CHAPTER 4: OO MODELING WITH UML

Software Engineering

Contents

1 Motivation and origins.

2 View of a Software System:

- Static aspect.
- Dynamic aspect.

3 OO Methods

4 The UML Notation.

Motivation and Origins

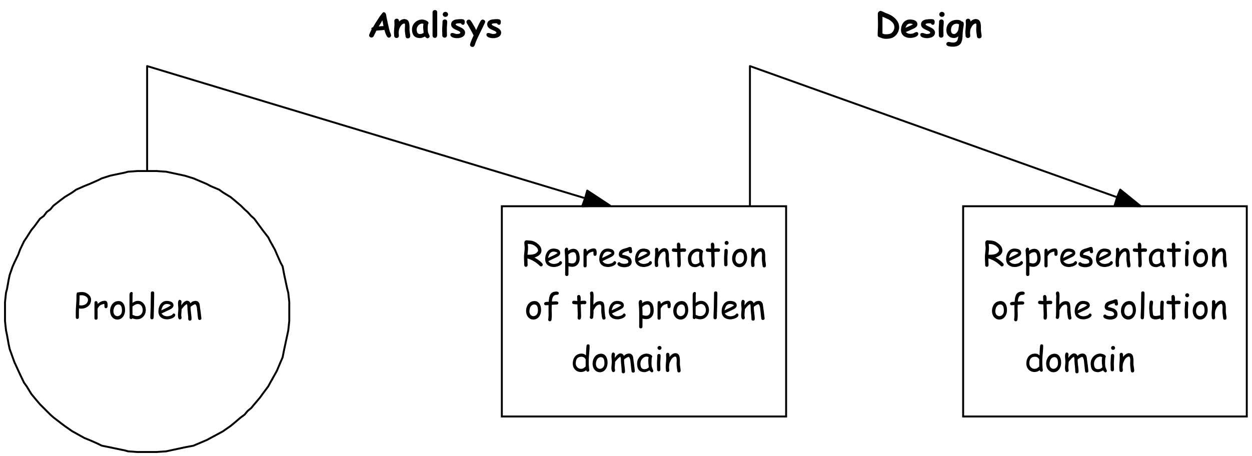
- OO Programming languages appear.
- The use of these languages requires a new viewpoint with respect to analysis and design.
- First OO analysis and design methods appear.

Motivation

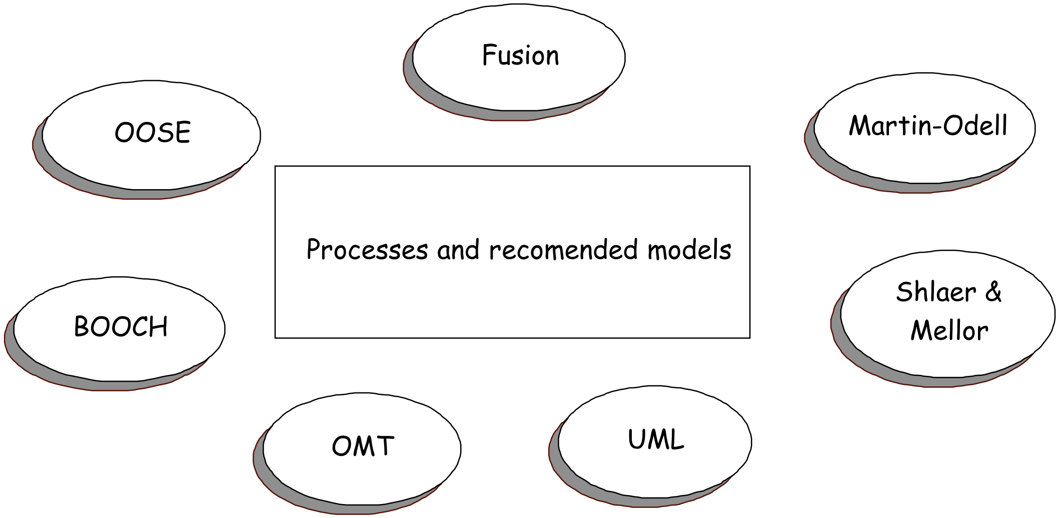
- OO methods represent requirements in terms of objects and the services they offer.
- OO methods are more “natural” than traditional ones:
  - Functions/processes vs objects.

Motivation

- In OO methods the decomposition of the system is based on objects or classes that are discovered in the problem domain

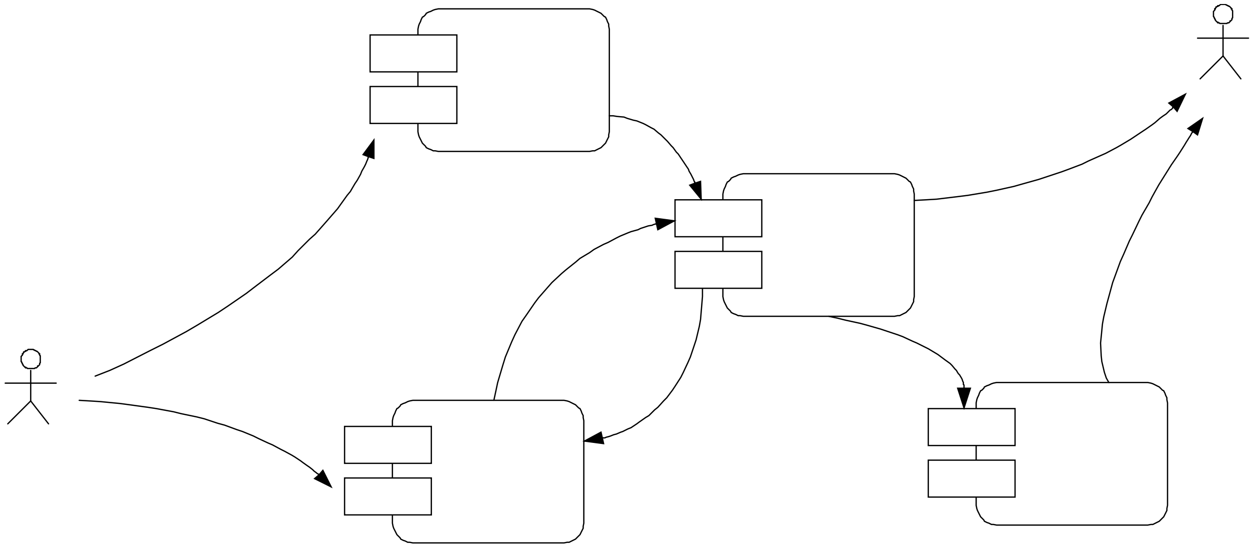

Analisys
Design
Problem
Representation of the problem domain
Representation of the solution domain

