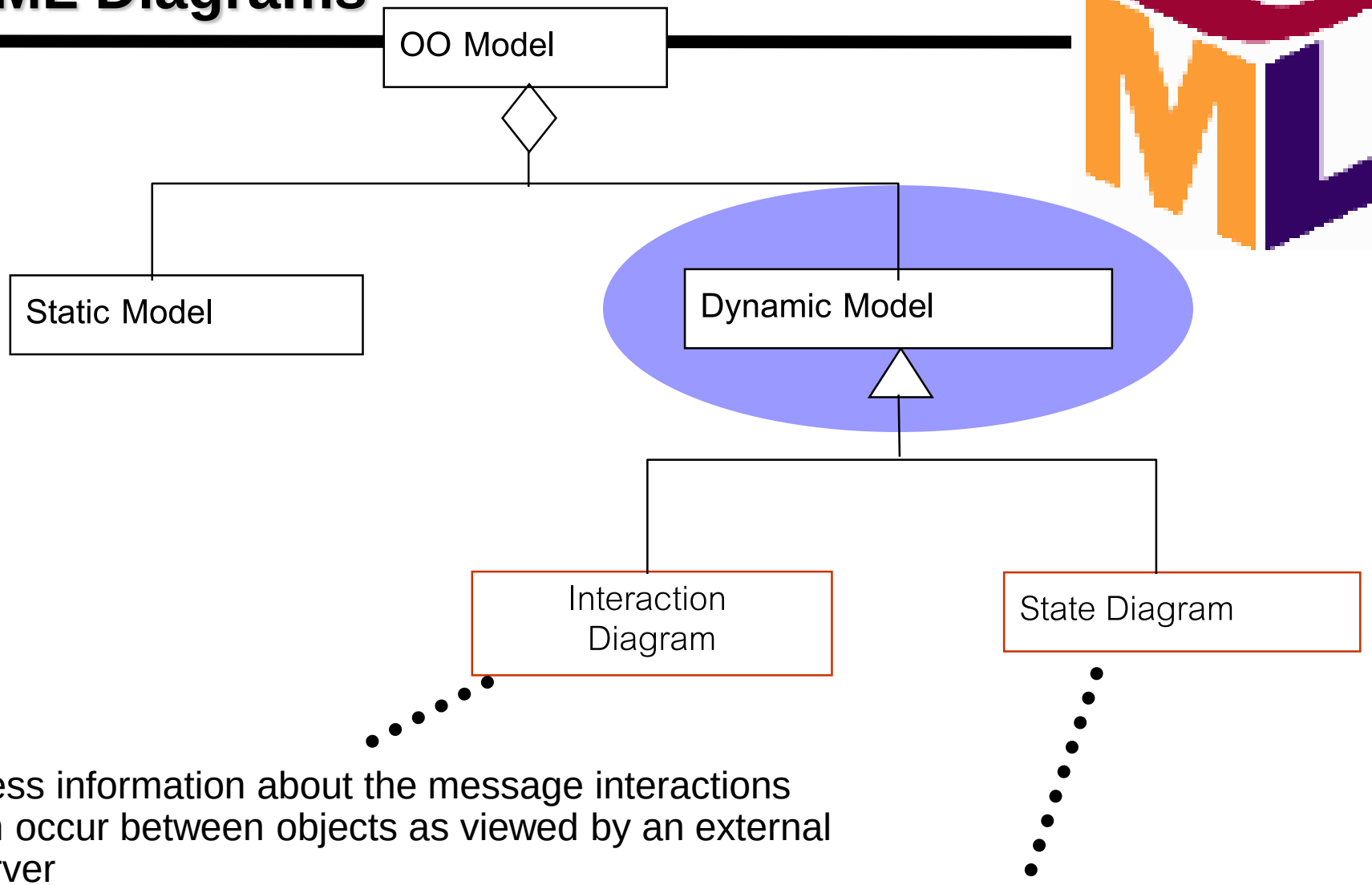
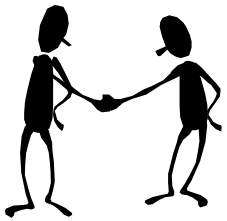
UML Diagrams



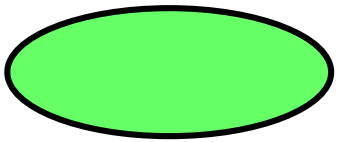
Express information about the message interactions which occur between objects as viewed by an external observer

Express information about the internal state changes which occur at objects of some particular class

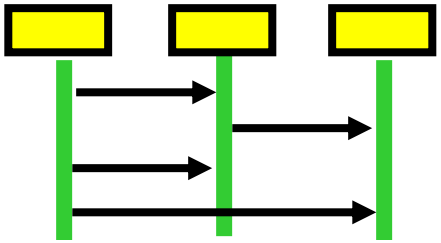
INTERACTION DIAGRAM - *Dynamic Model*



- Describes how groups of objects in the system collaborate in some behavior
- Shows a pattern of interaction among objects



Use case

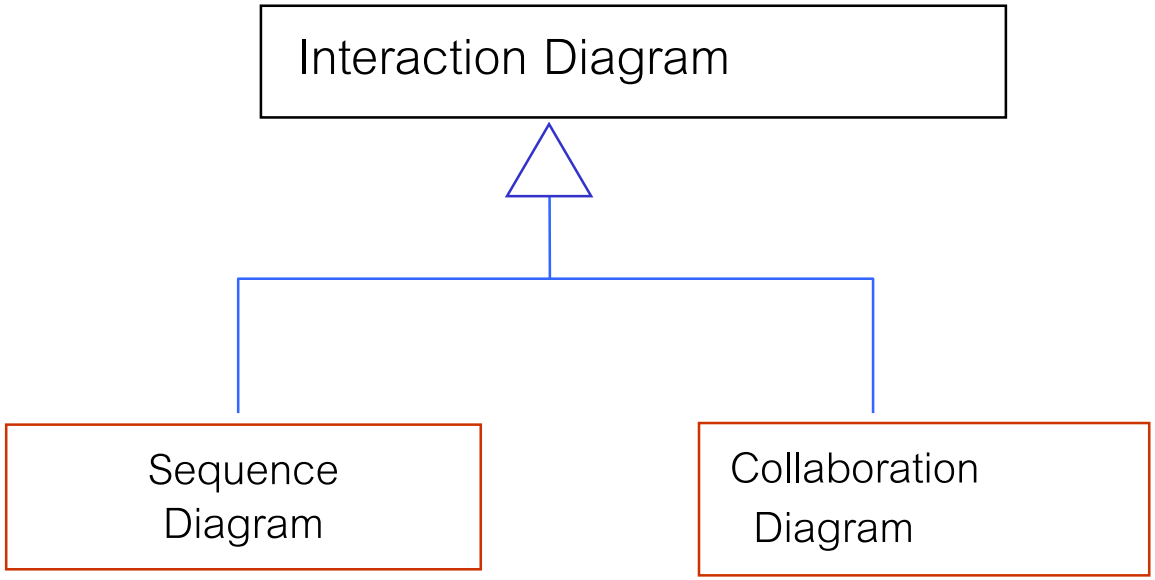


- Typically,
  an interaction diagram captures the **behavior of a *single use case***.

- The diagram shows a number of *example* ***objects*** and the ***messages*** that are passed between these objects within the use case.

There are two kinds of interaction diagrams:

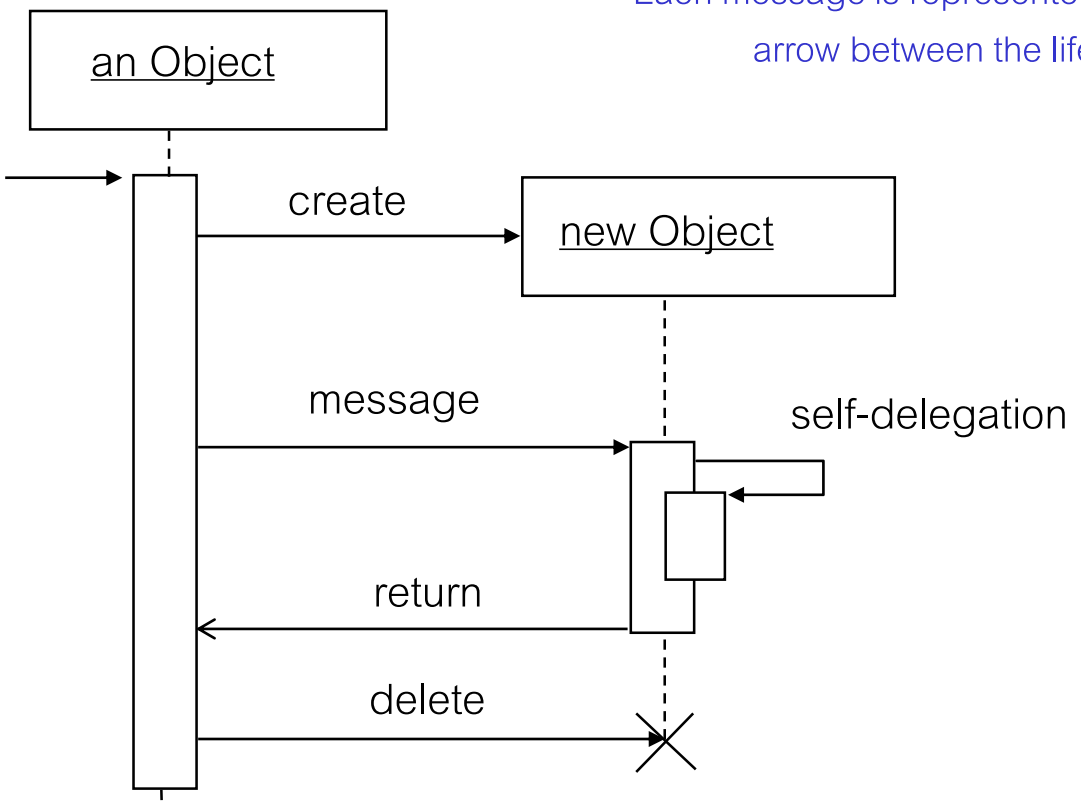
- **Sequence Diagram**
- **Collaboration Diagram**



