

# CO-OPN: Concurrent Object-Oriented Petri Nets

**Giovanna Di Marzo Serugendo**  
*University of Geneva, Switzerland*

*(Courtesy of Didier Buchs, EPFL, Switzerland)*

## Contents

- CO-OPN**
  - Synchronised Petri Nets
  - Contexts
  
- Semantics**
  
- Simulator / Prototyping**
  
- CO-OPN Framework**

## CO-OPN Specification Language

- ❑ **CO-OPN: Concurrent Object-Oriented Petri Nets**
  - **Data Structures** = Algebraic Abstract Data Types
  - **Concurrency (Intra- Inter-), Control** = Algebraic Petri Nets
  - **Modularity** = Object orientation with strict encapsulation
  - **Dynamicity** = References, Object Creation
  - **Communication** = Synchronisation between Objects
  - **Distribution** = Contexts = Units + Localisation + Migration information
  
- ❑ **CO-OPN = Data Structures + Petri Nets + Object Orientation + Contexts**

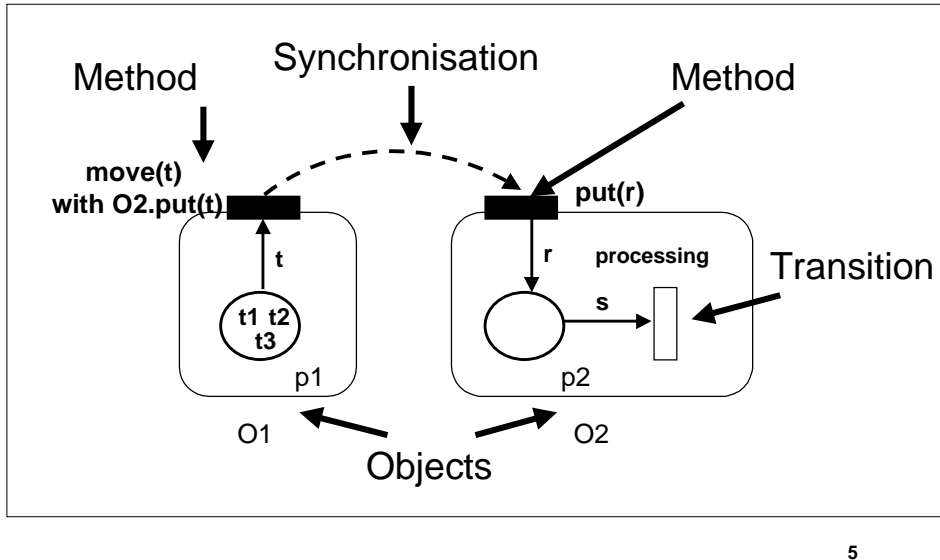
3

## CO-OPN: Why?

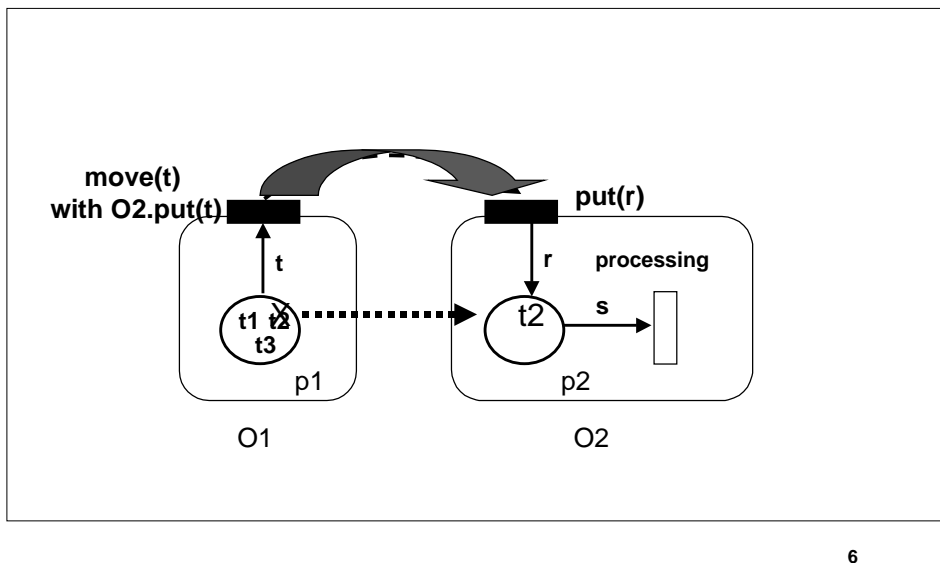
- ❑ **Development of embedded systems**
  
- ❑ **Use of a formal notation**
  - Suitable for formal verification
  - Suitable for code generation
  
