Universidade do Minho
Escola de Engenharia
Departamento de Informática

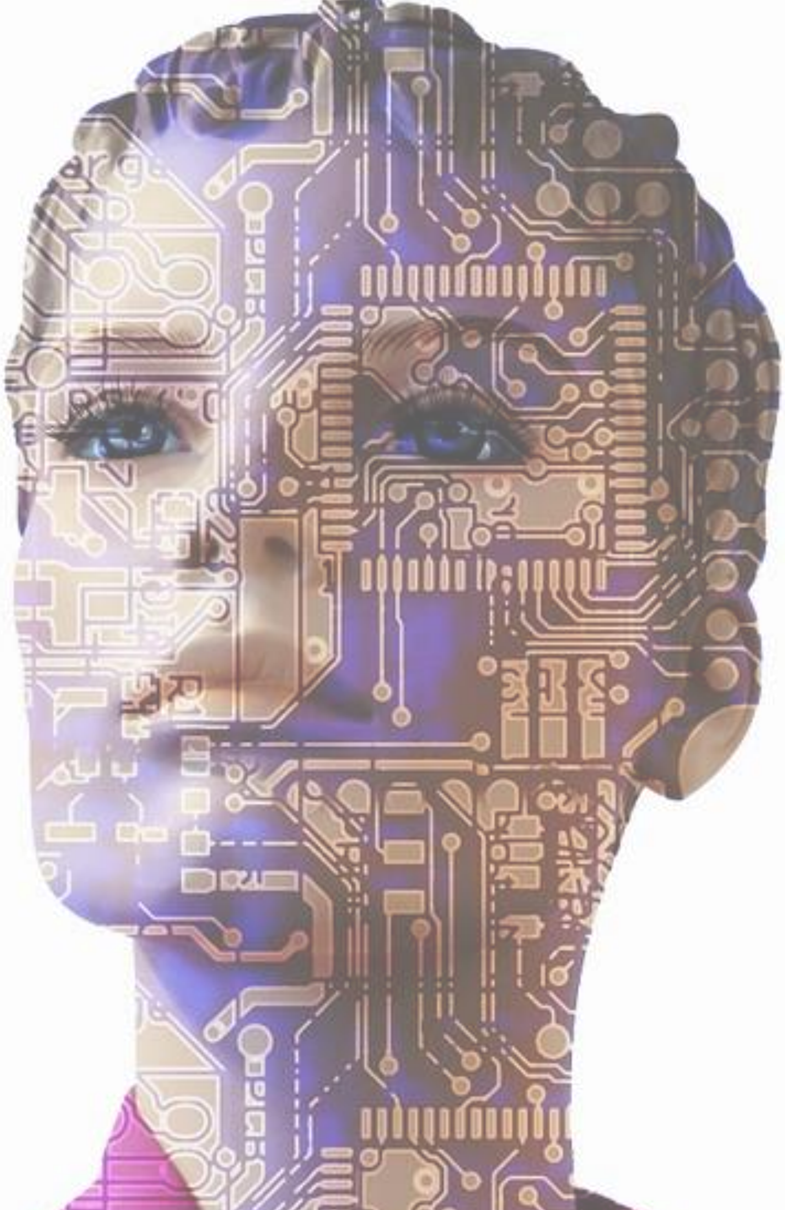
Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Agentes Inteligentes
2019/2020

Paulo Novais, Filipe Gonçalves

- Paulo Novais – pjon@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

Agent UML



Software Agents:

- Computational entity located in an environment in which it performs actions with autonomy and proactivity, according to its own perception. May have reasoning and adaptability (e.g. network management, process management, information search, etc.)