SEQUENCE DIAGRAM

Shows an interaction arranged in time sequence

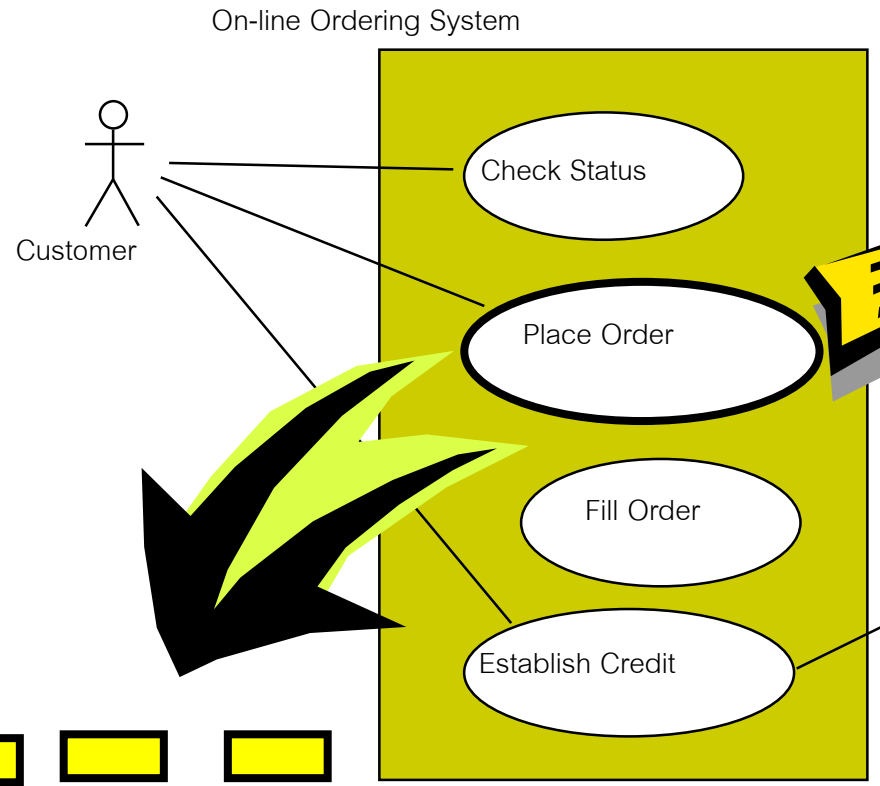
Each message is represented by an arrow between the lifelines of two objects.



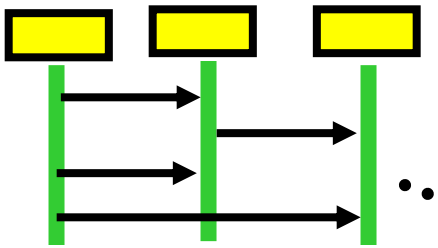
object's ***lifeline*** *(representing the object's life during the interaction)*

The description of a use case specifies
(from a point of view of *outside* the system)

"*What* the system should do".



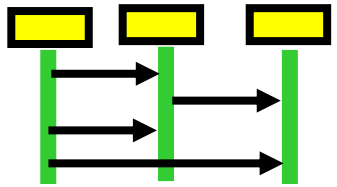
The Description of a Use Case



A sequence diagram captures
"**interaction among objects**"
(*inside* the system) in a use case.

I.e., "*How* the system should do it".

*From the description of a use case, we now think about the **interaction among objects (inside the system)***

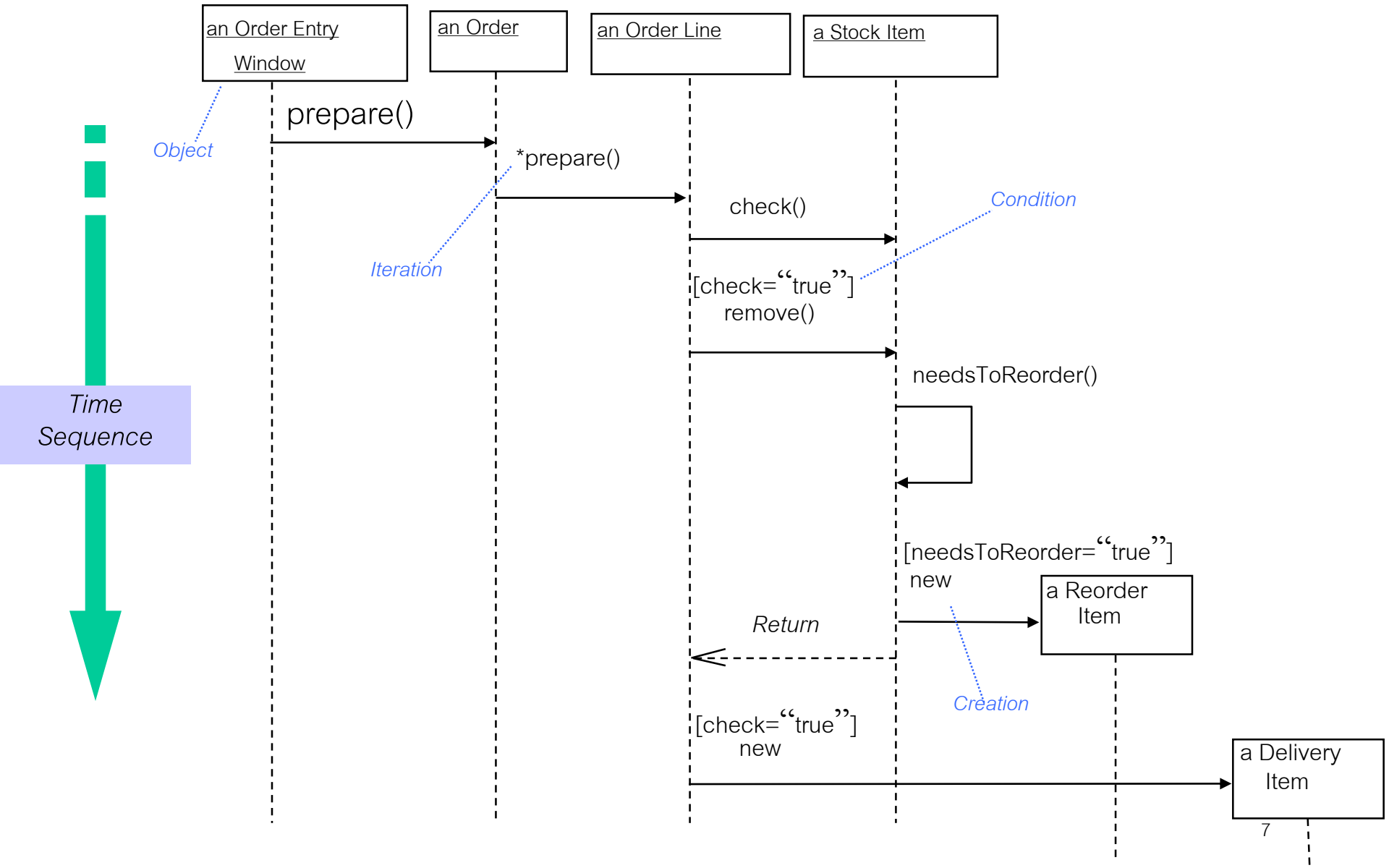


Consider the *use case* "place order" that exhibits the following behavior:

- The Order Entry window sends a "prepare" message to an Order.
- The Order then sends "prepare" to each Order Line on the Order.
- Each Order Line checks the given Stock Item.
  - If this check return "true", then
    - The Order Line removes the appropriate quantity of Stock Item from stock. Now, if the quantity of Stock Item has fallen below the reorder level, that Stock Item requests a new Reorder Item.
    - The Order Line requests a new delivery.