- ❑ **Test the prototype behaviour in the target execution environment**
  
- ❑ **A formal notation : CO-OPN (graphical, textual)**

4

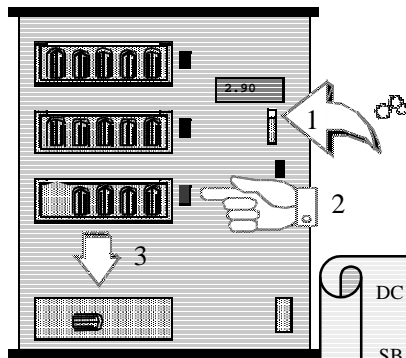
## Synchronised Petri Nets



## Synchronised Petri Nets

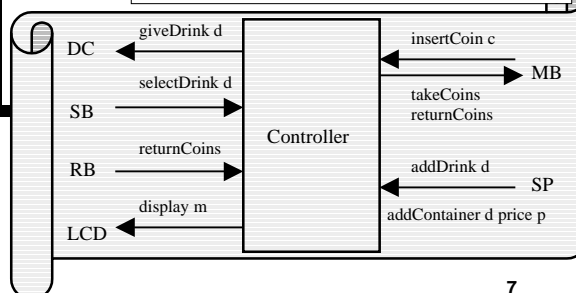


## Context: Drinks Distributor Controller



### DD Components

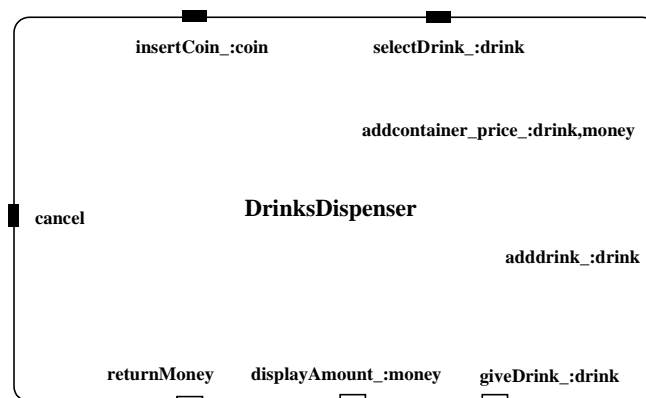
- Controller
- DC - Drink Container
- MB - Money Box
- SB, RB - Buttons
- LCD - Display
- SP - Service Person



7

## CO-OPN Component Model

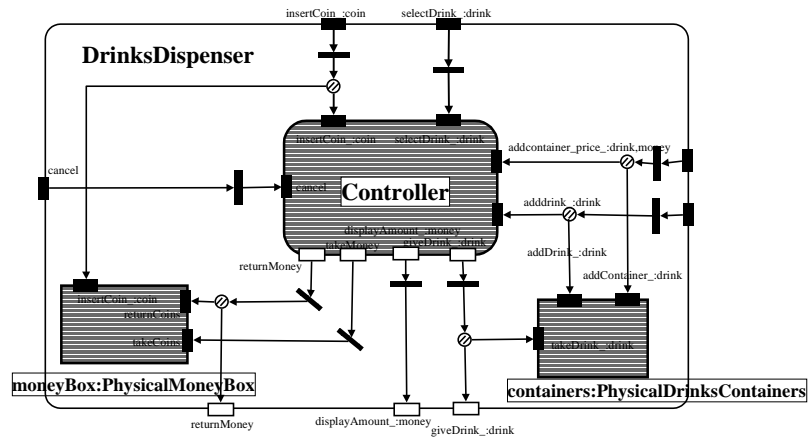
- Methods (provided services, incoming signals, ...)
- Gates (required services, outgoing signals, ...)



