

## 3.4 Activity Diagrams

Subject/Topic/Focus:

- Introduction to Activities

Summary:

- States
- Actions
- Transitions
- Branches, Merge
- Concurrency, Synchronization
- Swimlanes, Object Flow
- Signals

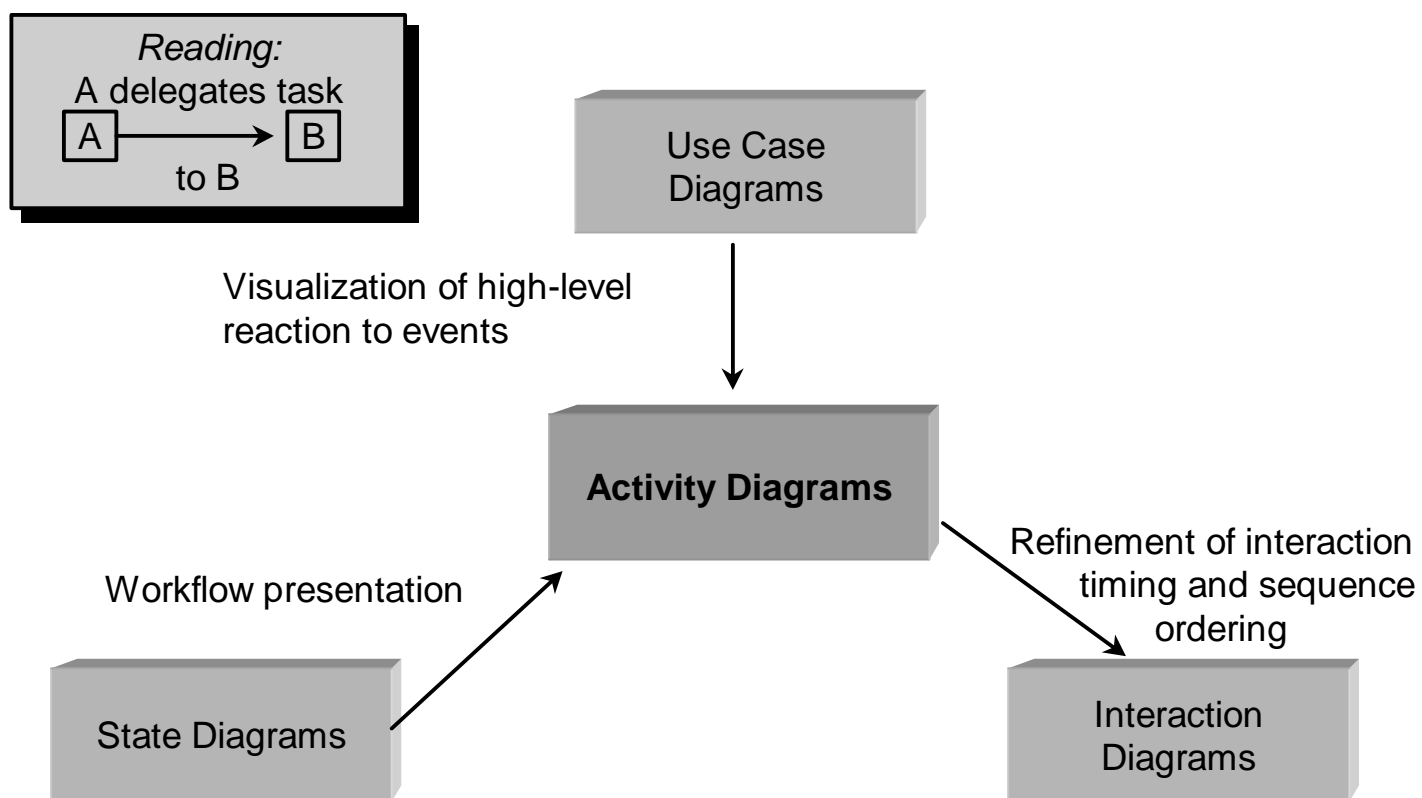
Literature:

- [Fowler97]
- [Booch98]

Activity Diagrams

3.4.1

## Role of Activity Diagrams in UML



Activity Diagrams

3.4.2

## Specification of Behavior in UML

---

### Specification models

- state machines
- activity diagrams

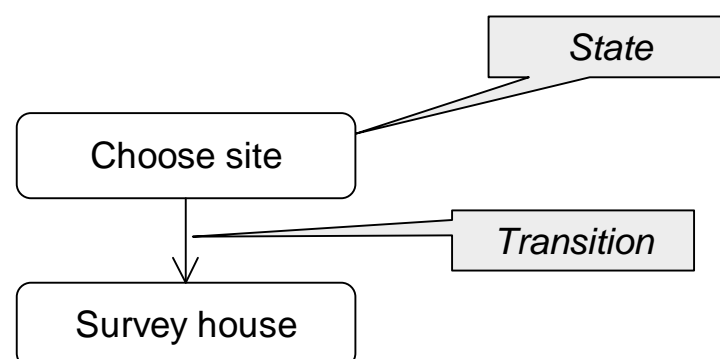
### Goals

- Specification of state changes of an object (or an interaction) triggered by an external event or a reserved signal.
- Definition of protocols, i. e., legal sequences of operations of a class or an interface.