*We then describe **this interaction** using a sequence diagram.*

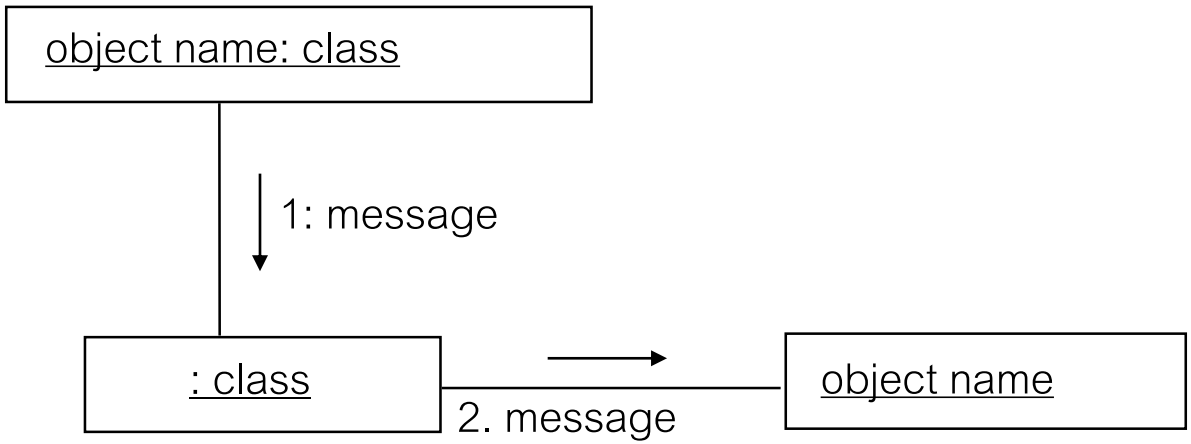
Example


an Order Entry Window
an Order
an Order Line
a Stock Item
Object
prepare()
*prepare()
Iteration
check()
Condition
[check="true"]
remove()
needsToReorder()
Time Sequence
[needsToReorder="true"]
new
a Reorder Item
Return
Creation
[check="true"]
new
a Delivery Item

COLLABORATION DIAGRAM

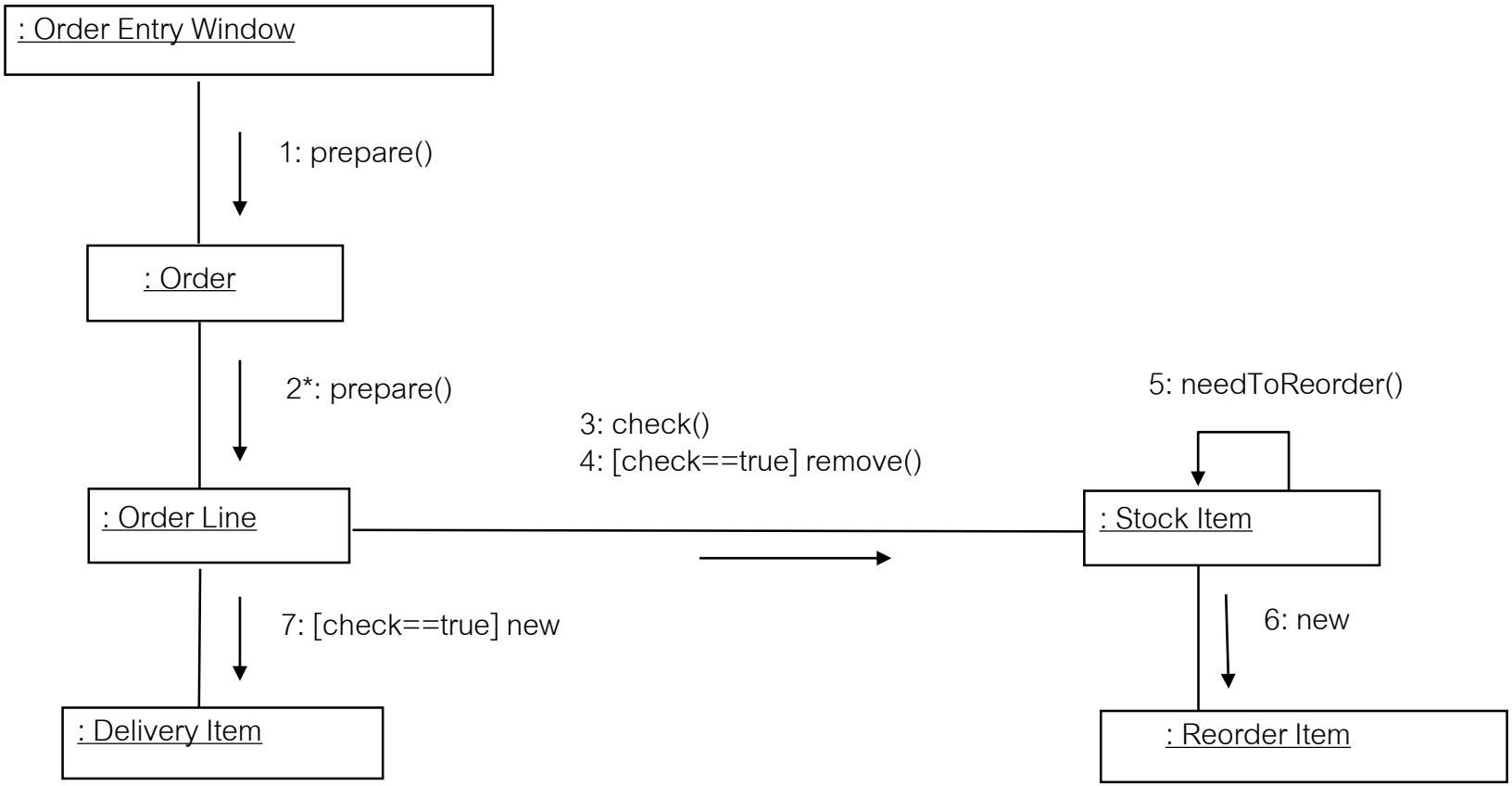
An alternative representation of an interaction

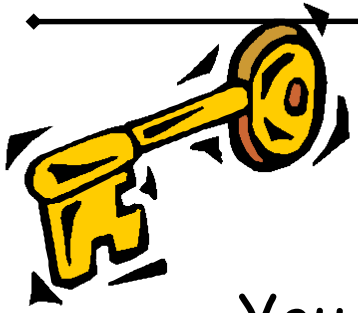
The sequence of messages passing is indicated by numbering the messages.



Sequence diagrams and collaboration diagrams express similar information but show it in different ways.

Example


: Order Entry Window
1: prepare()
: Order
2*: prepare()
3: check()
4: [check==true] remove()
5: needToReorder()
: Order Line
: Stock Item
7: [check==true] new
6: new
: Delivery Item
: Reorder Item



Interobject behavior

You should use **interaction diagrams**

when you want to look at the behavior of

"*several objects* within a *single use case*".

If you want to look at the behavior of

"*a single object* across *many use cases*",

use a **state-transition diagram**.

Intraobject behavior

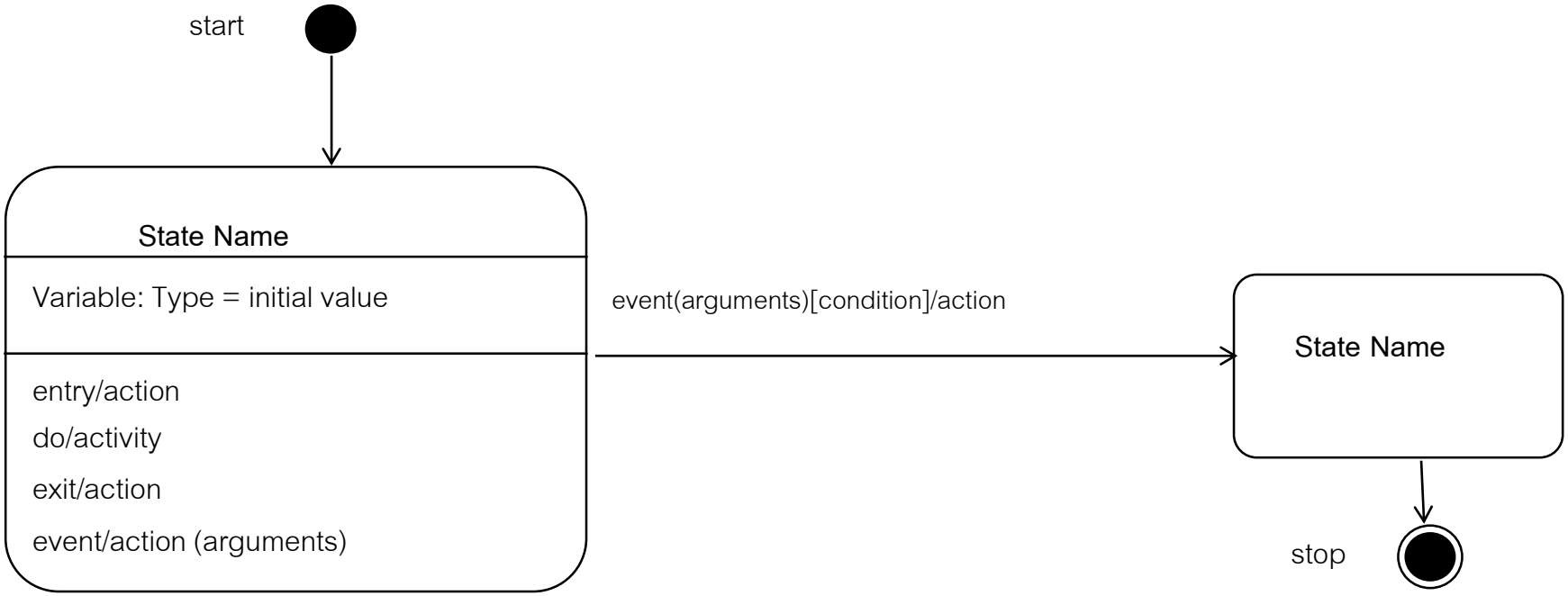
STATE (-TRANSITION) DIAGRAM

- Dynamic Model

Mô tả hành vi của một đối tượng (lớp):

