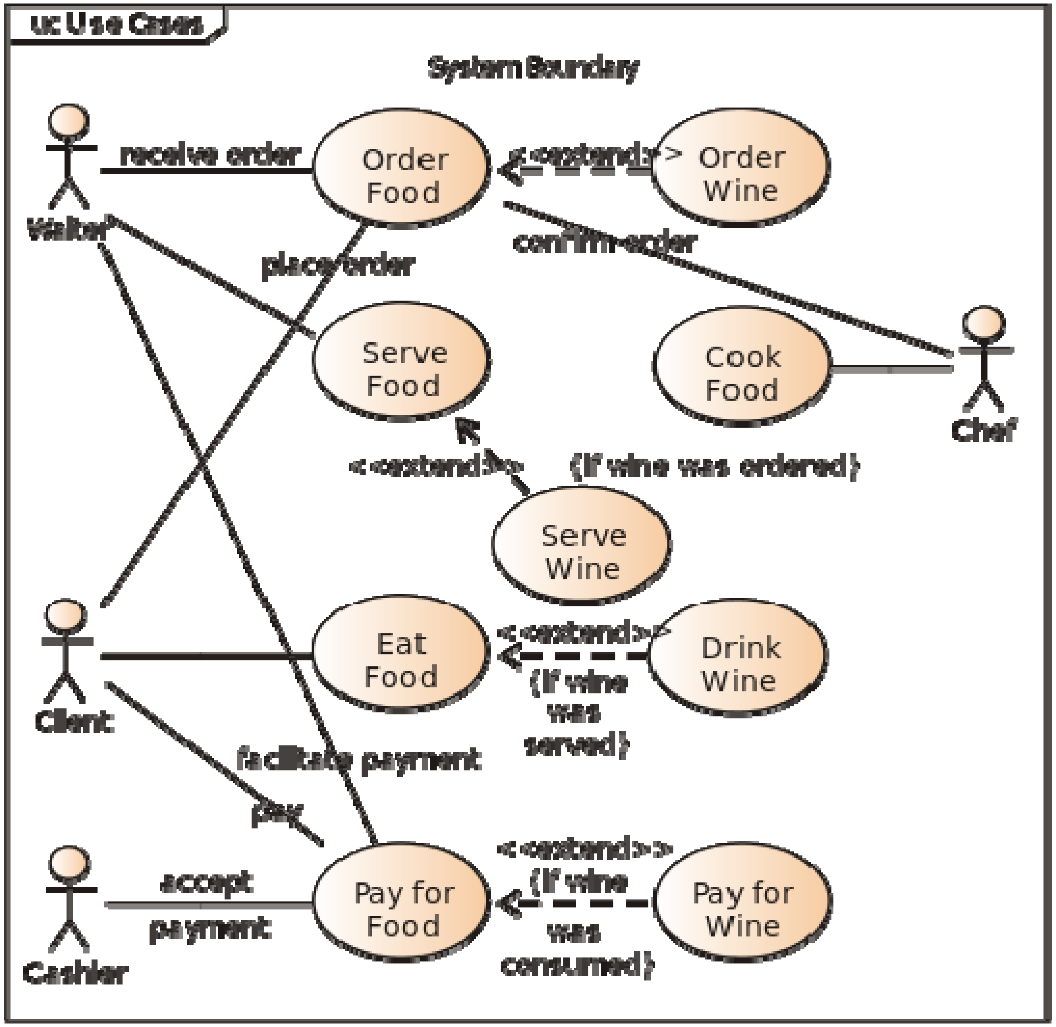
UML Use cases diagrams describes the behavior of the target system from an external point of view. Use cases describe "the meat" of the actual requirements.