Origins



- Rumbaugh, Blaha, (OMT)- 1991
- Coad, Yourdon – 1991
- Shlaer, Mellor- 1992
- Booch- 1992
- Odell, Martin –1992
- Jacobson (OOSE) –1992
- Fusion – 1994
- Booch, Rumbaugh, Jacobson (UML) -1997

2 View of a Software System

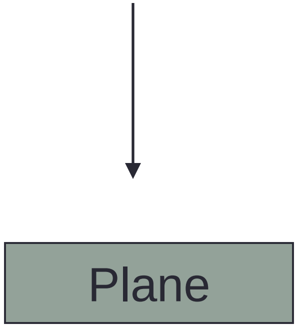


Static View

- Object:
  - Entity that exists in the real world.
  - Have identity and are differentiated.
  - Examples:
    - The bill 2003/0010
    - The plane with plate number 123
    - A customer
    - The plane with plate number 345

Static view

- Object Classes: Describe a collection of objects with:
  - Same properties.
  - Shared Behavior.
    - The plane with plate number 123
    - The plane with plate number 345



Abstraction

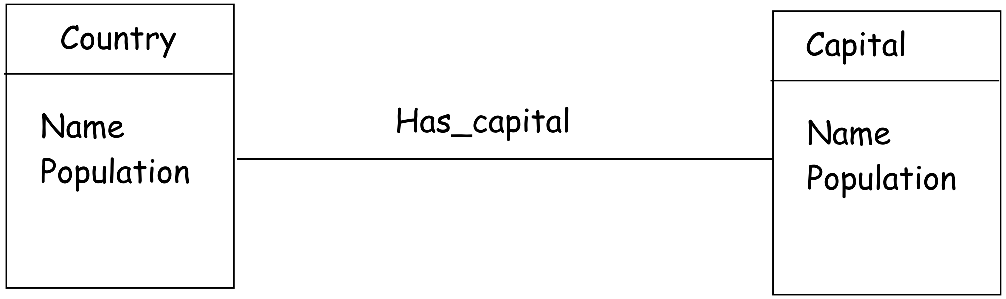
Eliminate differences among objects to keep shared aspects.

Plane

Static View: Associations

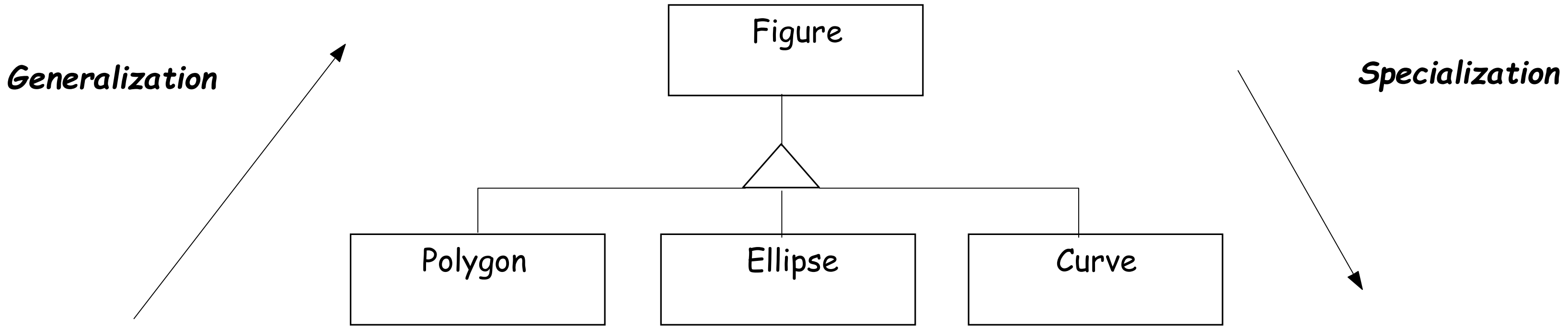
- Association: Allows linking or connecting objects of different classes.

- Example: A country has only one capital.



Static View: Generalization/ Specialization

- Generalization: Act or result obtained after distinguishing a concept that is more general than another.



- Inheritance: Allows properties and operations of a class to be accessible by a subclass.

Static View

- Static Aspect: Describes the static structure of a system and its interrelationships.

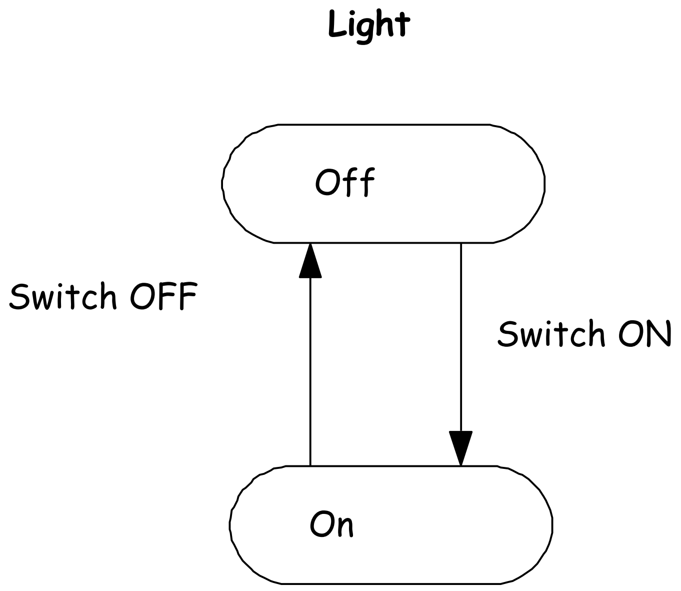
	Intra-objects	Inter-objects
Static Aspect	Object classes. Attributes Operations	Association Generalization ….

Dynamic View

- Objects communicate by means of invocation of operations on other objects.
- The dynamic view describes the aspects of a system that change over time:
  - Interactions between objects.
  - Possible states of an object.
  - Transitions between states.
  - What events are produced.
  - What operations are executed.