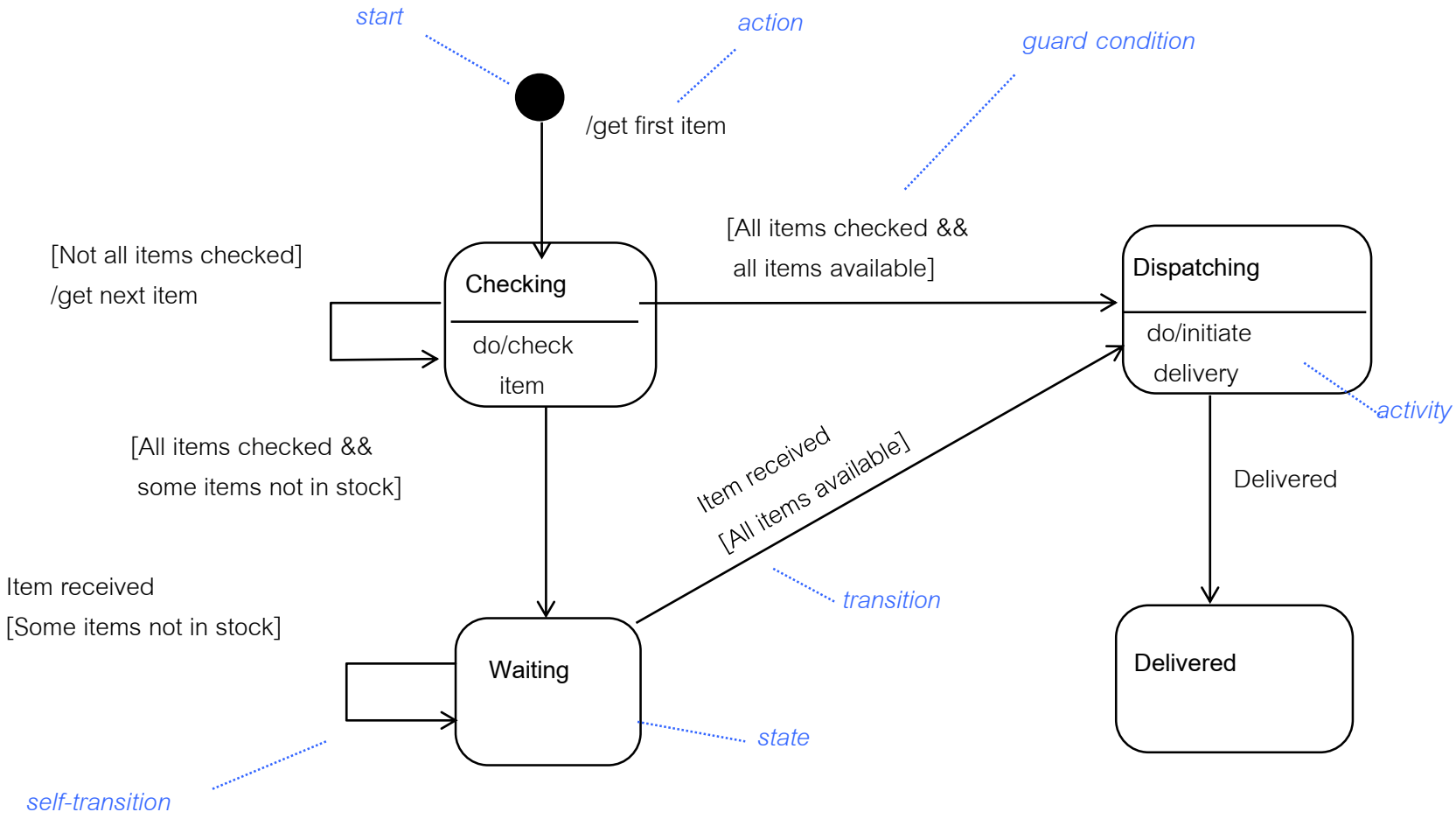
+ Đi qua nhiều use cases
+ Đáp ứng với thông điệp gửi từ bên ngoài vào (external message)

• Mô tả tất cả các trạng thái có thể có của một đối tượng và sự thay đổi trạng thái của nó khi có các sự kiện (events) tác động vào.



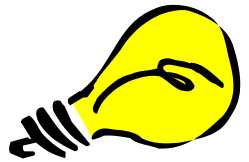
At any instant in time, an object is in a particular state.

Example: *a state diagram for an order*



Source: Martin Fowler (1997), *UML Distilled*, Addison-Wesley.

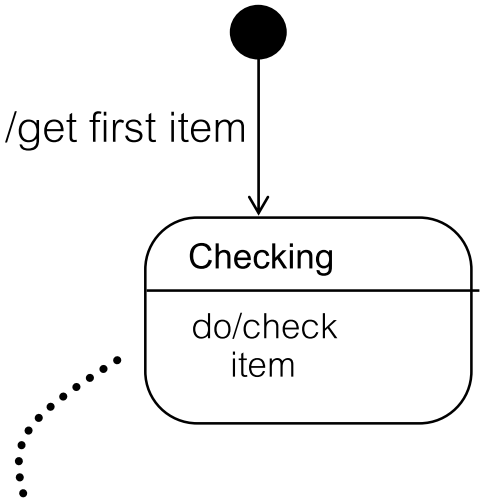
Explanation for the previous state diagram ...



The syntax for a transition label has three parts, all of which are optional:

Event **[*Condition*]** ***/ Action***.

Xét biểu đồ trạng thái ở trang trước:



Bắt đầu ở start point và chuyển (transition) vào trạng thái Checking.
This transition is labeled "/get first item".
Trong trường hợp này chỉ có hành động (action) "get first item"; không có sự kiện cũng như điều kiện
Once we perform that action, we enter the Checking state.

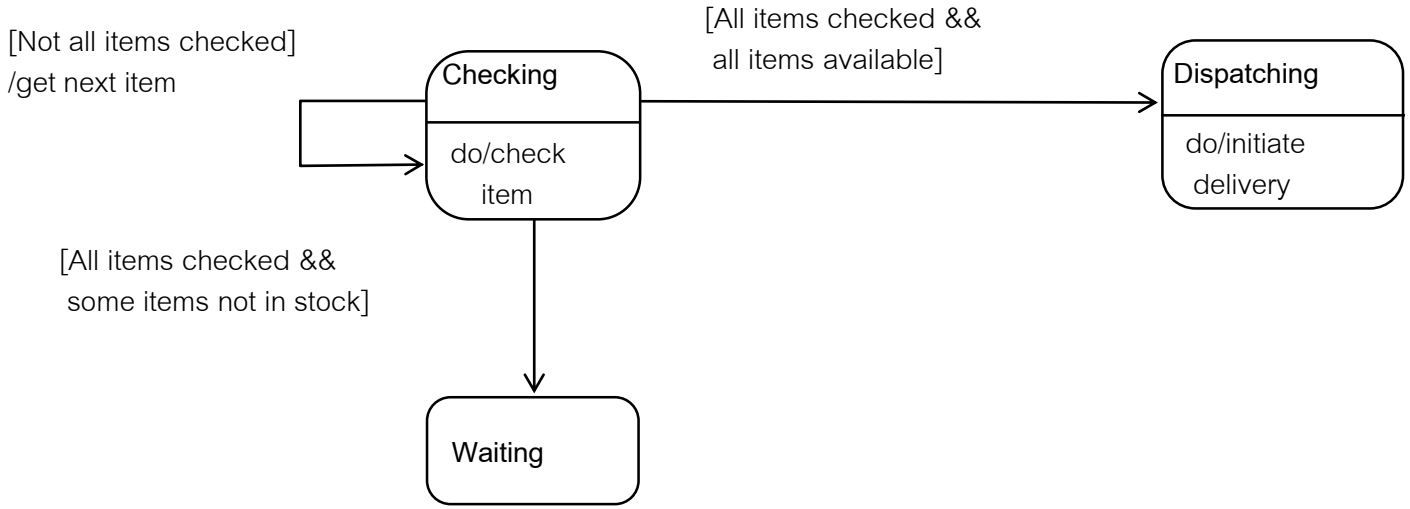
Trạng thái này có 1 hoạt động(activity) phối hợp với nó, biểu diễn bởi: *do / activity*.
In this case, the activity is "check item".

Chú ý: Sử dụng "action" cho transition và "activity" cho state. Mặc dù chúng đều là processes, nhưng biểu diễn ý nghĩa khác nhau.