## Activity Diagrams

---

An **activity diagram** shows an activity graph being a special case of a state machine. The states are action states or activity states and the transitions are fired (triggered) by the termination of activities.

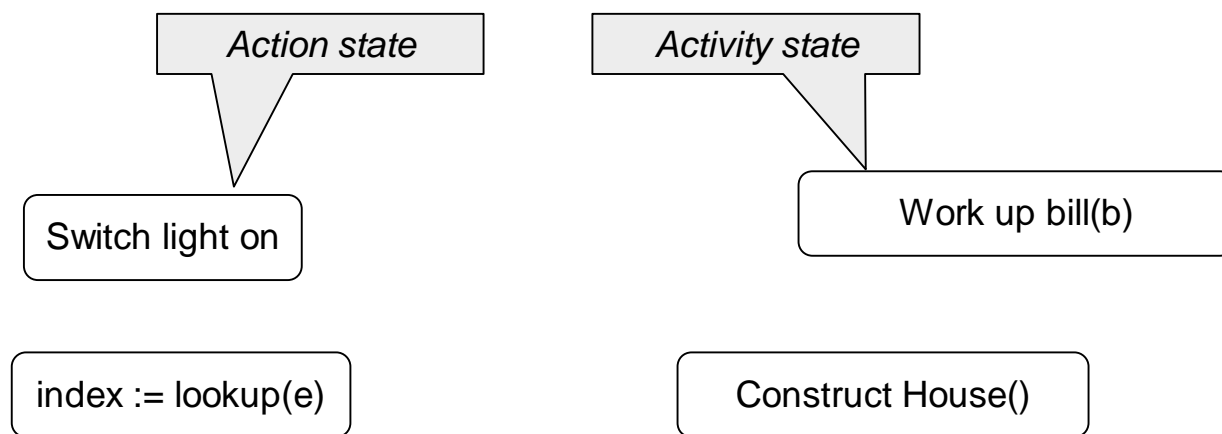


## Action States and Activity States

An **action state** describes an atomic state transition of a system without temporary state, e. g., an operation call or the calculation of a value.

An **activity state** describes an enduring activity, which can be interrupted and typically is described by a submachine (activity diagram).

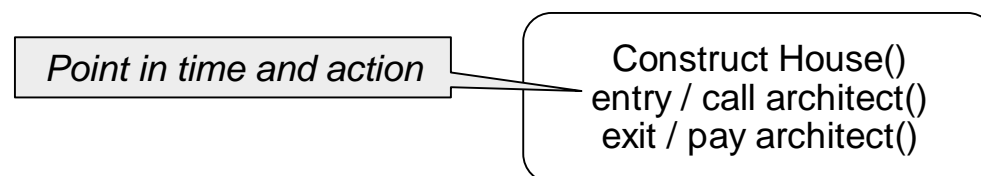
### Notation:



## Activity States: Actions

In activity states actions can be executed at the beginning or end of an activity.

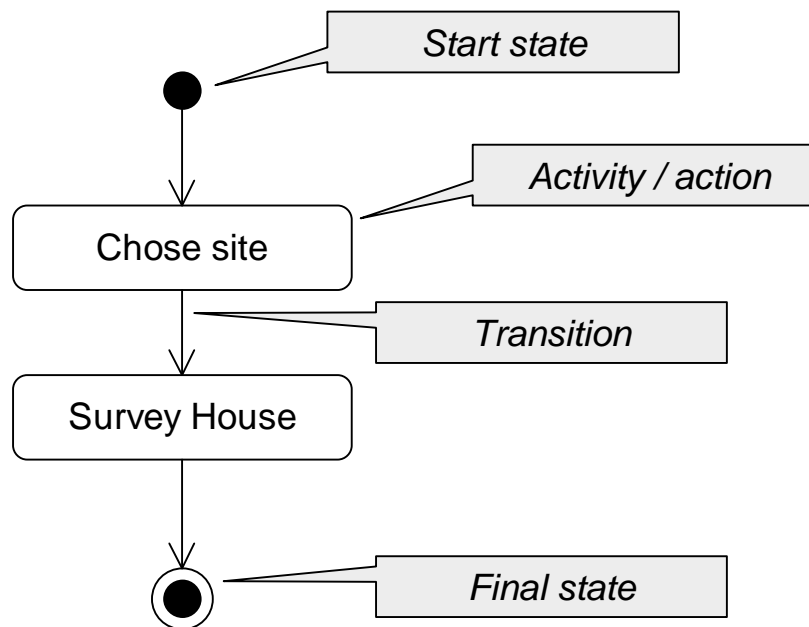
### Notation:



## Transitions

Transitions are the transients between action or activity states. A transition is executed when the action or activity is terminated.