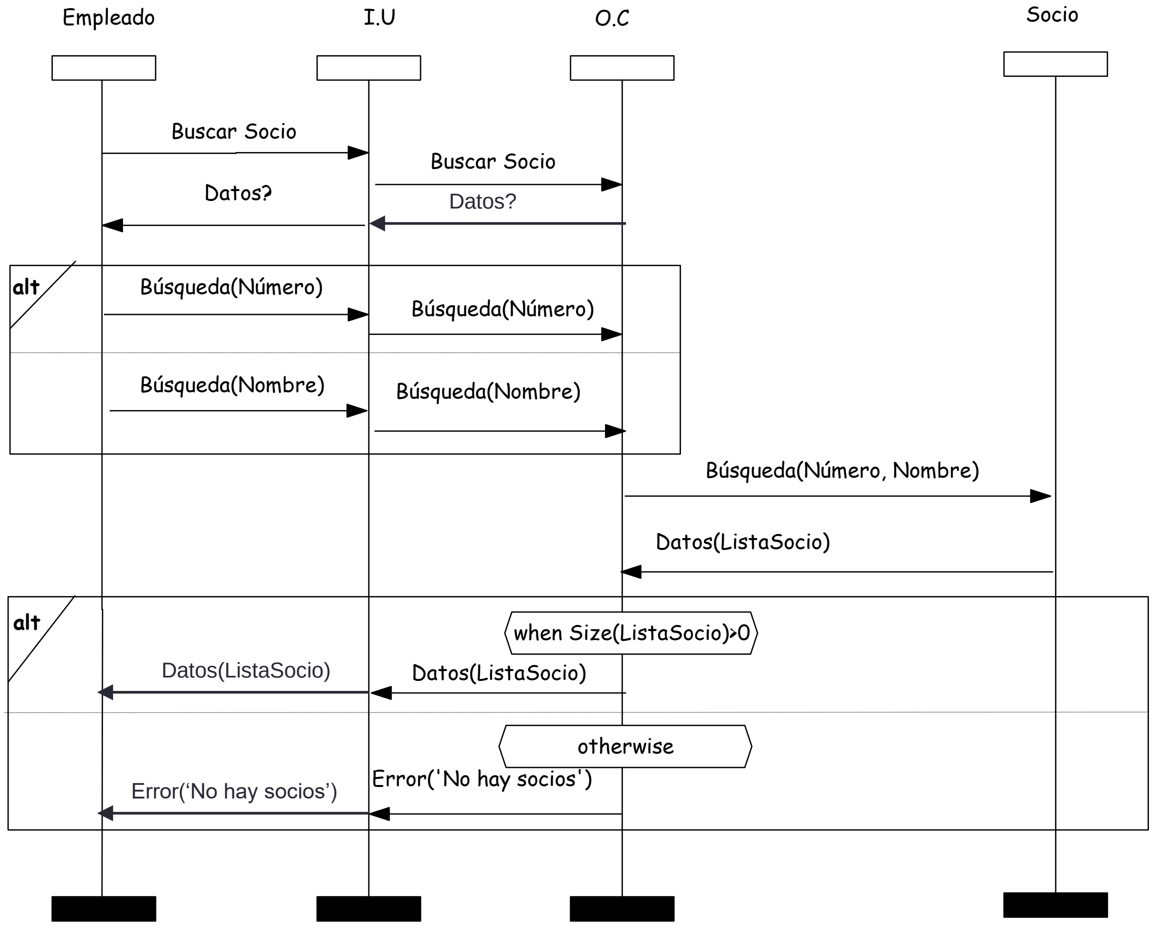
Dynamic View

- State transition diagram.



Dynamic View

- MSCs: describe interactions between different objects.



Static/Dynamic Views

- Static View: Structure and interrelationships.
- Dynamic View: Aspects that change overtime.

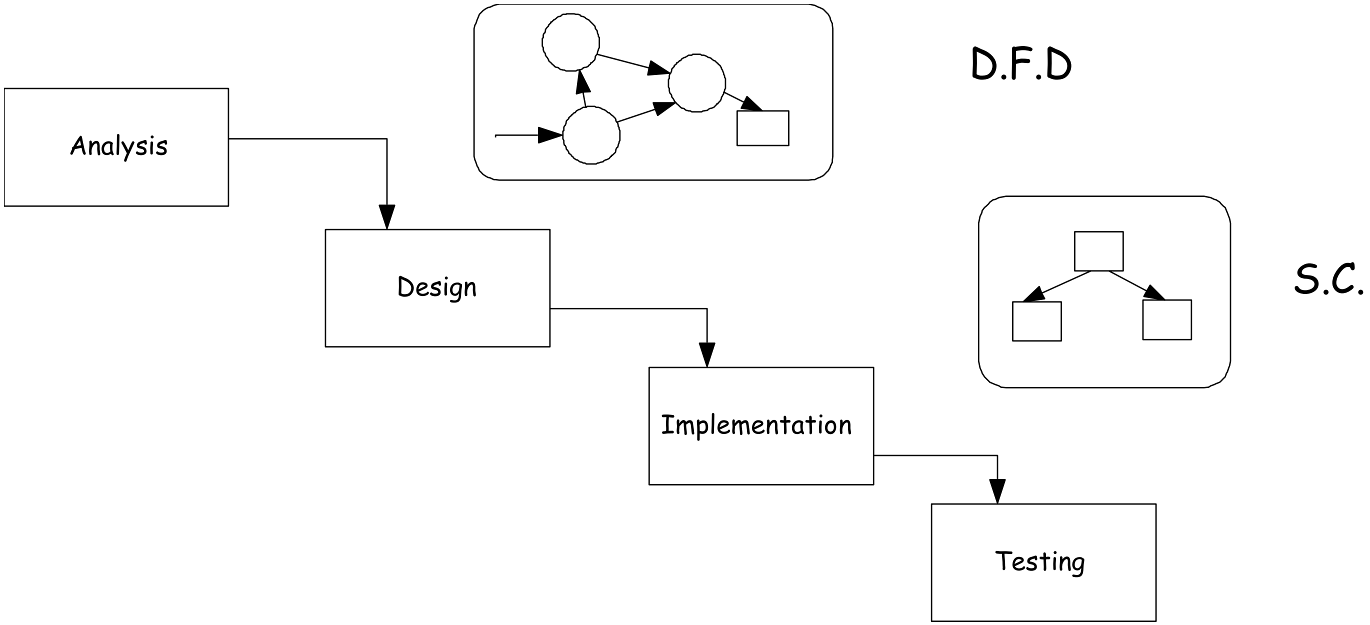
	Intra-object	Inter-objects
Static Aspect	Object classes. Attributes Operations	Association Generalization ….
Dynamic Aspect	State Transition Diagrams	MSCs ….