+ Actions phối hợp với transitions và được xem như processes that occur **quickly and are not interruptible**.

+ Activities phối hợp với states; can take longer. An activity may be interrupted by some event.

Explanation …



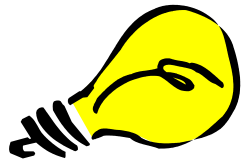
Khi 1 bước chuyển trạng thái không có sự kiện nào kết hợp với nó, điều đó có nghĩa là việc chuyển trạng thái sẽ xãy ra ngay khi hoạt động hòan thành.

Trong trường hợp này, ngay khi kết thúc **Checking**, 1 trong 3 transitions sẽ rời khỏi trạng thái Checking. Cả 3 đều có conditions, nhưng không có events.

Chỉ có 1 trong 3 bước chuyển trạng thái được thực hiện tại 1 thời đểm:

1. If we have not checked all items, we get the next item and return to the Checking state to check it.
2. If we have checked all items and they were all in stock, we transition to the Dispatching state.
3. If we have checked all items but all of them were in stock, we transition to the Waiting state.

Explanation …



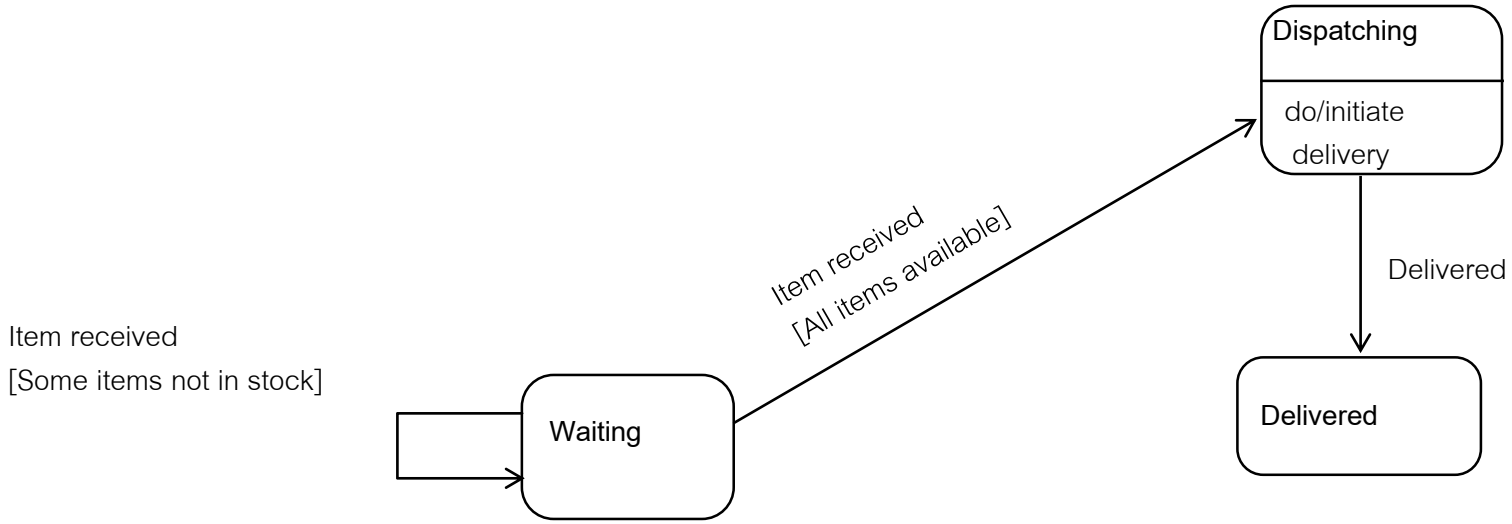
… Xem xét trạng thái **Waiting:**

There are no activities for this state, so the given order sits in this state waiting for an event.

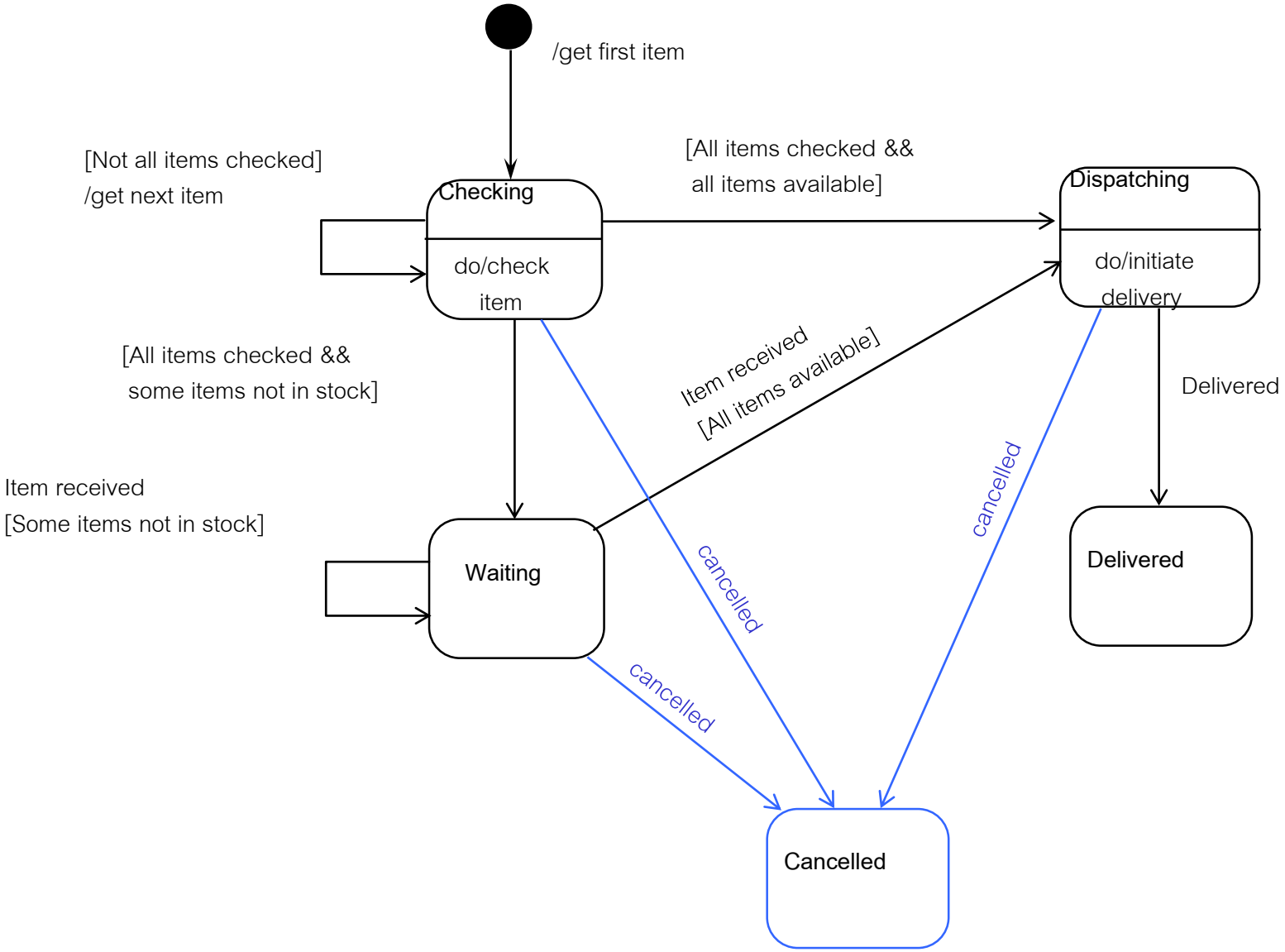
Both transitions out of the Waiting state are labeled with the **Item Received** event.

This means that the order waits until it detects this event.

At that point, it evaluates the conditions on the transitions and makes the appropriate transition (either to Dispatching or back to Waiting).