Multi-agents System:

- Group of agents that interact by understanding and coordinating in global tasks involving cooperation or competition

Agents as extensions to **Active Objects:**

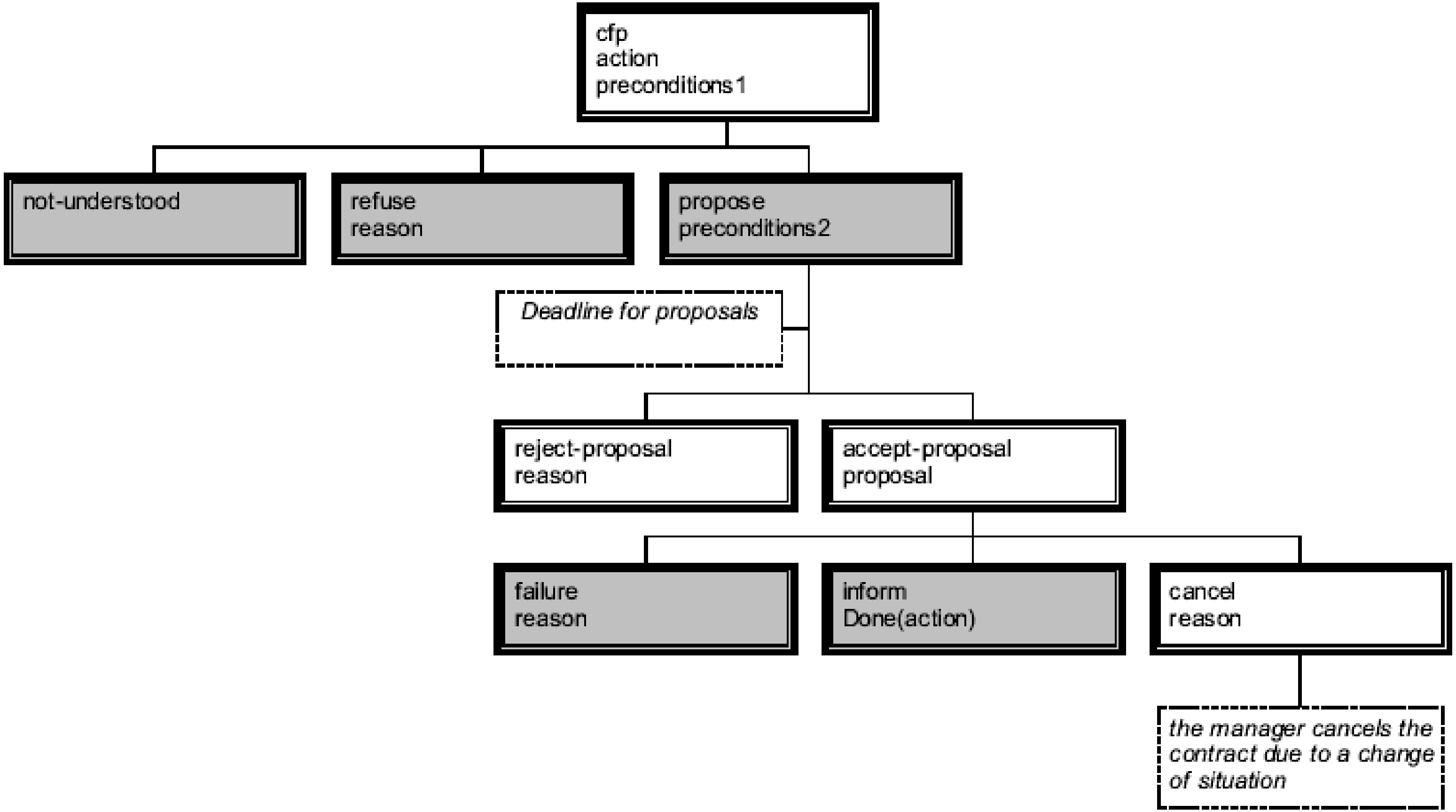
- ○ Passive Agents (Accept/Refuse Requests)
- ○ Proactive Agents (Starts activities without external intervention)

Unified Modeling Language (UML) applied in object-oriented software modeling (adopted by OMG in November 1997)