3 OO Methods

- OO Analysis:
  - A **specification** of the problem is created.
  - Describes **what** to do with the system.
- OO Design:
  - Definition of a software **solution** to satisfy the requirements.
  - Describes **how** the system will work

OO Methods: Continuity between models

- Structured techniques:



Analysis

Design

Implementation

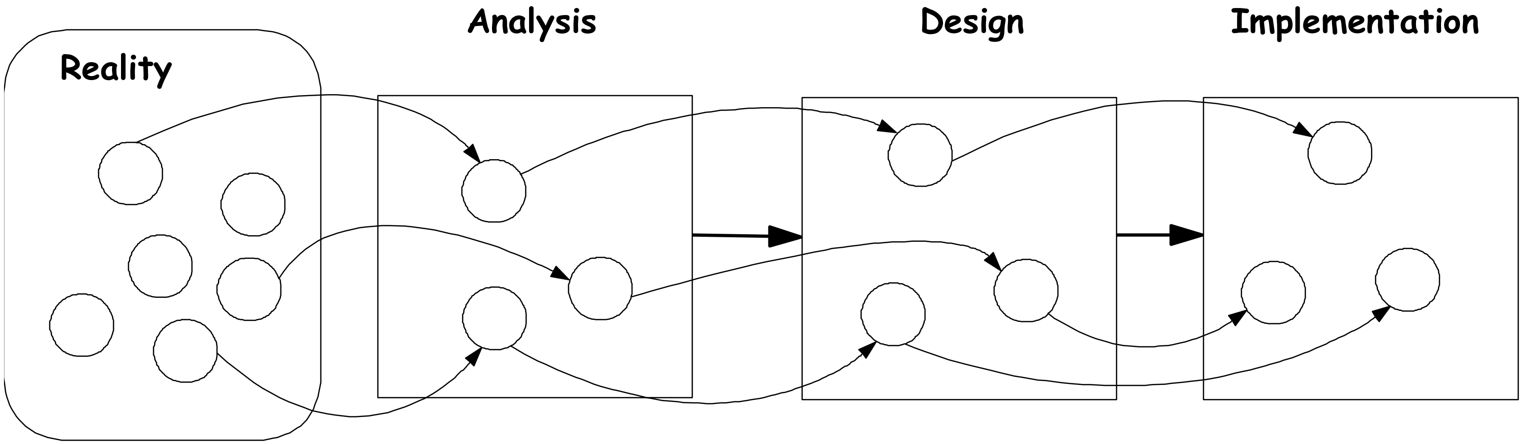
Testing

D.F.D

S.C.

OO Methods: Continuity between models

- In OO:



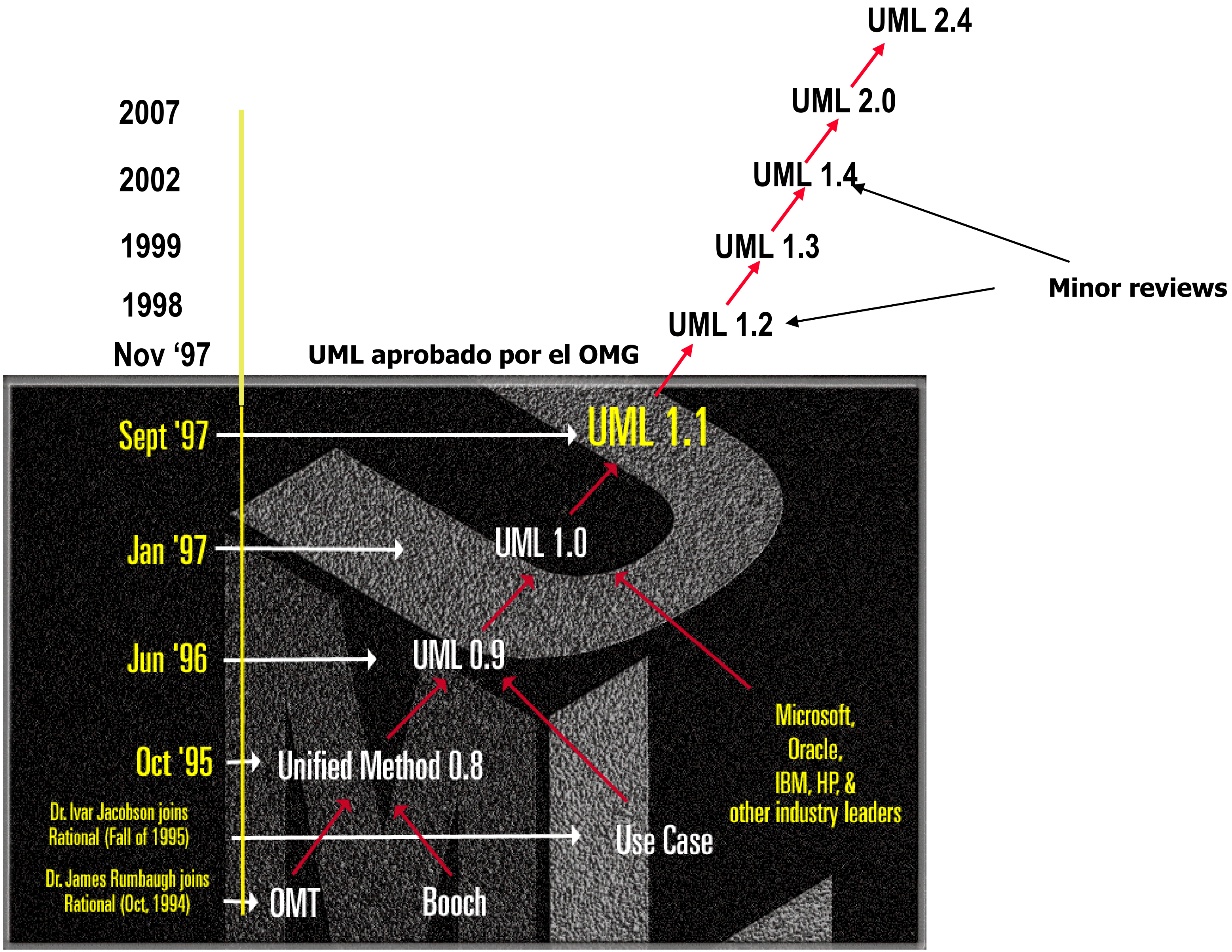
4 The UML Language

- UML = Unified Modeling Language
- UML: A general purpose language for OO modelling
- Starting Point:
  - Many OO methods with different notations.
  - Learning and tool construction inconvenients.
  - A Uniform notation needed.

UML History

- Started as the “unified method”with the participation of J. Rumbaugh and G. Booch in 1995.The same year I. Jacobson is incorporated.
- Partners in Rational Software, CASE tool Rational Rose.