Use cases. A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

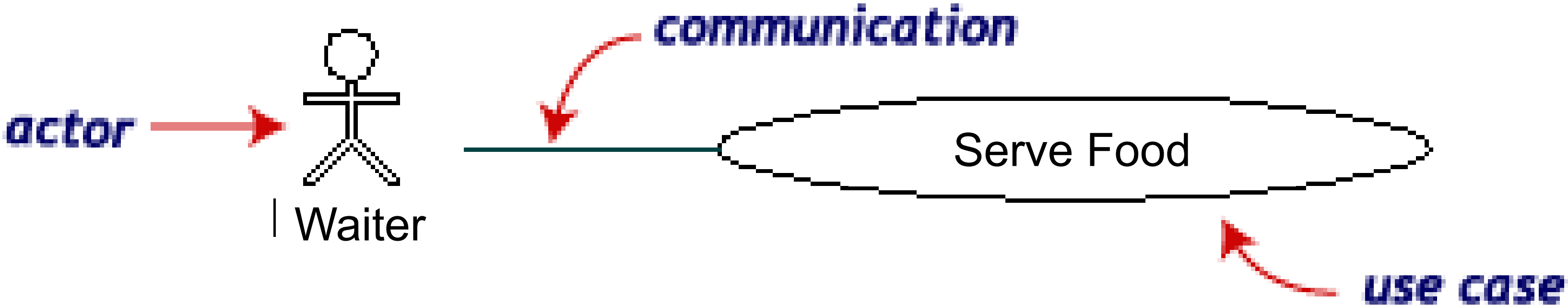
Actors. An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

Associations. Associations between actors and use cases are indicated by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

Use case diagram



Use case diagram

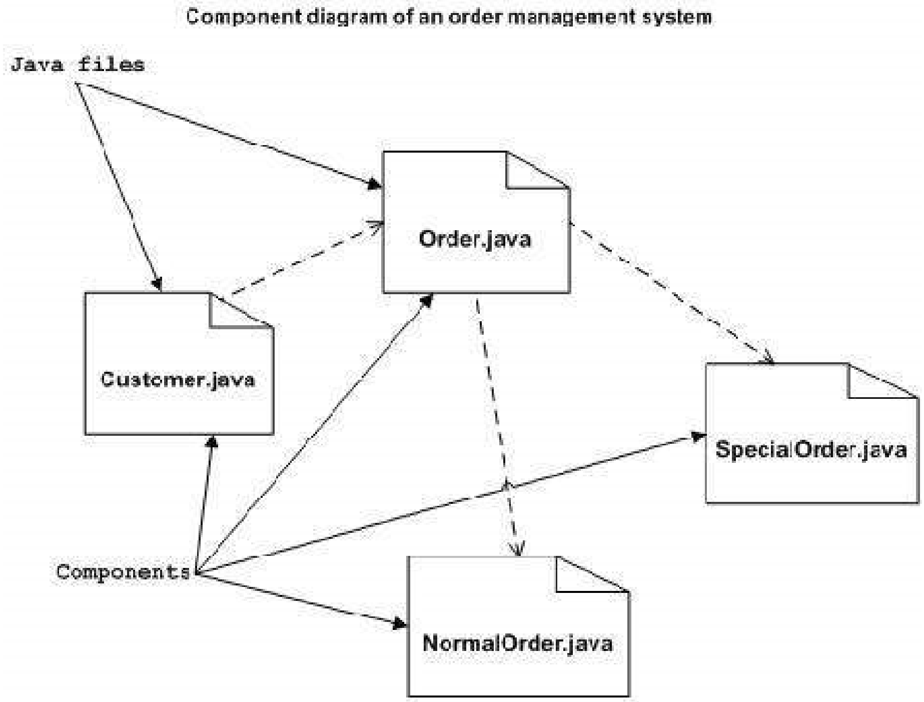


Component Diagram

Component diagrams are used to model physical aspects of a system.

Physical aspects are elements such as executables, libraries, files, documents etc., which reside in a node.

Component Diagram



Dynamic Modelling

Structural Diagrams model the static aspect of the system. Most of the behavioral diagrams model the dynamic behavior of the system.

> This may lead to identification of new classes.