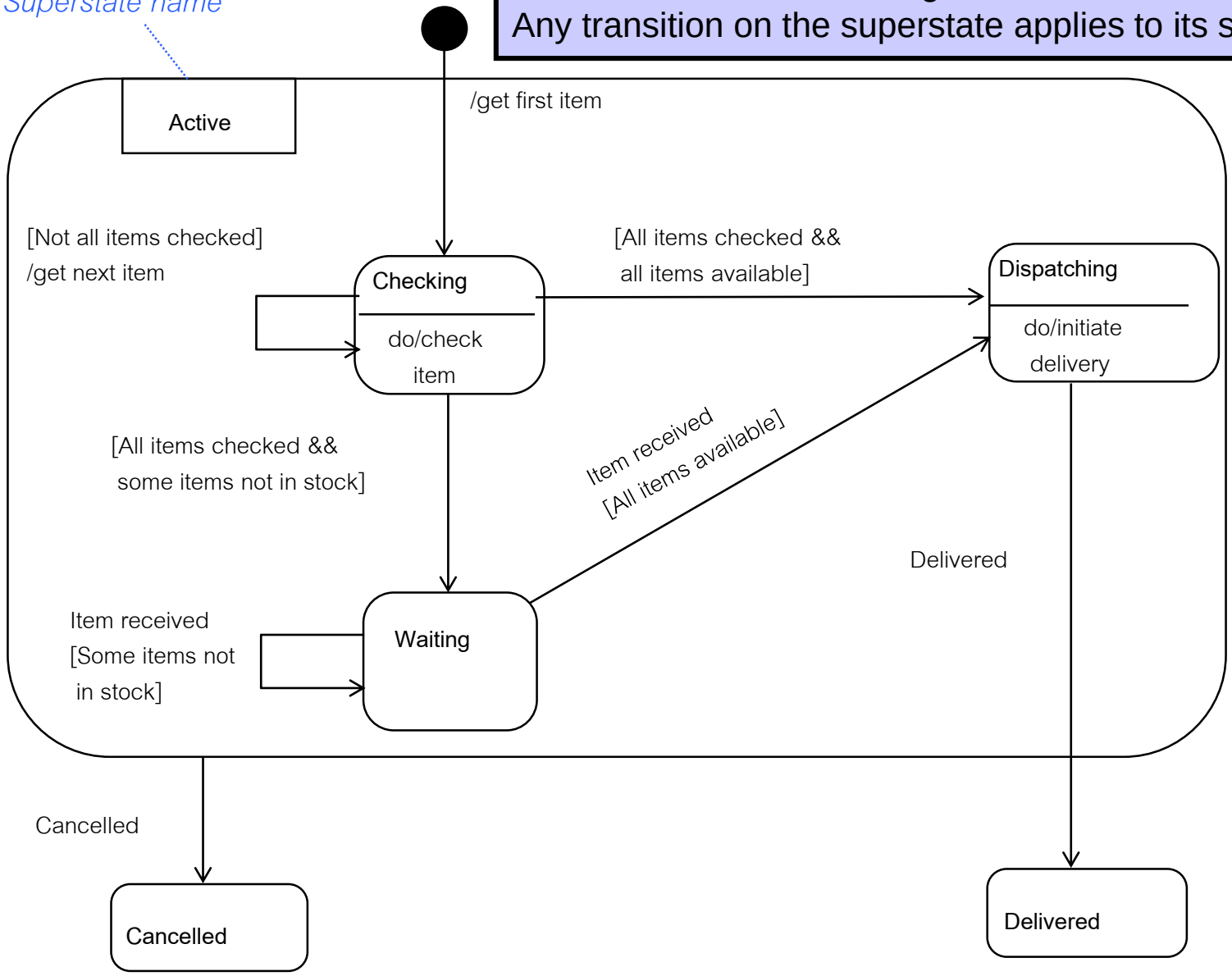


Now suppose that we want to be able to cancel an order at any point before it is delivered.

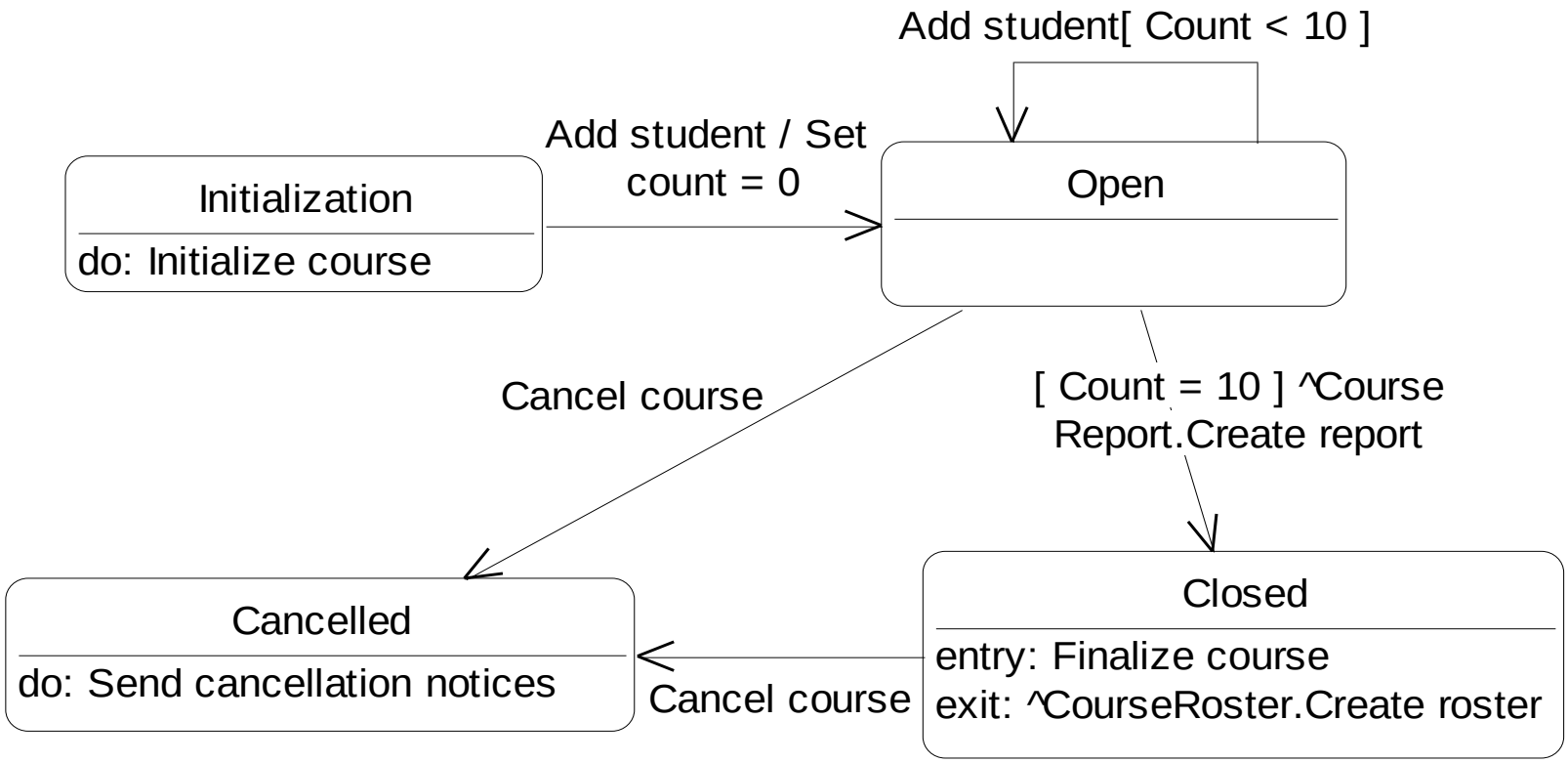


State Diagram with Superstates

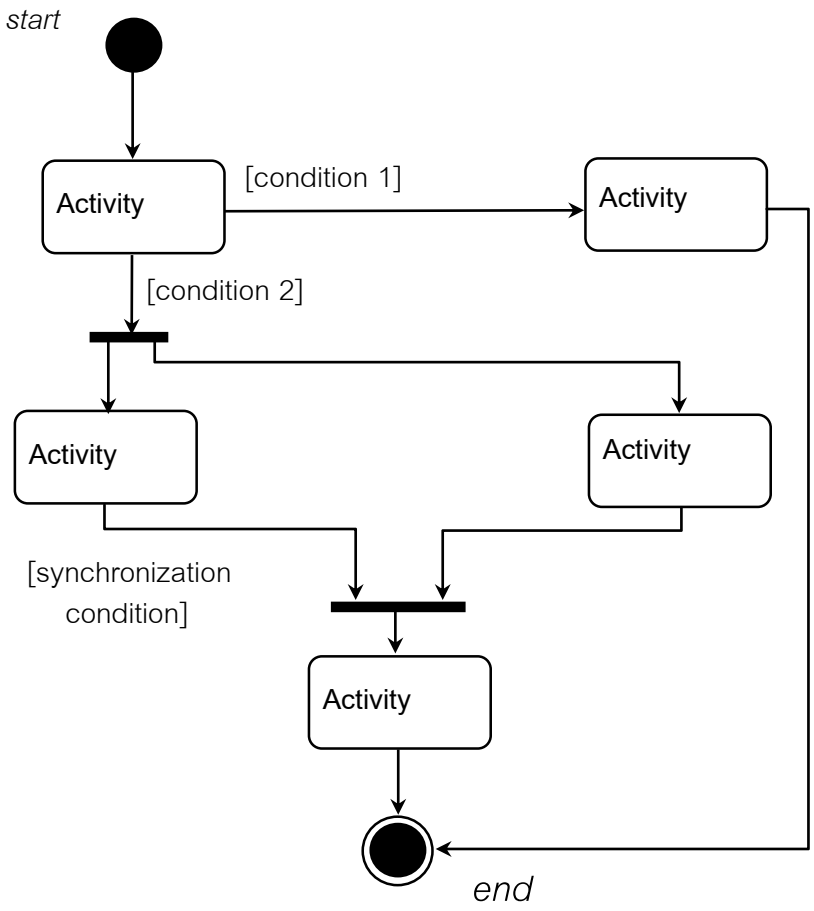
A useful alternative is to create a **superstate** of all three states and then draw a single transition from that. Any transition on the superstate applies to its substates.