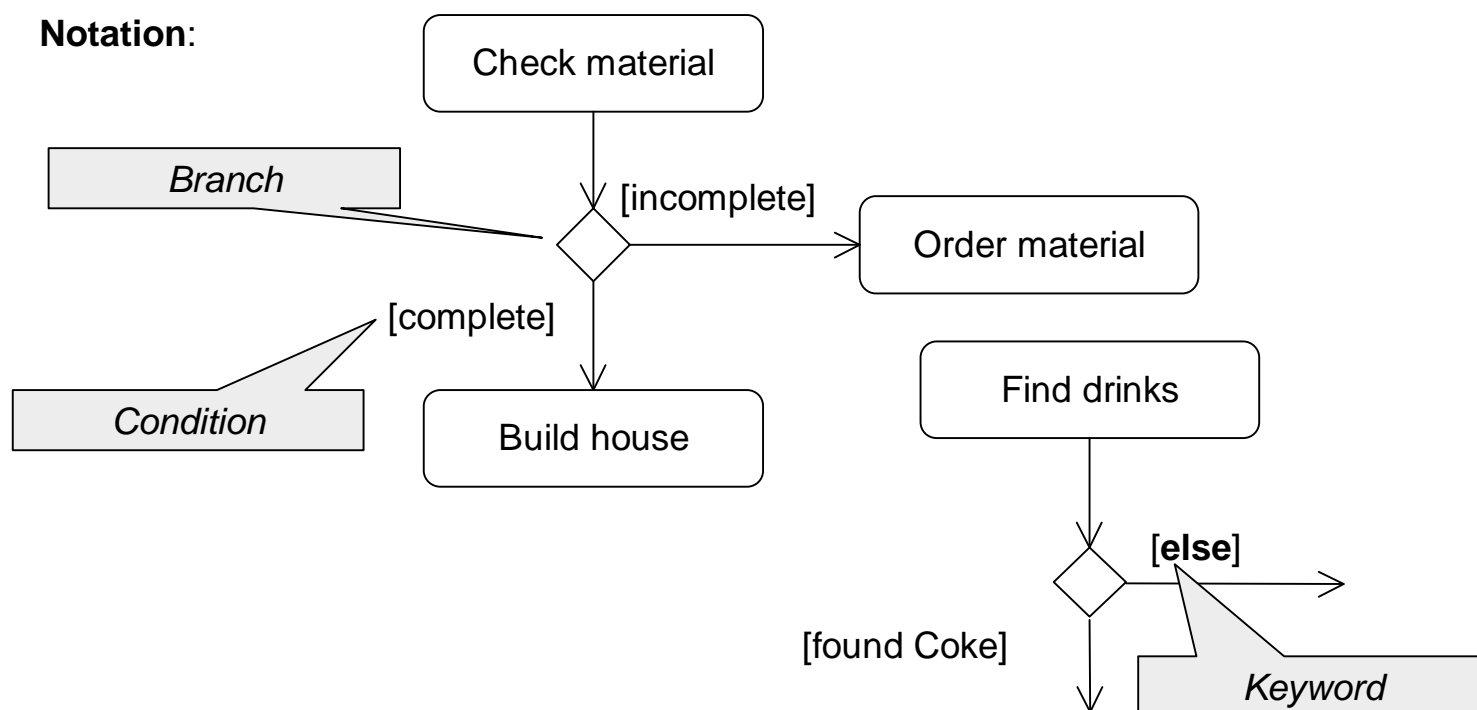
**Notation:**



## Branches

A **branch** appears, if a decision is to be made and depending on a condition several following actions/activities are to be initiated.