Dynamic modelling can be done by:

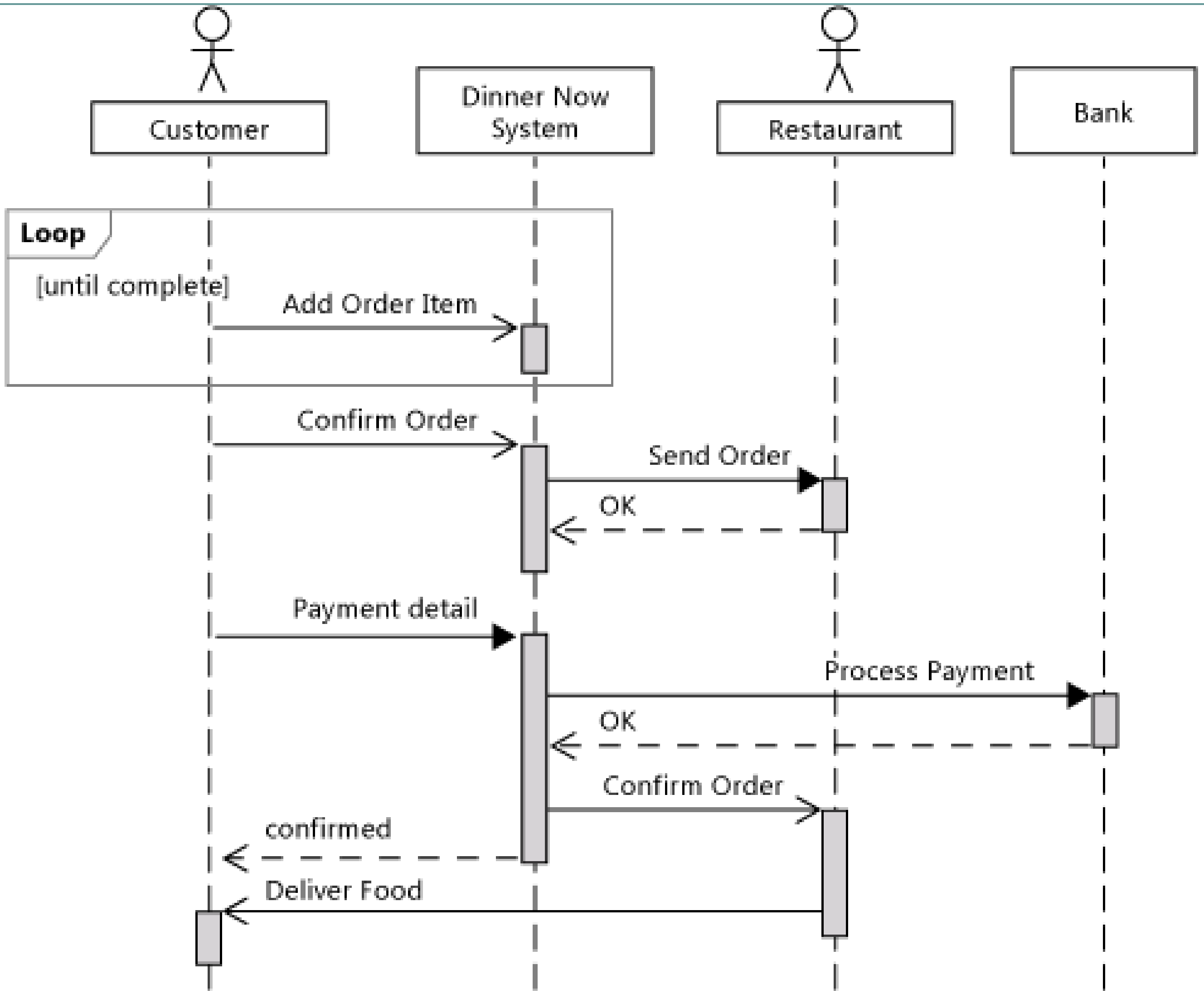
Sequence Diagrams

State Diagrams

Sequence diagram

UML Sequence diagrams models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case.