AUML: UML Variations and Extensions for Agent Activity Modelling

- FIPA (www.fipa.org)
- OMG_AUML Agent Group (http://aot.ce.unipr.it/auml/)
- Interaction Protocol Representation for Agents

FIPA Notation



AUML – Agent UML

- The goal of AUML is to develop a formal specification of agent interaction protocols (AIP).
- UML sequence diagram adaptation to model agent interactions
- This was followed by the adaptation of other diagrams

UML Representation Extensions:

- "Packages"
- Templates
- Sequence Diagrams
- Collaboration Diagrams
- Activity Diagrams
- State Diagrams
- Class and Object Diagram

AUML – Agent UML

AUML models application:

- Agent Interaction Protocols (AIP) Specification
- More detailed specification of the invocation of shares
- Package Extension
- Deployment Diagram Extension

AUML takes a layered approach to protocols:

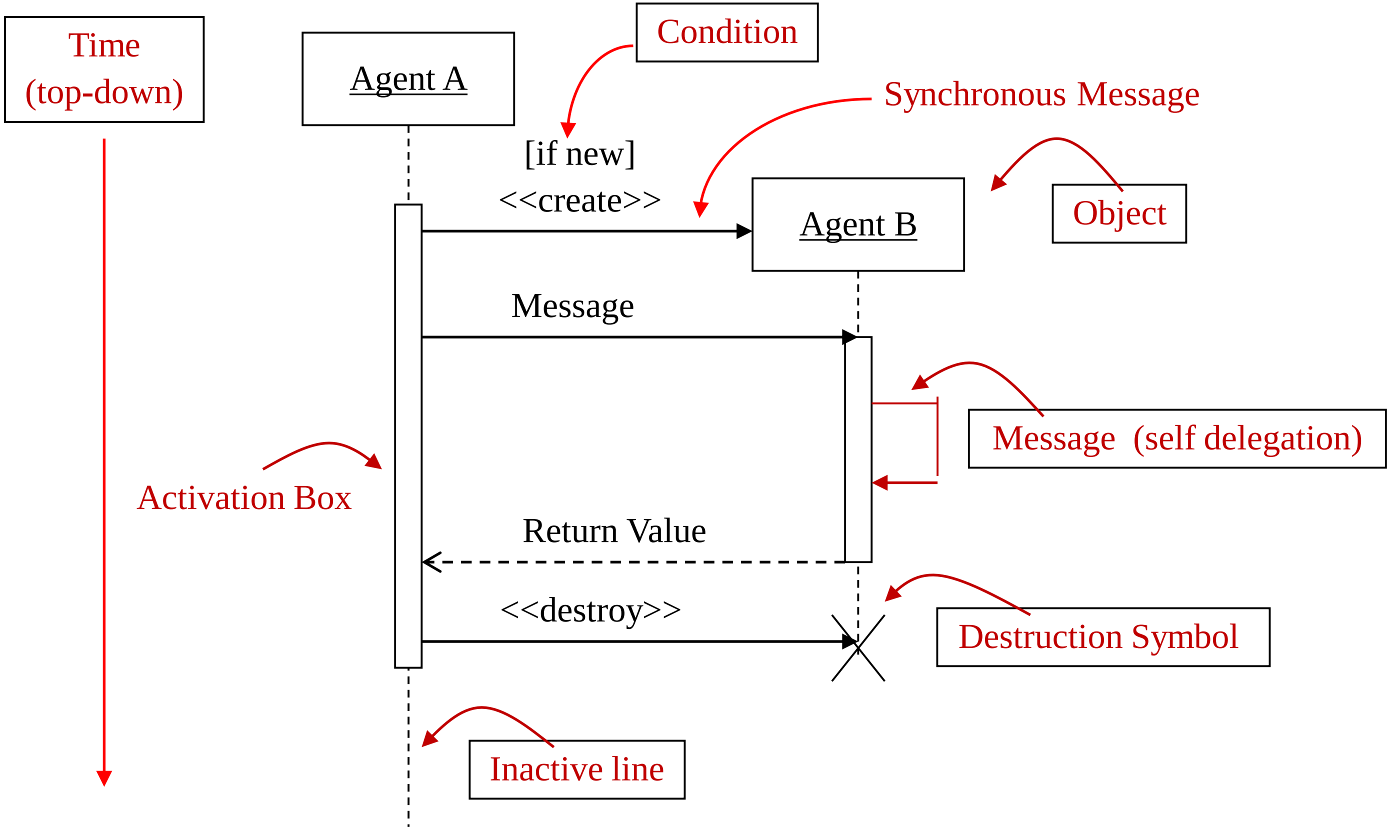
- **Level 1:** Represents the general protocol (sequence diagrams, packages, models)
- **Level 2:** Represent agent interactions (sequence, collaboration, activity, status diagrams)
- **Level 3:** Represent internal agent processing (activity and state diagrams)