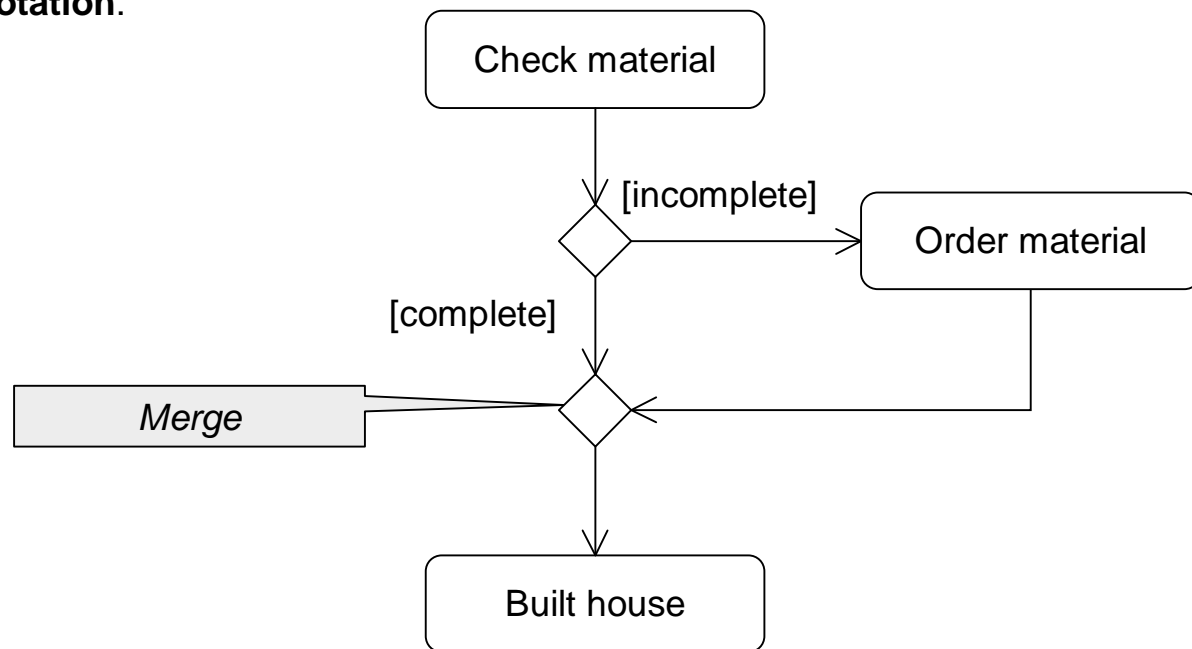
**Notation:**



## Merge

A **merge** merges optional branches in activity diagrams.

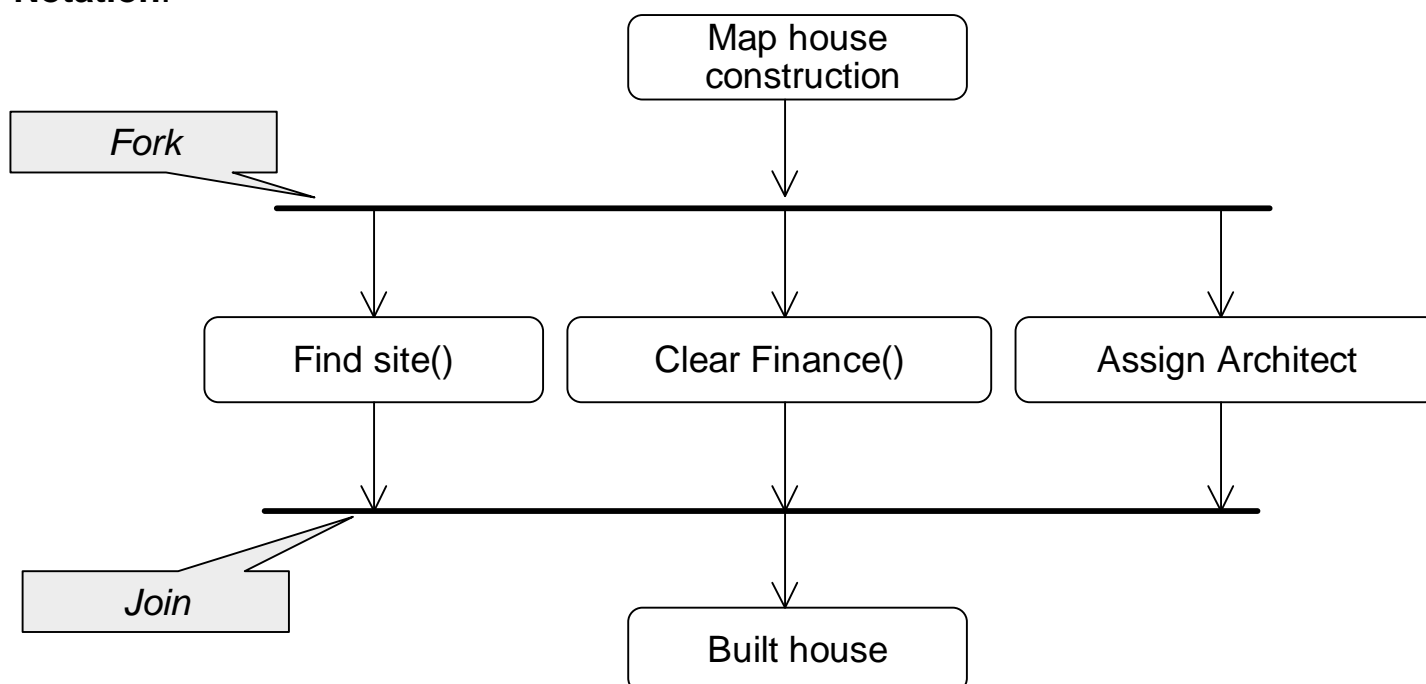
**Notation:**



## Concurrency

Actions and Activities can be executed concurrently and synchronized again.