UML evolution



UML 2.4

2007 UML 2.0

2002 UML 1.4

1999 UML 1.3

1998 UML 1.2

Minor reviews

Nov '97 UML aprobado por el OMG

Sept '97 UML 1.1

Jan '97 UML 1.0

Jun '96 UML 0.9

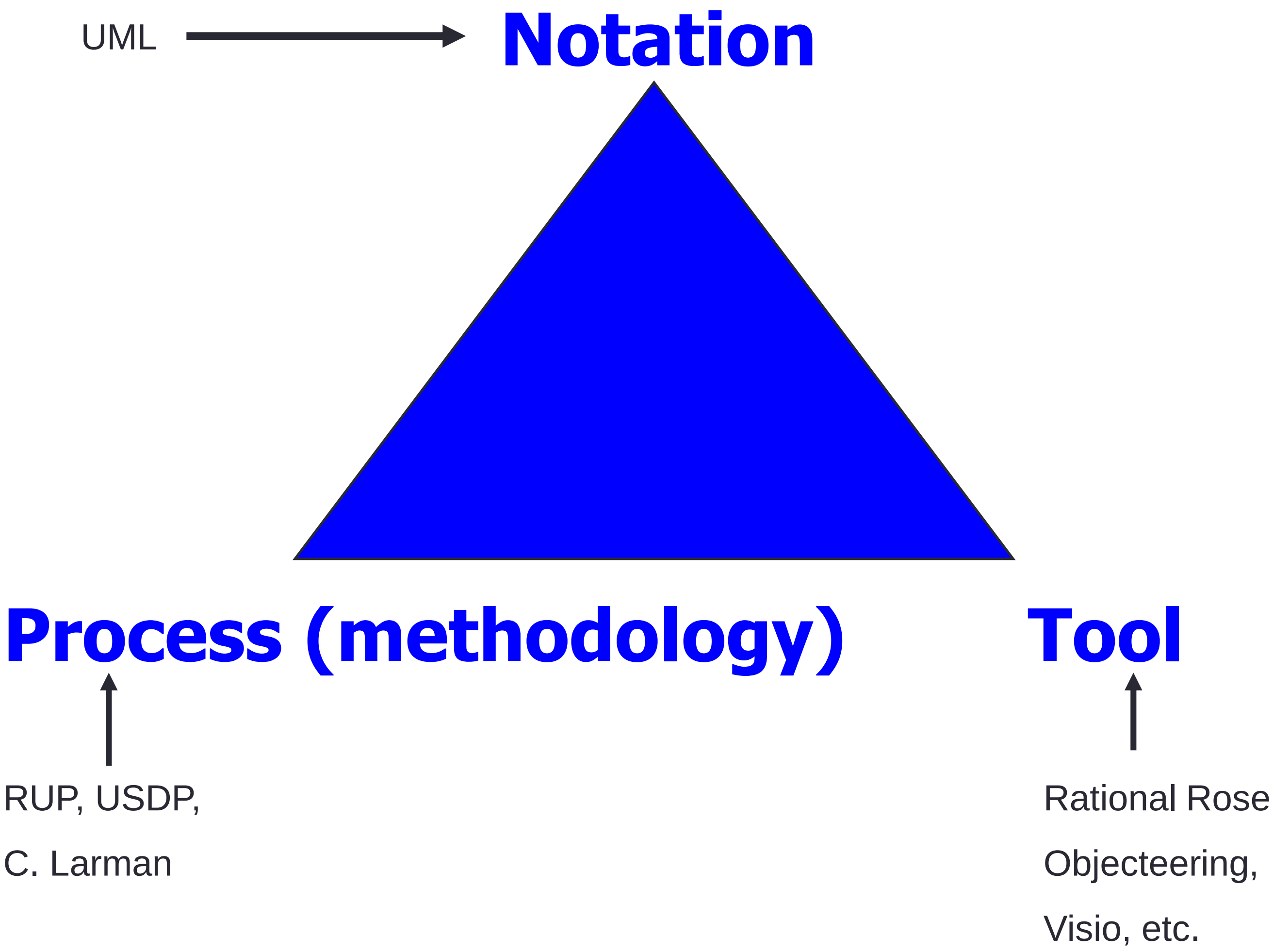
Microsoft, Oracle, IBM, HP, & other industry leaders

Oct '95 Unified Method 0.8

Dr. Ivar Jacobson joins Rational (Fall of 1995) Use Case

Dr. James Rumbaugh joins Rational (Oct, 1994) OMT Booch

UML: the success triangle



UML → **Notation**

Process (methodology)