8

## CO-OPN Component Model

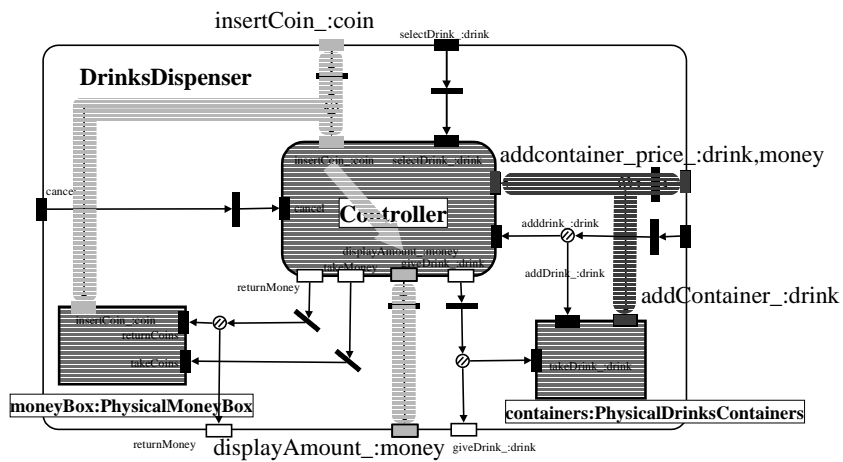
- Classes - encapsulated Petri Nets
- Contexts - coordination entities, hierarchical
- Synchronization - coordinate activities of components



9

## Semantics of CO-OPN Synchronization

### □ Transactional



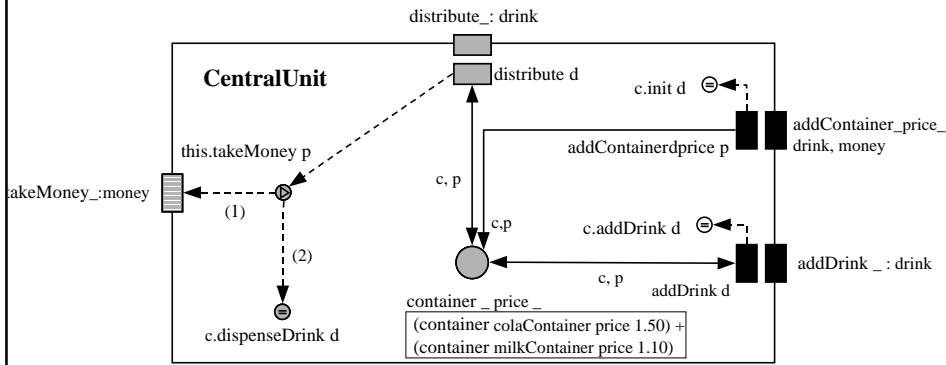
10

# How an Object Works?

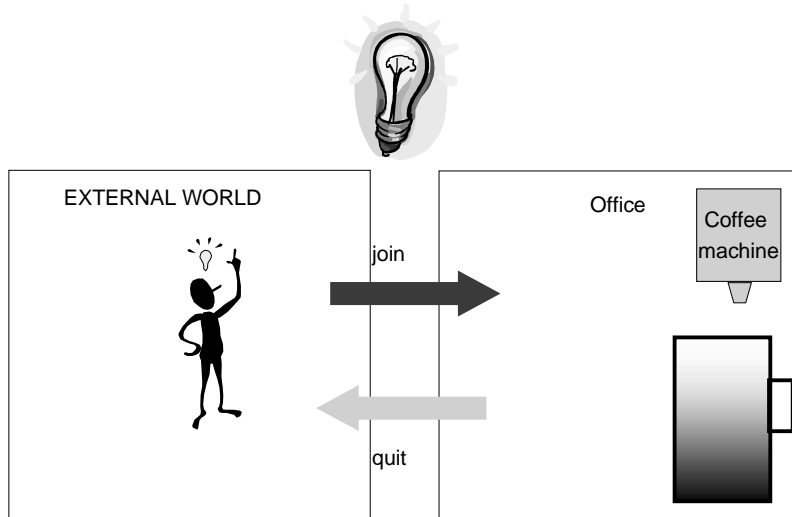
```

distribute cola
takeMoney p.50 dispenseDrink dispenseDrink cola ::
container colaContainer price 1.50 container milkContainer price 1.10

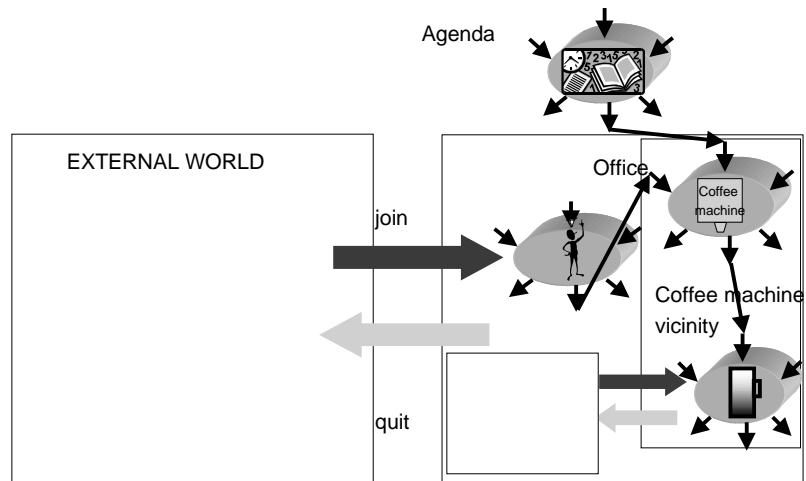
```