State Diagram for class LopMonHoc



ACTIVITY DIAGRAM

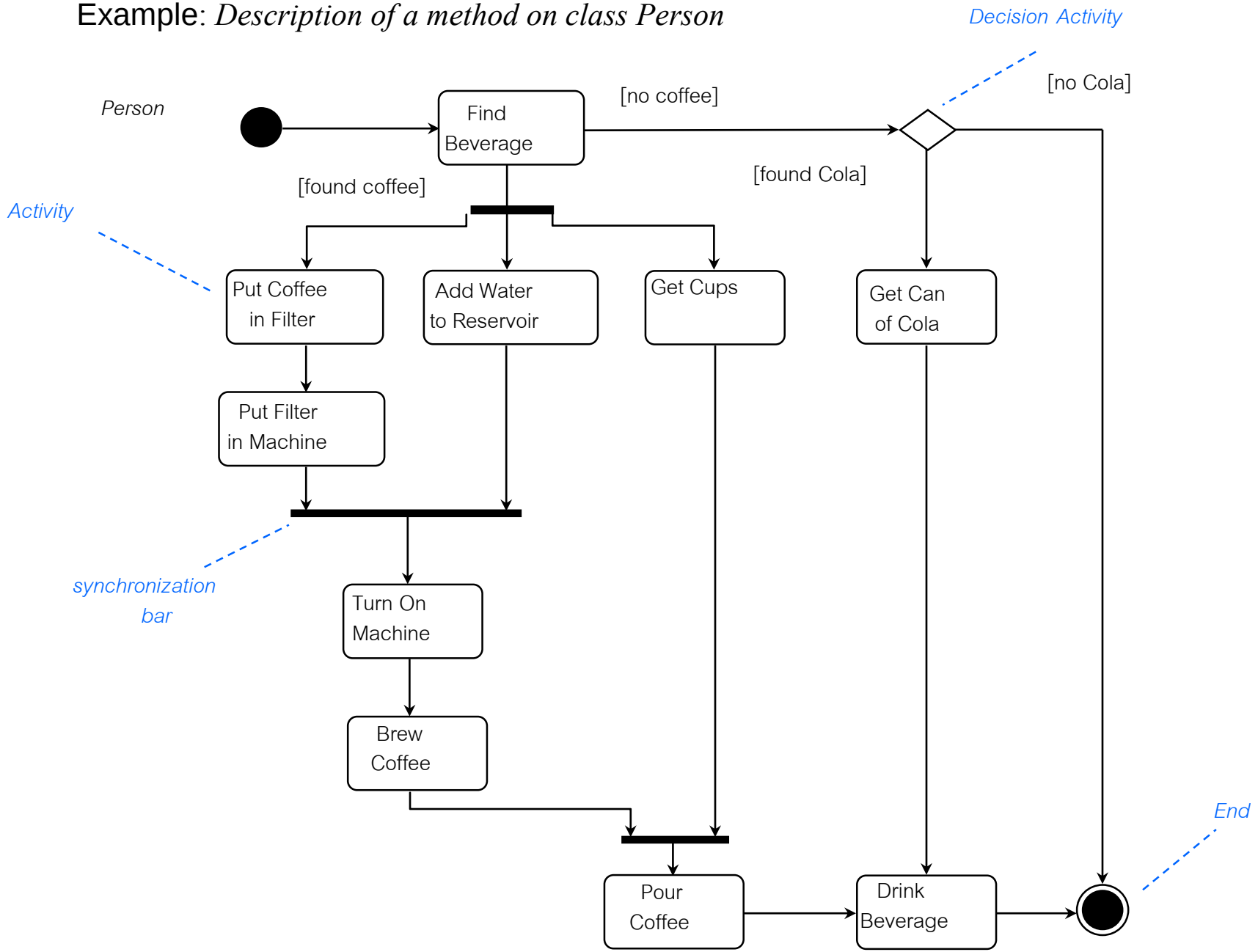


Biểu đồ hoạt động được gắn với một lớp, hoặc việc thực hiện một thao tác hoặc một use case.

Activity có thể là những công việc or Activity is a method on a class.

- Mô tả các hoạt động của 1 lớp. Tương tự như biểu đồ trạng thái, nhưng mô tả hành vi của 1 lớp trong việc đáp ứng đến những xữ lý bên trong hơn là những sự kiện bên ngoài (internal processing rather than external events).
- Có thể dùng để trình bày trạng thái của một thủ tục:
  - Mỗi trạng thái đại diện cho 1 bước trong thủ tục. Sự kiện thoát khỏi 1 trạng thái là việc hoàn tất 1 bước trong thủ tục
- Hoặc dùng để biểu diễn dòng công việc của: Use Cases, Operations
- Mục đích của biểu đồ này là tập trung trên dòng công việc (work flow) xữ lý bên trong (internal processing, ngược lại với biểu đồ trạng thái – tập trung trên external events).

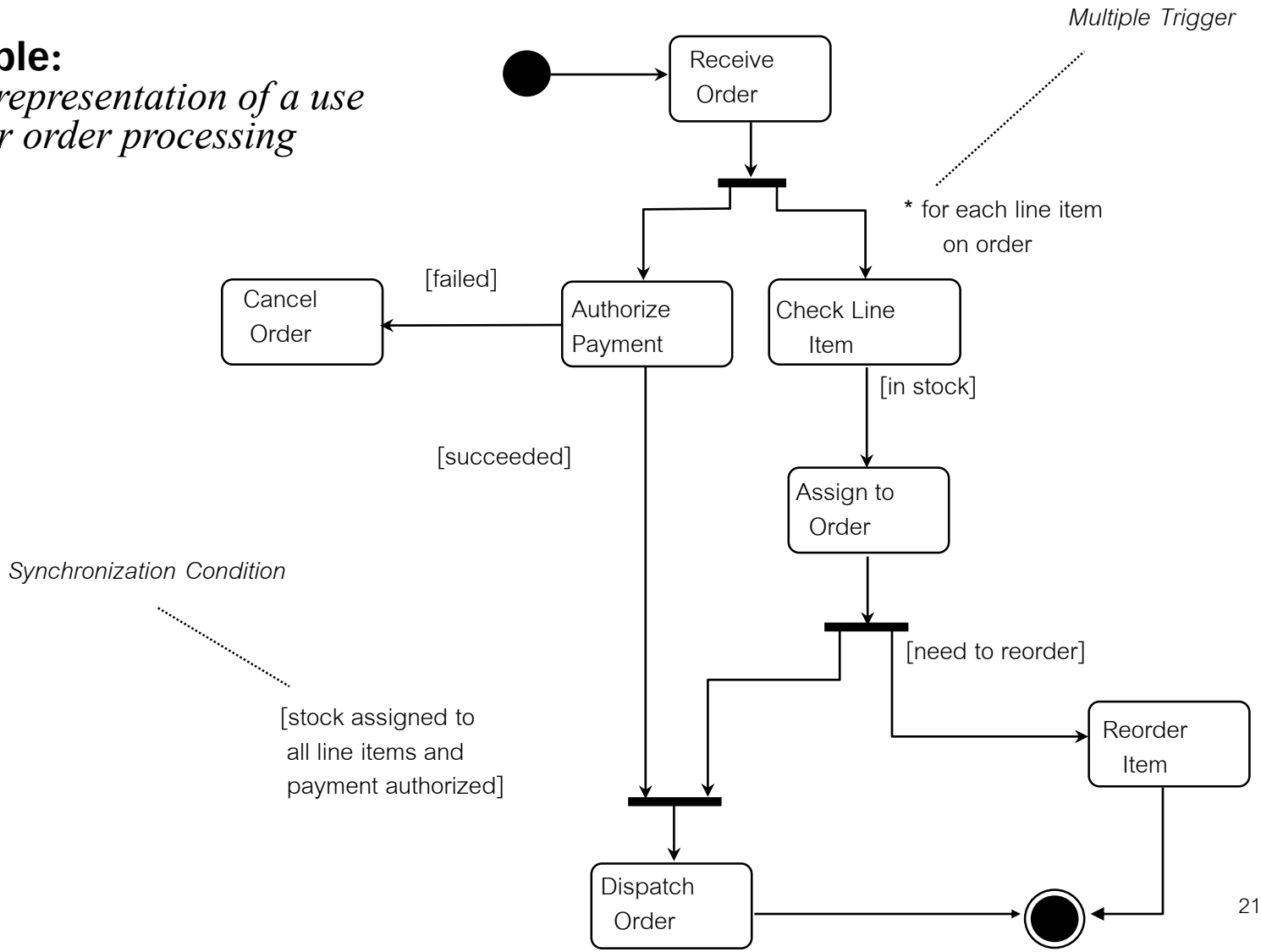
Example: *Description of a method on class Person*