Level 1: General Protocol

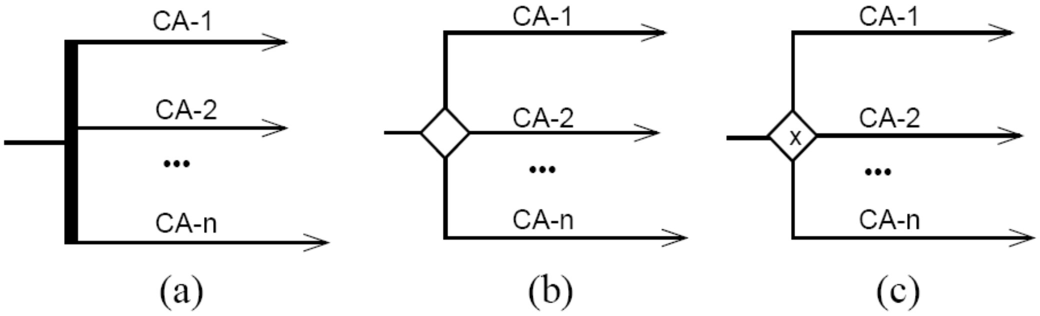
Sequence Diagram

- Defines the behaviour of object groups
- Basic interactions between objects at method invocation level
- In AUML, they enable demonstration of interactions / communications between System Agents