# Model of the Coffee machine



## Global Model based on contexts



13

## Semantics

### Labelled Transition System

### Non-Determinism

- Same Method has two or more possible behaviours
- Choice is made among several firable methods

### Concurrency/Parallelism

- Parallel firing of methods

### Synchronisation

- Transactional Semantics (all or nothing)

14

## Simulator

- ❑ **Simulation obtained from a specification**
  - CO-OPN -> Prolog (Javalog)
  - Transformation implemented in Java
  - Portable simulation
  
- ❑ **Close to original semantics**
  
- ❑ **Prolog expressive power  
(pattern matching, backtracking)**

15

## Prototyping

- ❑ **Program Generation from a Specification**
  - CO-OPN -> Java
  - Implemented in Java
  - Restricted Semantics
  
- ❑ **Operational aspects**
  - Term-rewriting
  - Petri-Nets execution
  - Implementation of Atomicity and Transactions
  
- ❑ **Architectural aspects**
  - Java Beans architecture
  - Integration in asynchronous systems

16

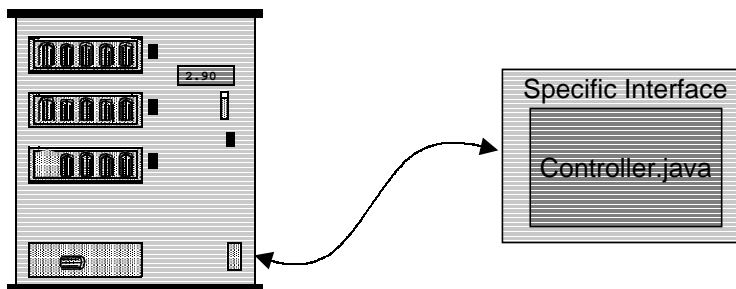
## Using the Generated Code for prototyping

### ❑ Simulation and animation

- GUI, Java Applets

### ❑ Prototyping

- Real Equipment
- Physical Model of Real Equipment (Lift with Lego RCX)



17

## CO-OPN to Java Translation: methods

Context Controller

```
Method insertCoin _ : coin;  
Gate distribute _ : drink;
```

CO-OPN

```
Controller c = new Controller();  
  
try{  
    CoopnTransaction T = new CoopnTransaction();  
    c.insertCoin(T, coinVariable);  
    T.commit();  
}catch(MethodNotFirableException e){}
```

Java

18

## CO-OPN to Java Translation: gates

```
Context Controller
  Method insertCoin _ : coin;
  Gate distribute _ : drink;
```

CO-OPN

```
Controller c = new Controller();

c.addDistributeListener(new DistributeListener(){
  public void distribute(DistributeEvent e){
    e.getDrink();
    ...
  }
});
```