RUP, USDP,

C. Larman

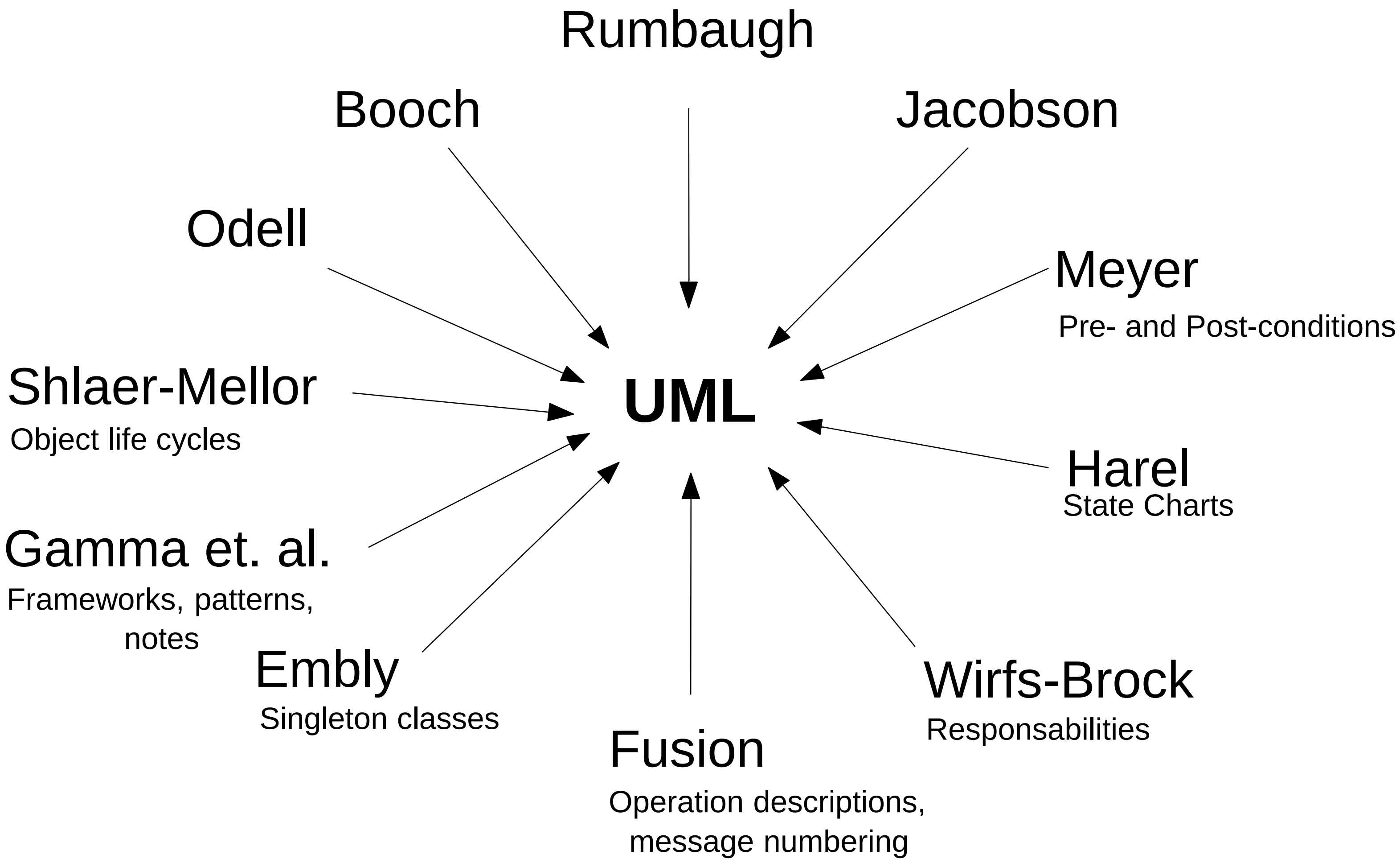
Tool

Rational Rose

Objecteering,

Visio, etc.

UML merges OO approaches



Rumbaugh

Booch

Jacobson

Odell

Meyer

Pre- and Post-conditions

Shlaer-Mellor

Object life cycles

UML

Harel

State Charts

Gamma et. al.

Frameworks, patterns,
notes

Embly

Singleton classes

Wirfs-Brock

Responsabilities

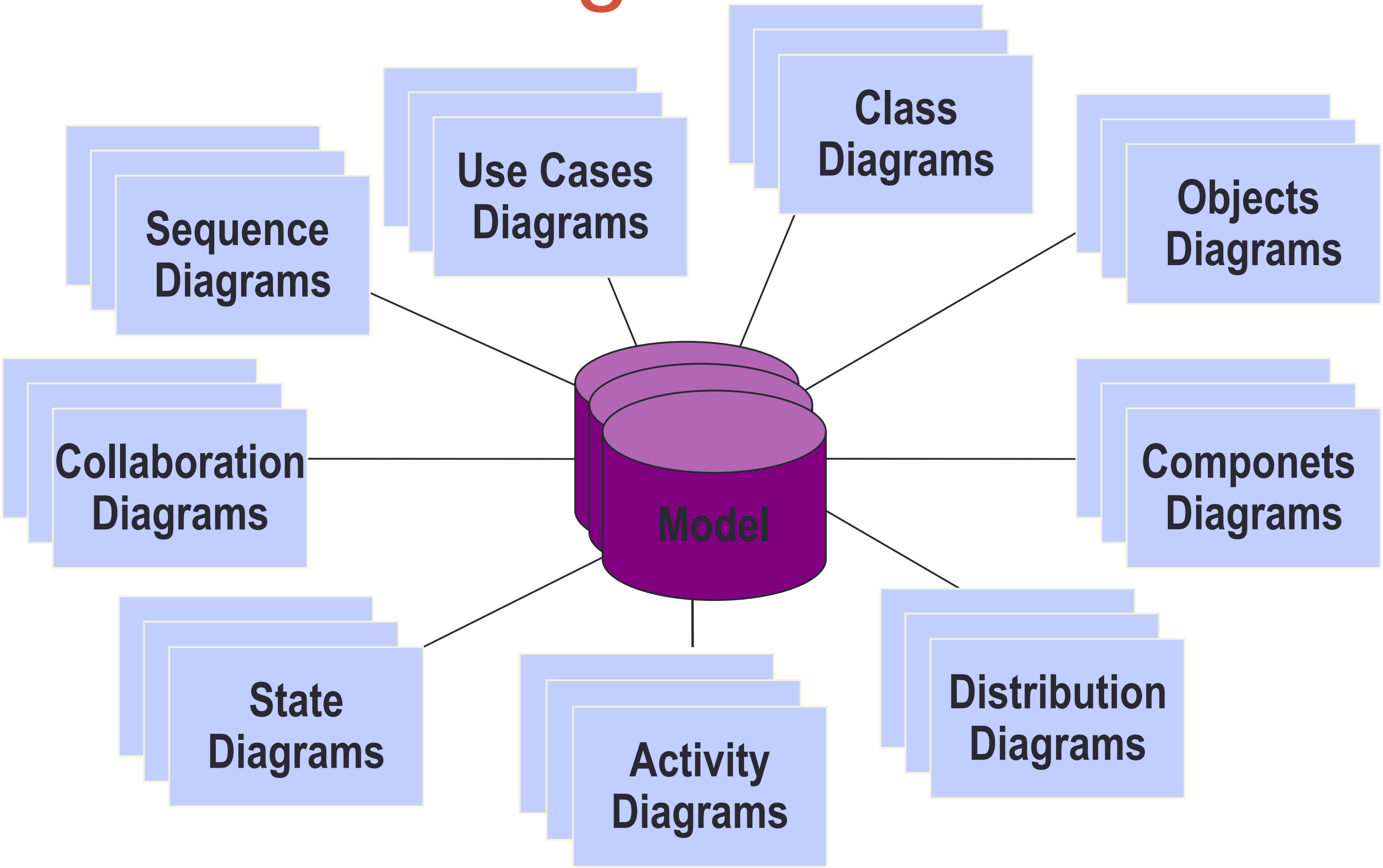
Fusion

Operation descriptions,
message numbering

UML

- UML is not a method, it is a notation to describe systems.
  - Processes based on UML:
    - USD “Unified Software Development Process” by I. Jacobson.
    - RUP “Rational Unified Process” by Rational Software.
    - C. Larman “UML and patterns”.

UML Modelling



“A model is a complete description of a system from a concrete viewpoint”

UML Charts

- Use cases Chart
- Class Chart (including instances chart)
- Behavior Charts
  - States Chart
  - Activity Chart
  - Interaction Diagrams
    - Sequence Diagram
    - Collaboration Diagram
- Implementation Diagrams
  - Components Diagram
  - Deployment Diagram