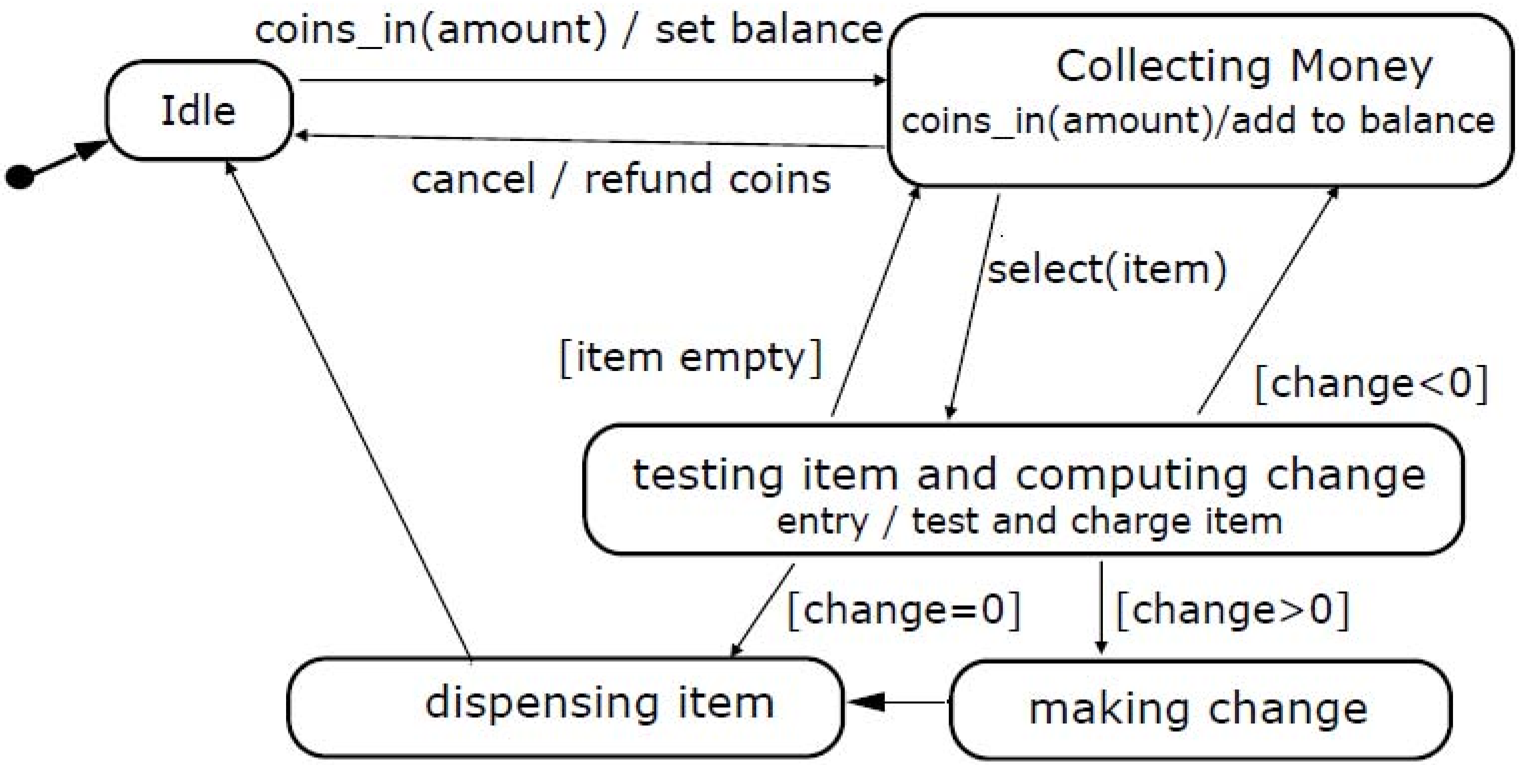
Sequence diagram



Statechart Diagram

- Graph whose nodes are states and whose directed arcs are transitions labeled by event names
- We distinguish between two types of operations in statecharts:
  - Activity: Operation that takes time to complete
    - associated with states
    - (in UML:) can be described by its own Activity diagram
  - Action: "Instantaneous" operation (in UML: elementary op.)
    - associated with events
    - associated with states (reduces drawing complexity): Entry, Exit, Internal Action
- A statechart diagram relates events and states for one class
  - An object model with a set of objects
    can have a corresponding set of state diagrams