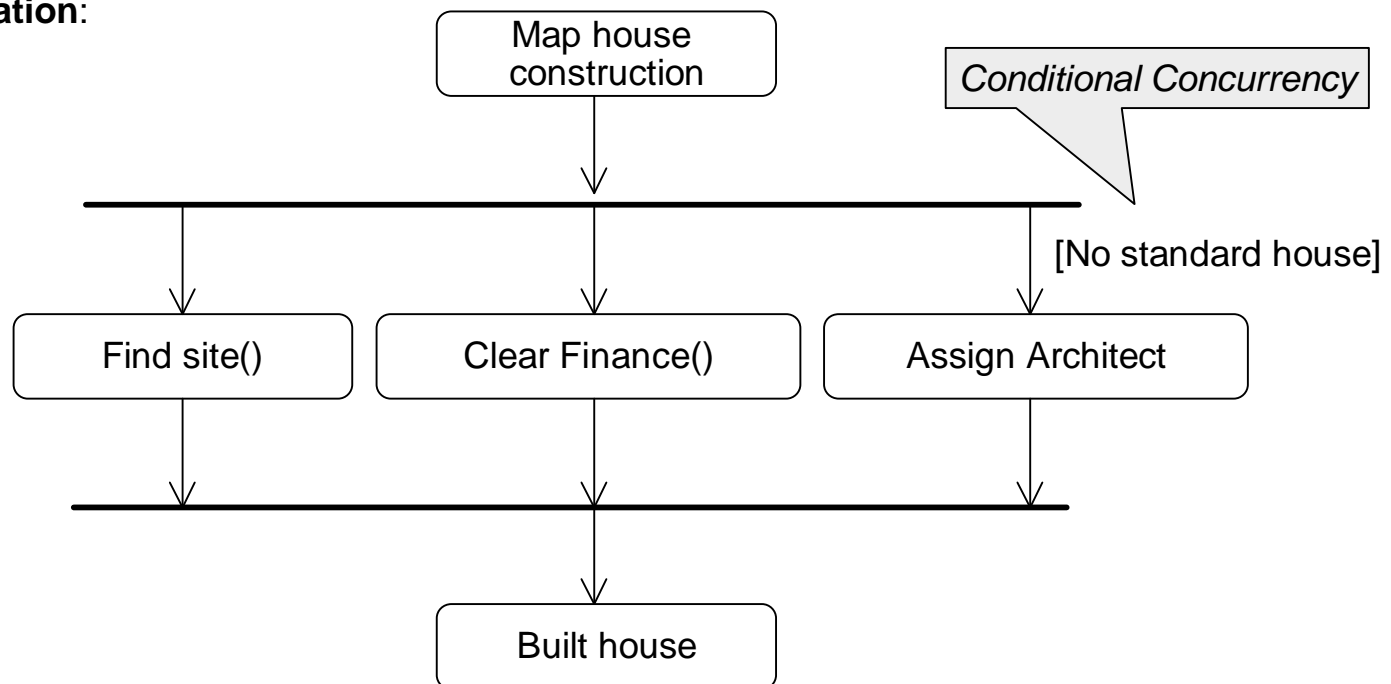
**Notation:**



## Conditional Concurrency

A concurrent sequence can only pass through on a special condition.

**Notation:**



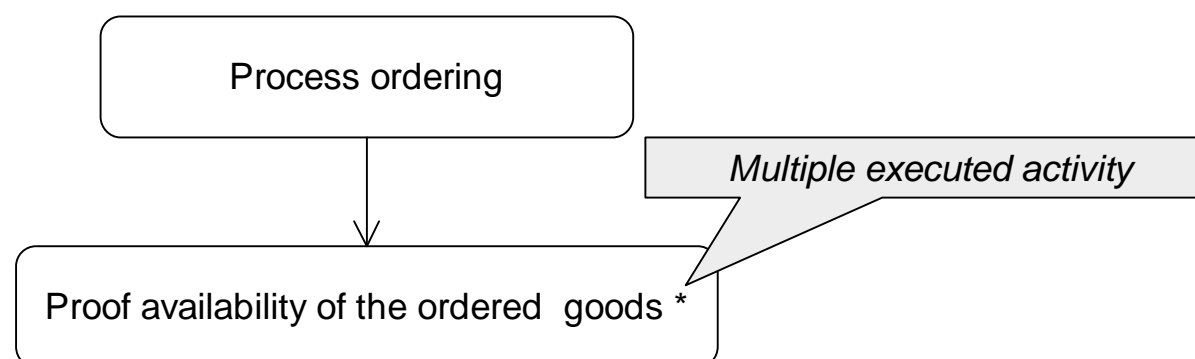
Activity Diagrams

3.4.11

## Dynamic Concurrency

An activity is executed concurrently multiple times where the number of concurrent activities depends on a list of arguments.