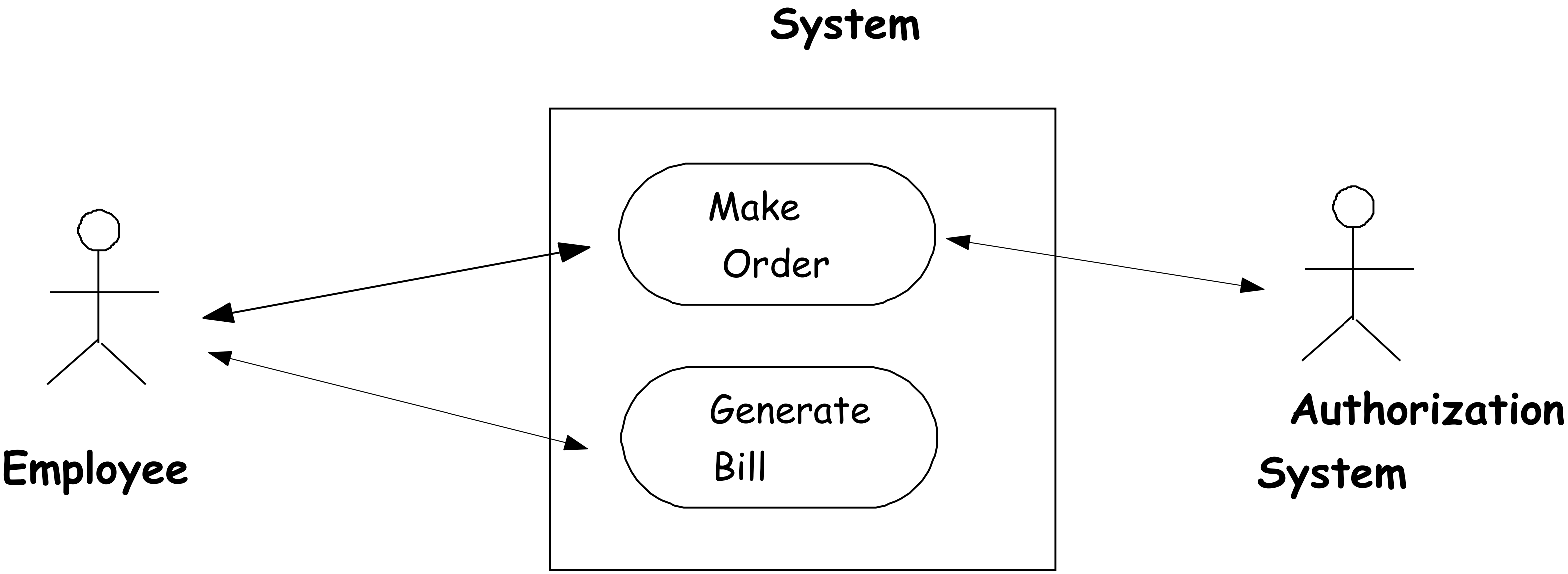
Use Cases model

- Use cases is a technique to capture information about how a system or business presently works and how it is required to work in the future.
- They are used during requirements gathering to capture functional requirements of a system to be developed.

Use Cases

- Actors: Entities that exchange information with the system.
- Types of Actors:
  - Humans.
  - Devices.
  - Other software systems.
- A use case contains a sequence of transactions that exchange the actors and the system whenever a given functionality must be executed.

Use Cases: notation



Make Order Use Case Description

User Intentions	**System Obligations**
1. Employee selects New Order	
	2. System requests customer code.
3.Empoloyee inserts customer code	
	4. System checks it exists
5. ***While not end lines selected***	
6. Employee inserts item code and quantity	
	7. System checks code exists
	8. System calculates total of line and echoes description and total of line
End While	
	9. System calculates Orde r total and echoes total.
	10. System requests customer payment card number
11. Employee inserts card number	
	12. The system verifies validity against authorization system
13. Employee selects process order	
	14. System generates order number, echoes it and stores all the information.

Synchronous extensions

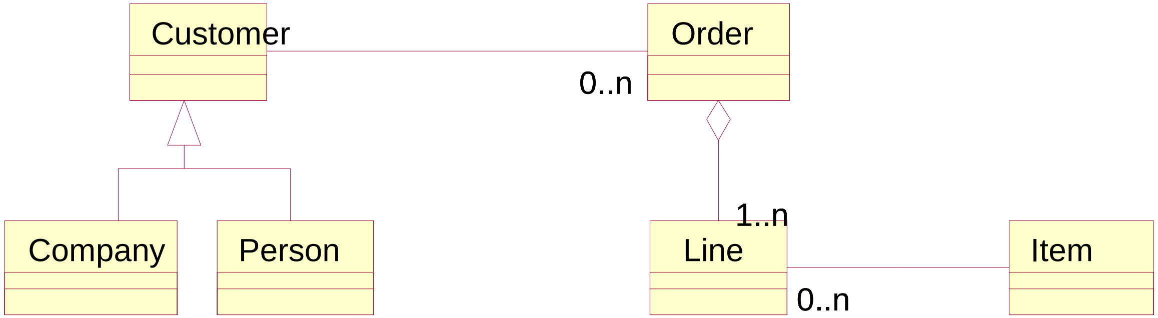
#1	15. At 4 the customer does not exist, the system reports error and go to setp 2.
#2	16. At 7, the item does not exist, the sytem reports error and go to step 5.
#3	17. At 12 The card number is not valid. Go to step 11.

Asynchronous Extensions

#4	
18. In every step the employee may select Abort	
	19. The use case ends without any information storage.

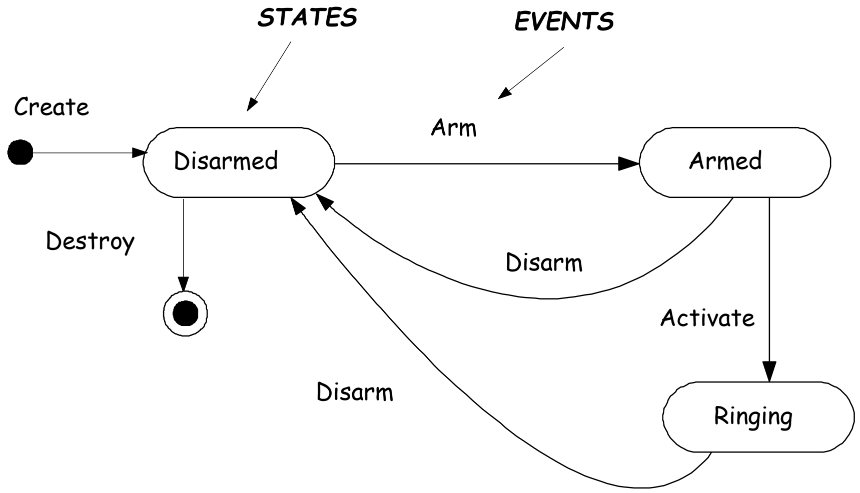
Static Models

- Show the classes of a system and the relationships between them.