Statechart Diagram



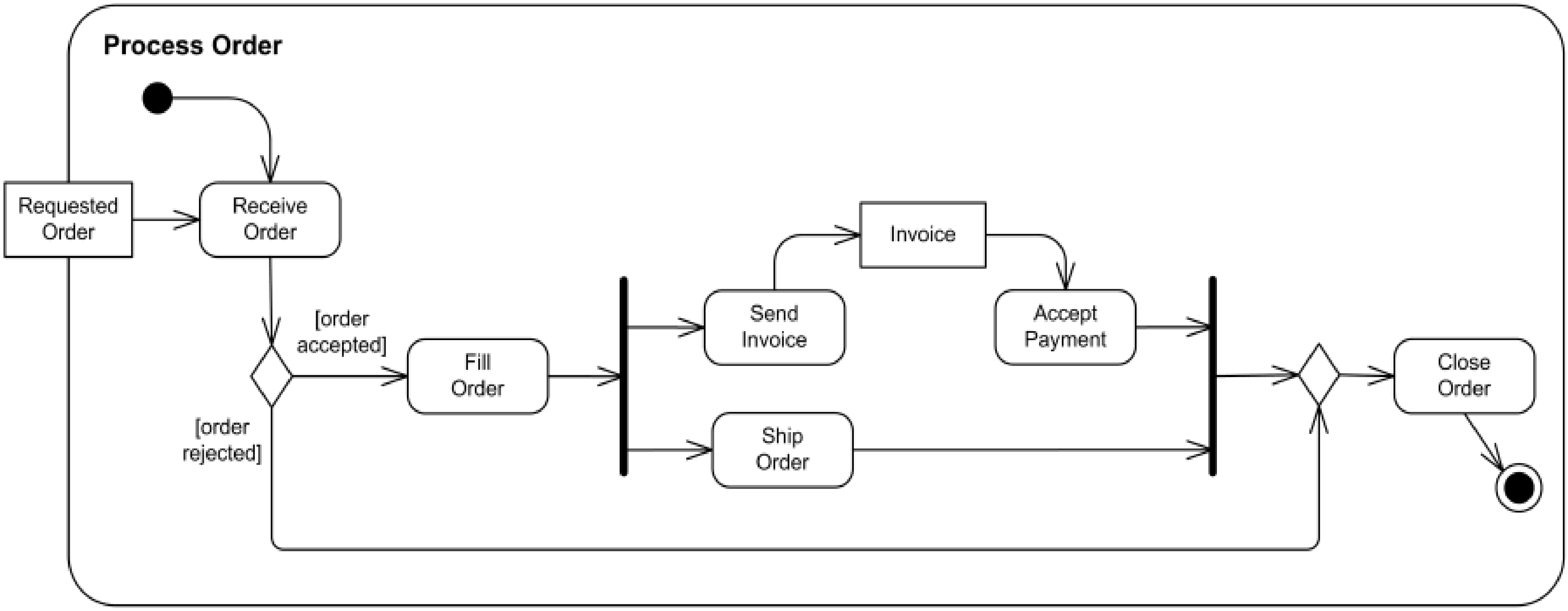
Note some states do not have (nor need) a name,
but need further details

Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions.

Activity diagrams may be regarded as a form of flowchart.

Activity Diagram


Process Order
Requested Order
Receive Order
[order accepted]
[order rejected]
Fill Order
Send Invoice
Invoice
Accept Payment
Ship Order
Close Order