Java

19

## Prototyping

### Close to intuition

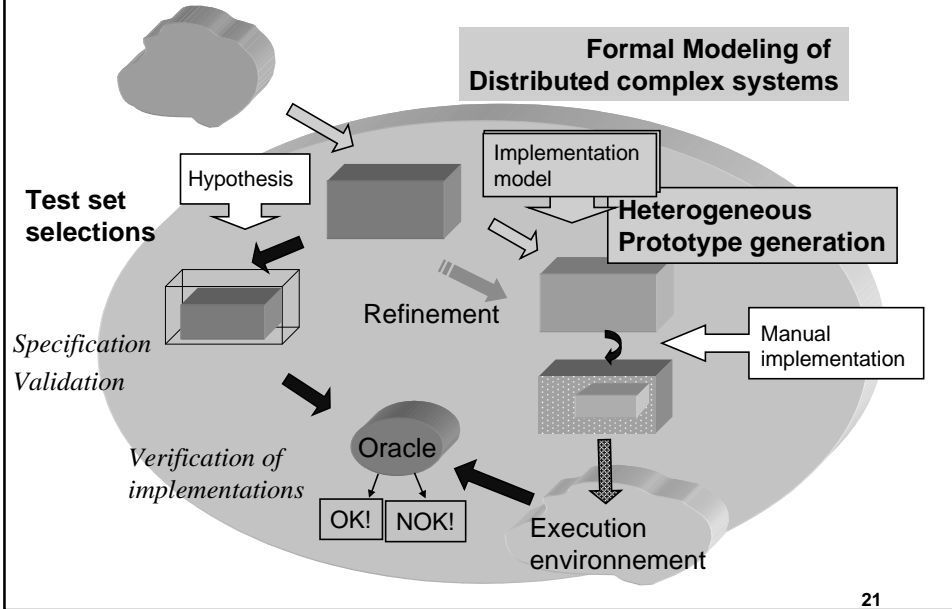
- CO-OPN Contexts -> Java Beans
- CO-OPN Classes -> Java Classes
- CO-OPN Objects -> Java Objects
- Gate activity -> Java event
- Context migration -> use of proxies + pattern matching for references

### Differences

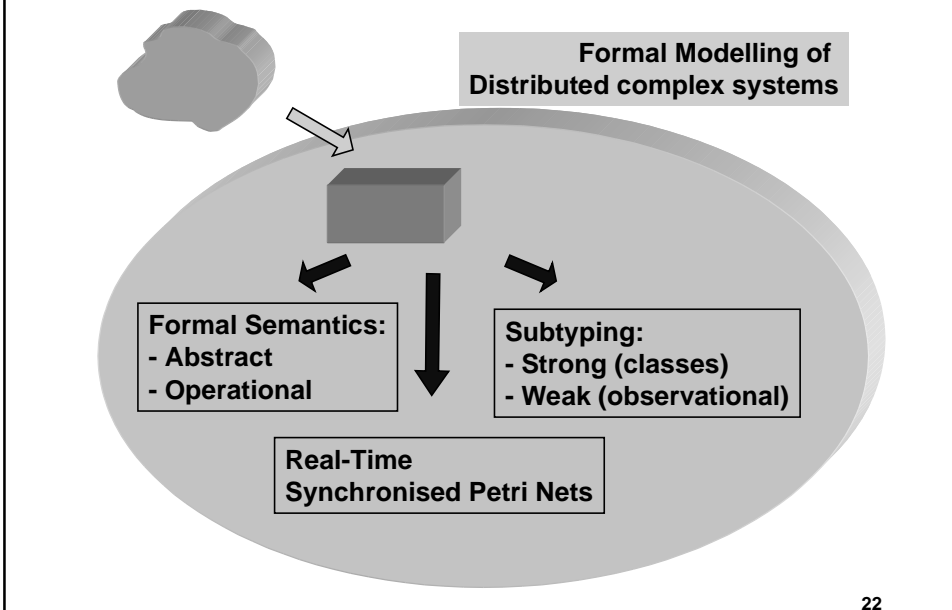
- Non-Deterministic choice among several behaviours of the same method
- Non-Deterministic choice among several fireable methods
- => Non-Deterministic Java
- Transactional semantics for methods
- Explicit Parallelism

20

# CO-OPN Framework



# CO-OPN Framework



## Modelling of distributed complex systems

- ❑ **Distributed Systems**
  - Concurrency (PN)
  - Localisation (Context)
  - Dynamics (OO)
  - Synchronisation
- ❑ **Structured Approach**
  - System level
  - Component level (Context)
  - Object level (PN)
- ❑ **Life cycle**
  - Refinements -> links with design

23

## Links to CO-OPN

- ❑ **CO-OPN**
  - [http://lglww.epfl.ch/Conform/home\\_page.html](http://lglww.epfl.ch/Conform/home_page.html)
- ❑ **CO-OPN Tools and papers**
  - <http://lglww.epfl.ch/Conform/CoopnTools>
- ❑ **CO-OPN Running examples (Lift, Philosophers)**
  - <http://lglww.epfl.ch/Conform/Examples/index.html>

24