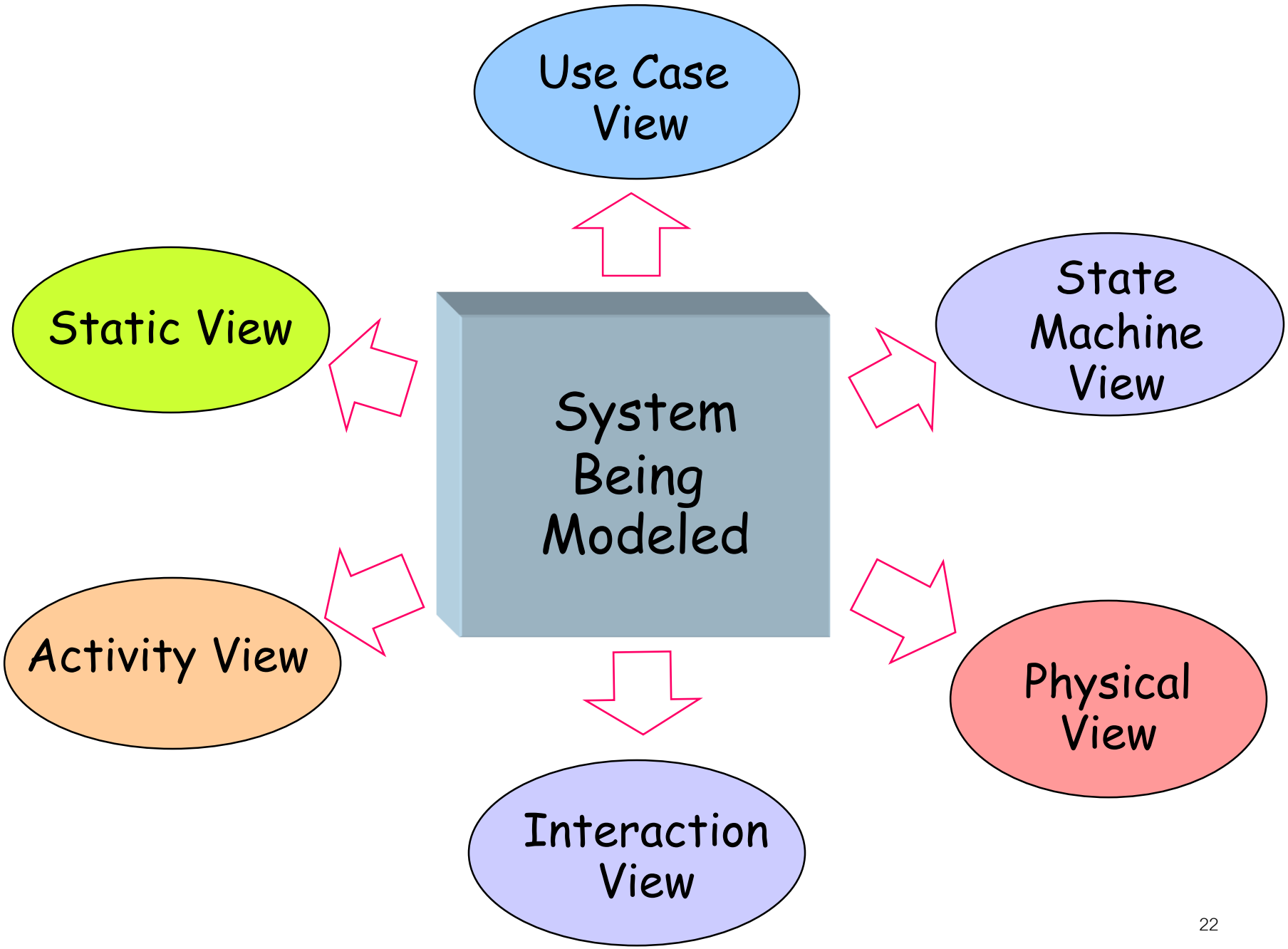


Activity Diagram

Activity diagram can also be used to describe a use case.

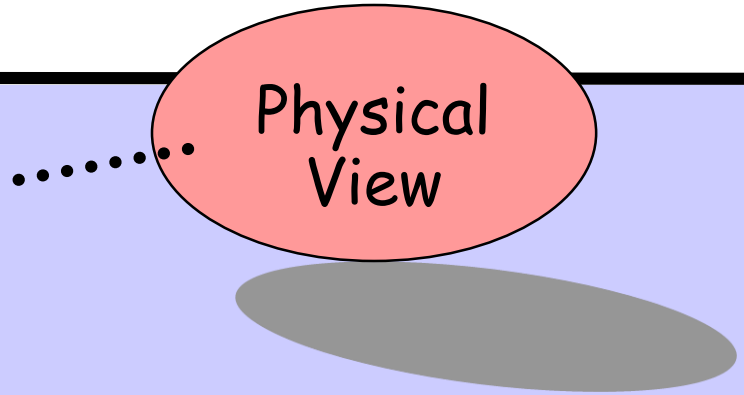
Example:
Visual representation of a use case for order processing




Use Case View
Static View
State Machine View
System Being Modeled
Activity View
Physical View
Interaction View

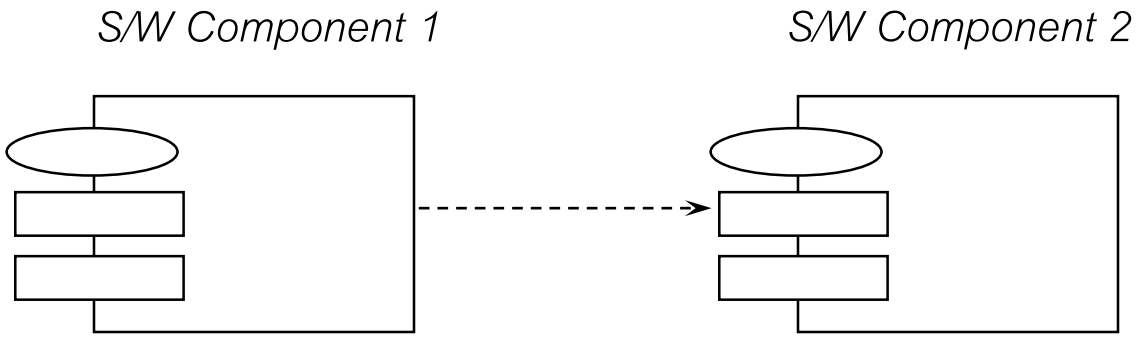
Architectural Modeling

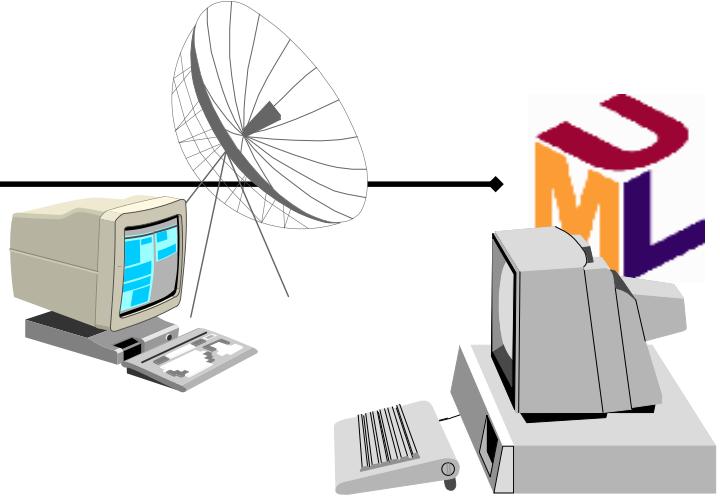
- Component Diagram
- Deployment Diagram



COMPONENT DIAGRAM

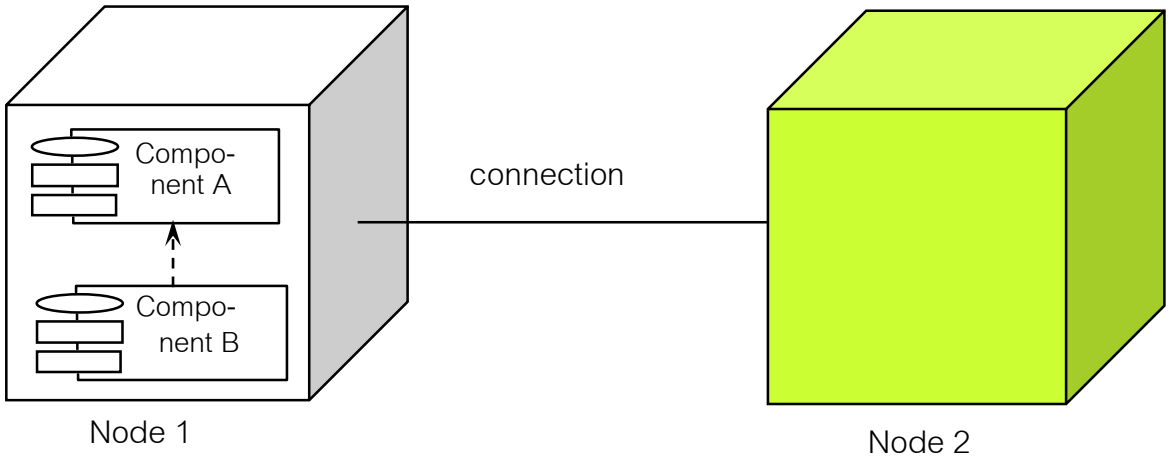
Shows the dependencies between software components





DEPLOYMENT DIAGRAM

Shows the configuration of runtime processing elements (the physical relationships among software and hardware components in the delivered system)



Deployment Diagram

Example