**Notation:**



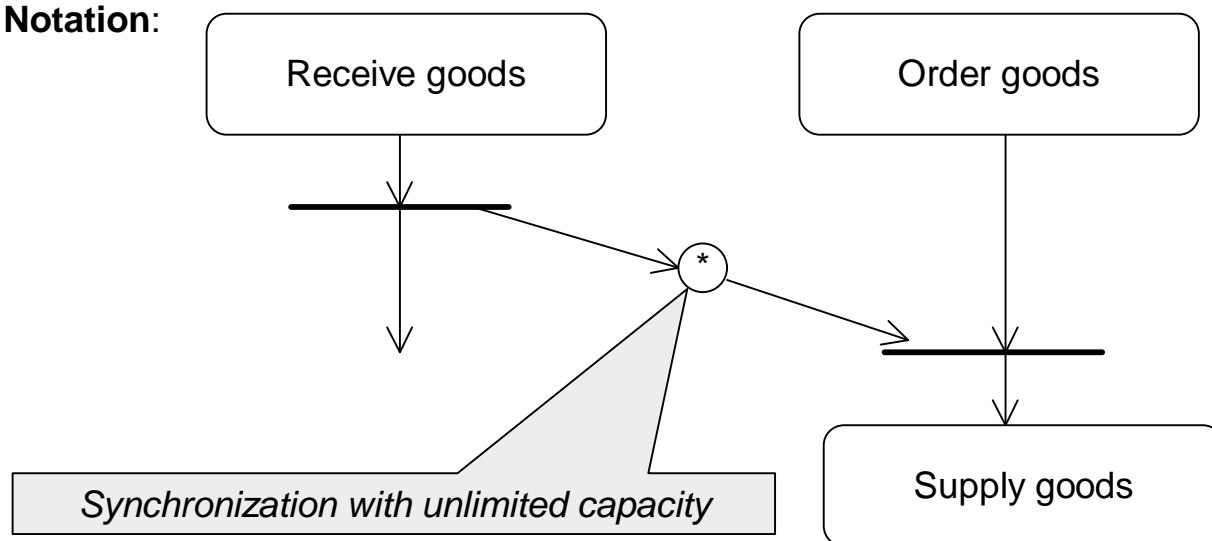
Activity Diagrams

3.4.12

## Synchronization of Concurrent Activities

Concurrent activities can be synchronized by a synchronization state. The state contains implicit a counter representing the number of waiting activities. The counter can be limited or unlimited.

**Notation:**



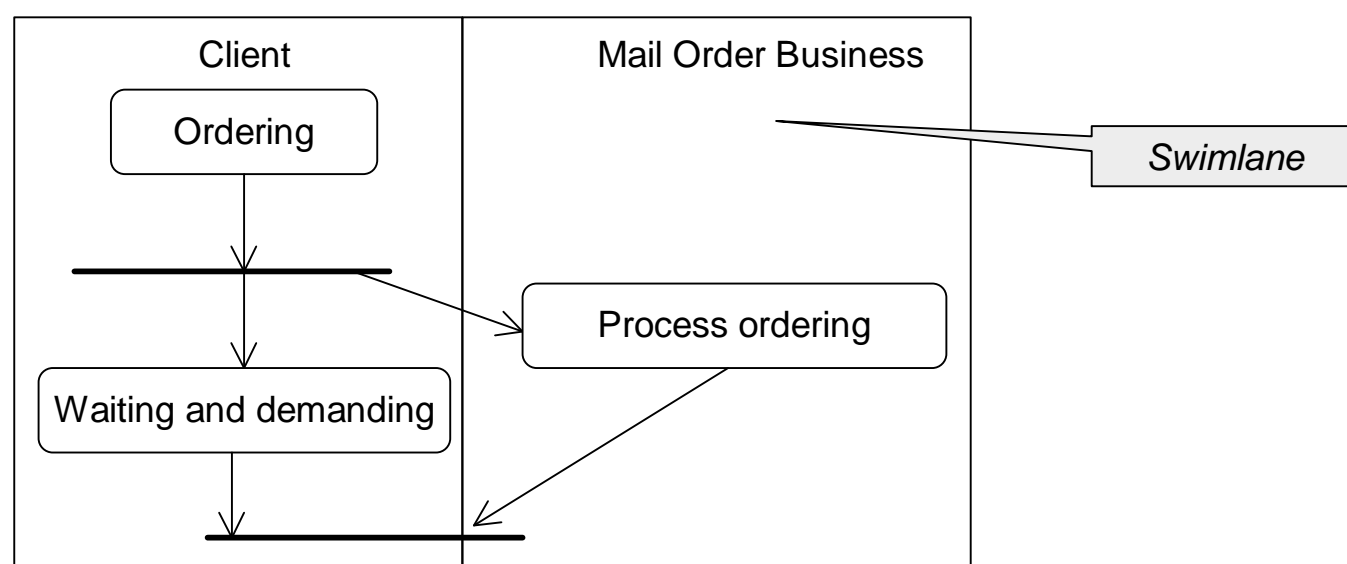
Activity Diagrams

3.4.13

## Swimlanes

A **swimlane** shows the actions and activities being executed by a unit, an object or a class, mostly concurrent to other actions/activities.