Dynamic Models

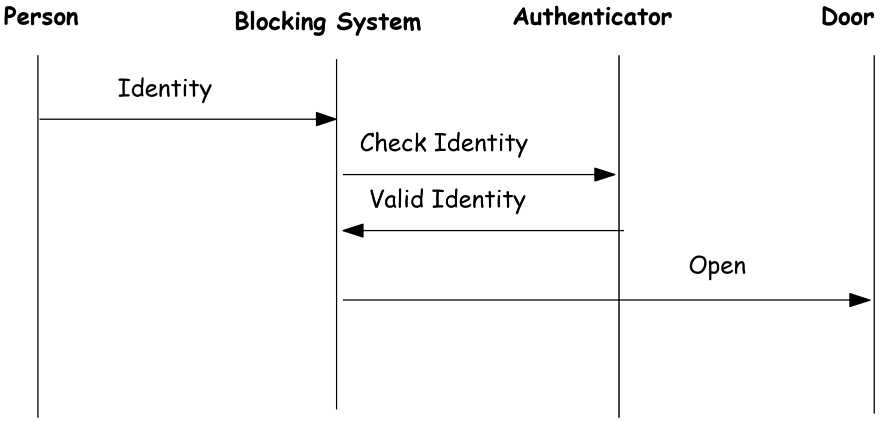
- State Transition Chart: It shows the lifecycles of the objects in the system.

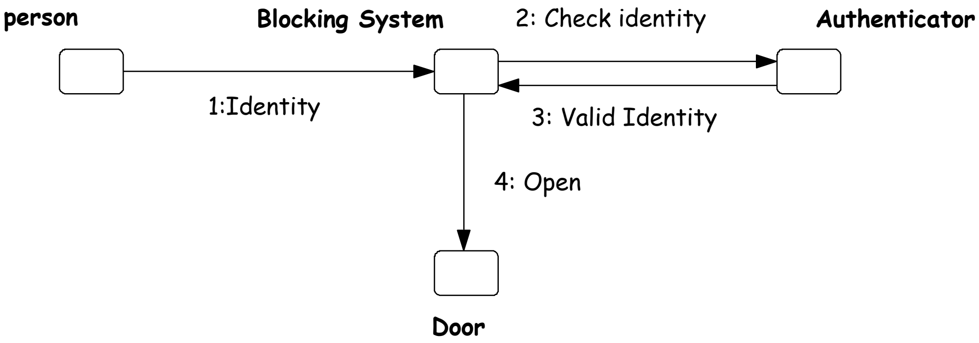


Dynamic Models

- Sequence and Collaboration Diagrams: Show messages that are exchanged by objects that participate in a scenario or use case.

Sequence & Collaboration





Activity Diagrams

- Special case of a States diagram where:

  All (or most) states are action-ones

  All (most) transitions are triggered by the finalization of actions.

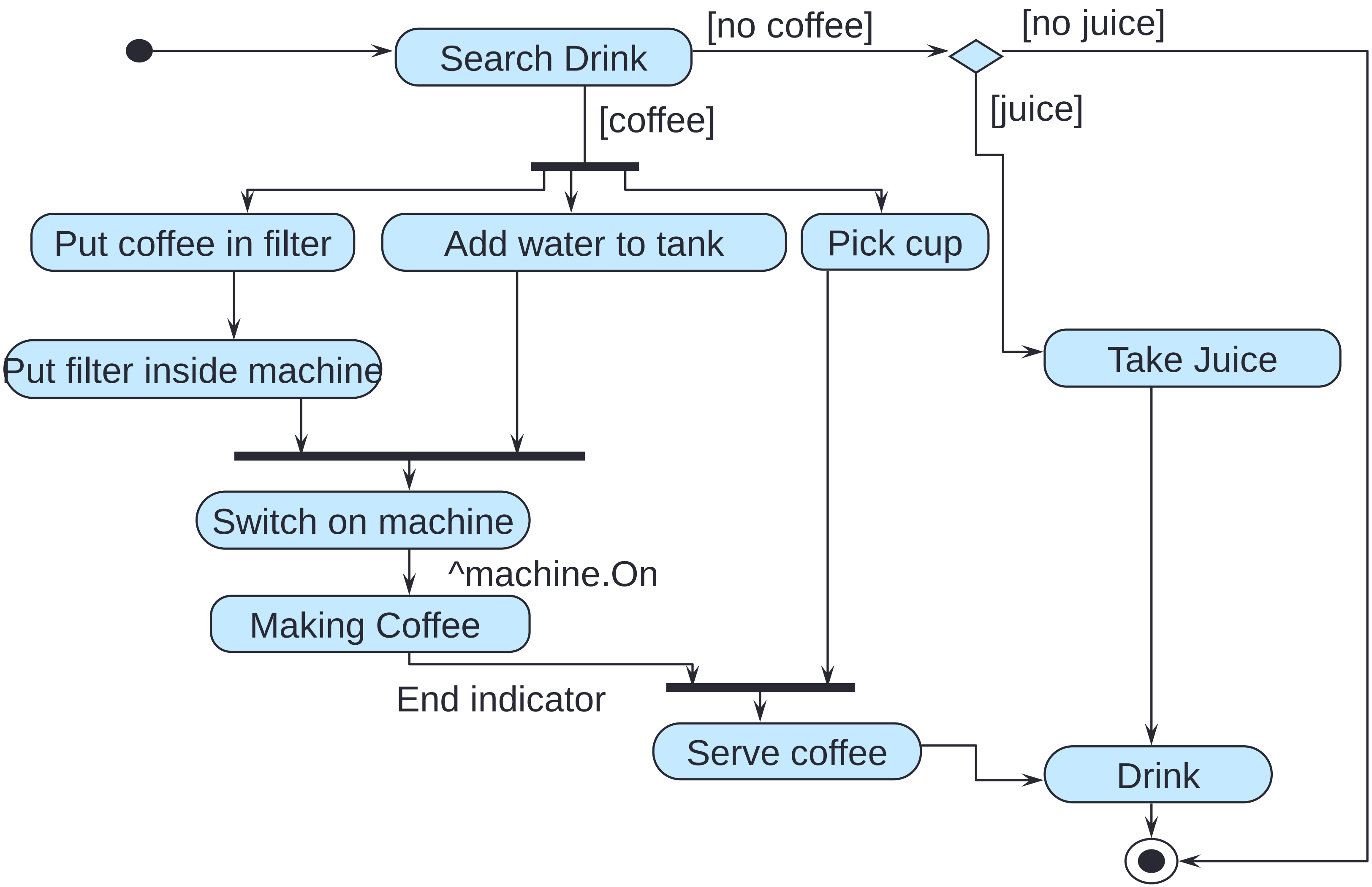
- The diagram may be associated to a :

  Class

  Implementation of an operation

  A use case

Example


[no coffee]
[no juice]
Search Drink
[coffee]
[juice]
Put coffee in filter
Add water to tank
Pick cup
Take Juice
Put filter inside machine
Switch on machine
^machine.On
Making Coffee
End indicator
Serve coffee
Drink