**Notation:**



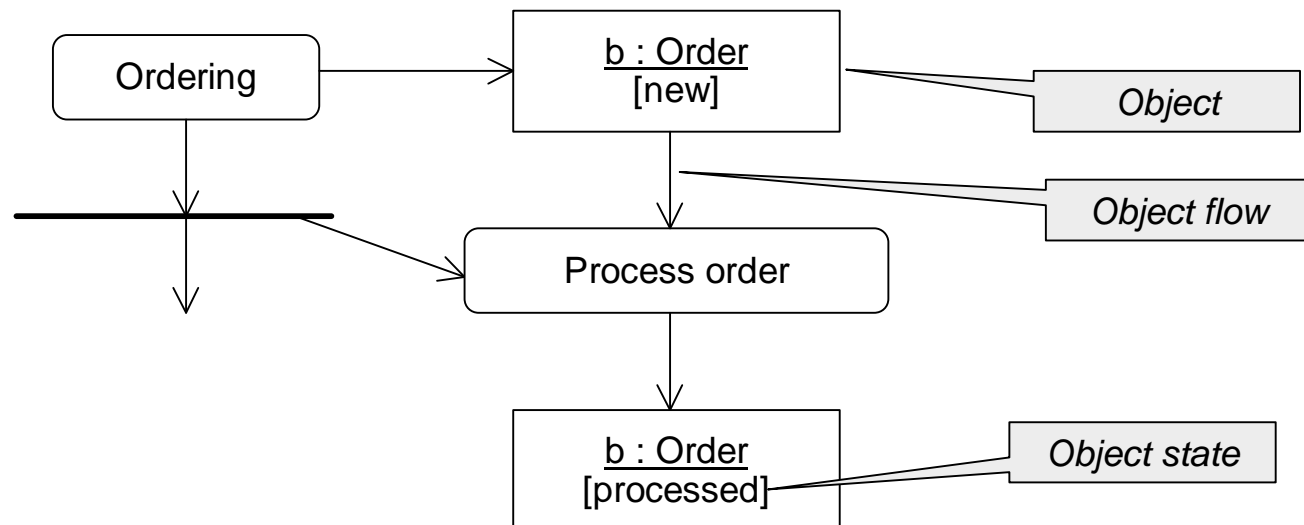
Activity Diagrams

3.4.14

## Object Flow

An **object flow** shows objects with their states being generated or used by actions or activities in activity diagrams.

**Notation:**



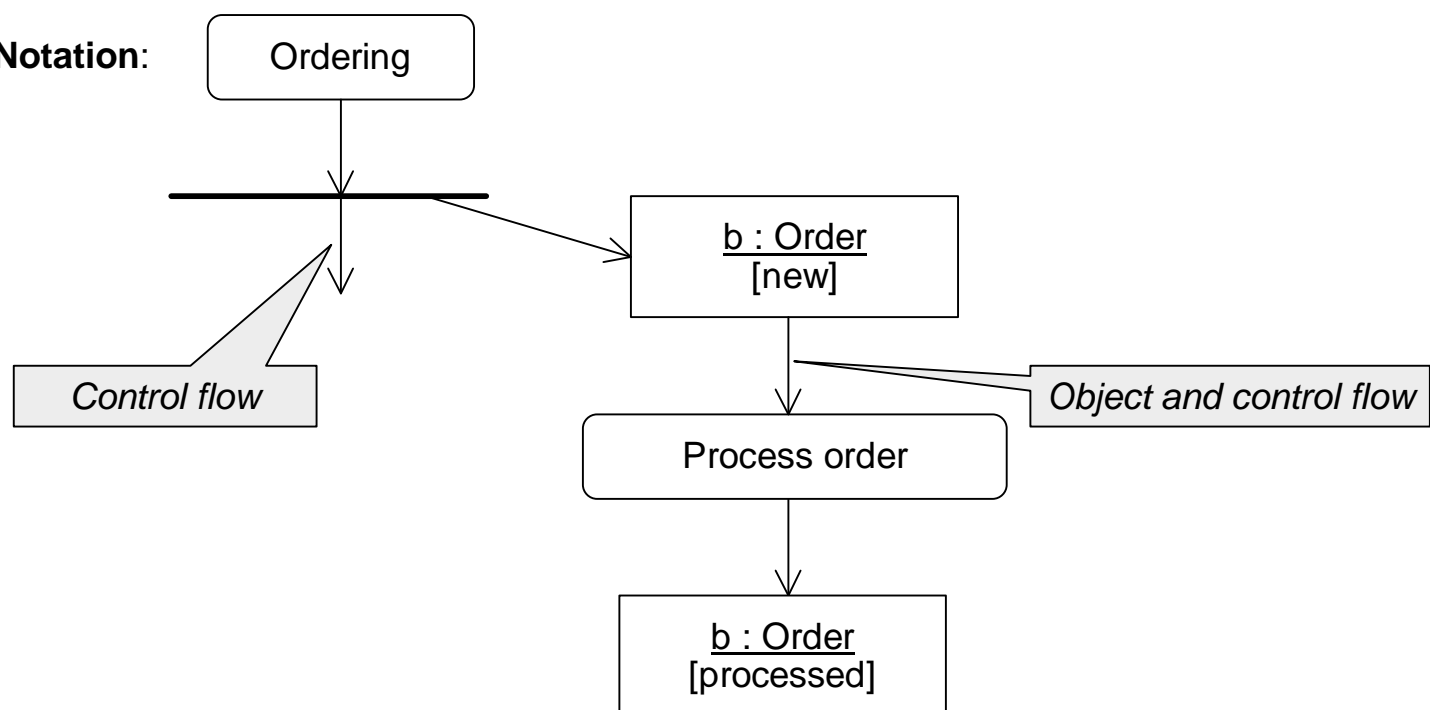
Activity Diagrams

3.4.15

## Object and Control Flow

Object flows can be merged with control flows.

**Notation:**



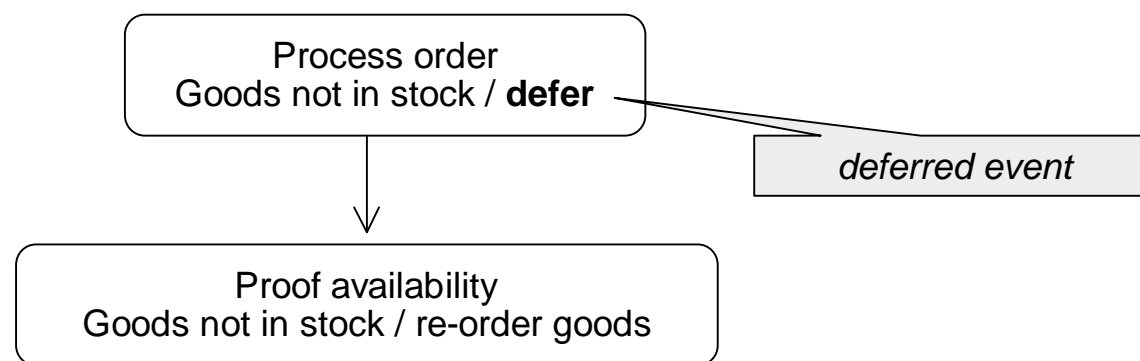
Activity Diagrams

3.4.16

## Deferred Events

Events which do not lead to a reaction during an activity but should not get lost are **deferred**. They can then lead in a following state to a reaction. Events not being deferred for which no reaction is defined expire.

**Notation:**



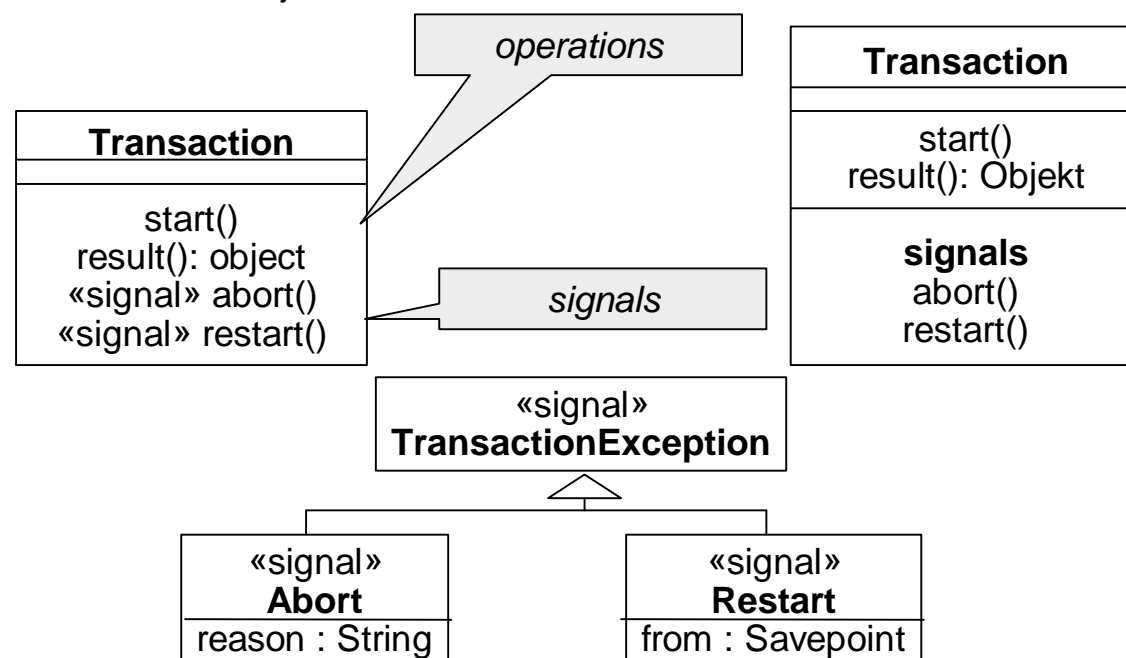
Activity Diagrams

3.4.17

## Signals and Operations *Revisited*

A **signal** is an asynchronous communication form between objects without return values while an **operation** represents a synchronous communication form permitting return values. Signals are sent, operations are called. Signals and operations are events triggering state transitions in objects.

**Notation:**



Activity Diagrams

3.4.18

## Sending and Receiving Signals

Signals can be sent during a sequence and objects can wait for a signal.

**Notation:**