UML Sequence Diagram

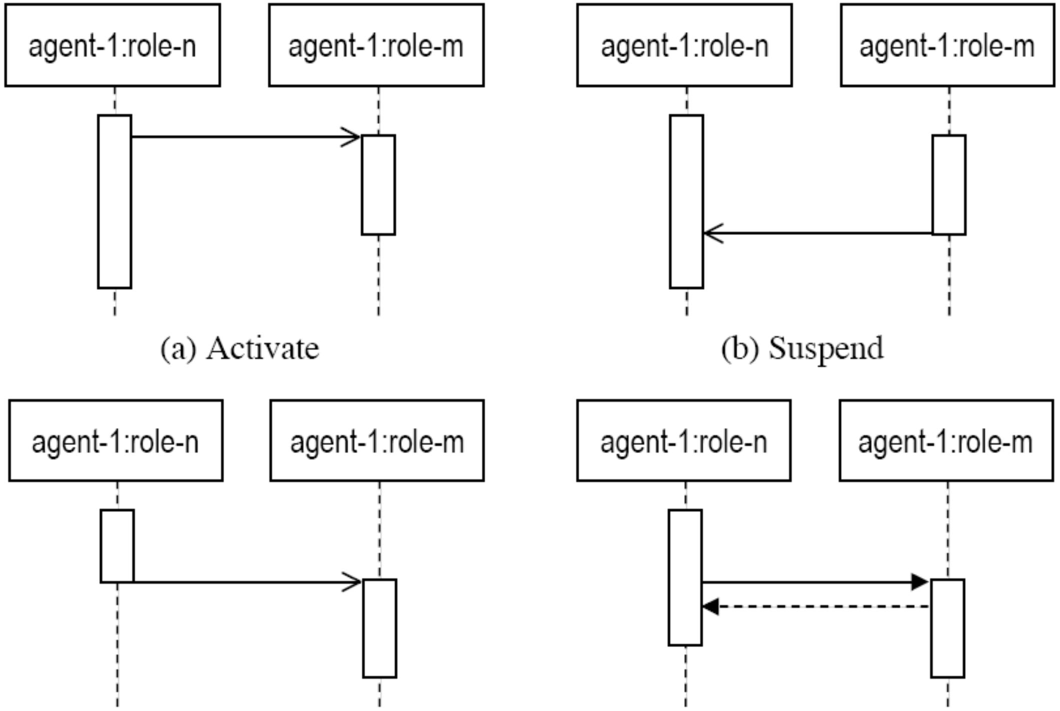


UML Sequence Diagram



paralelismo inclusivo exclusivo

Different Agent States



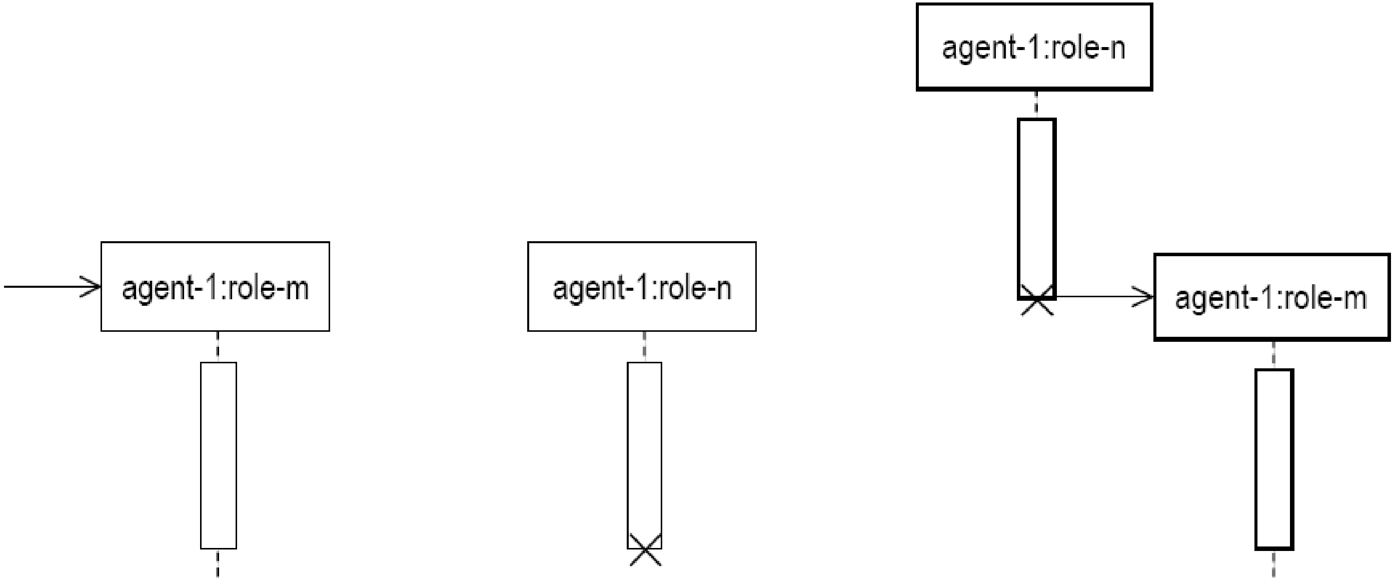
(a) Activate

(b) Suspend

(c) Shift (asynchronous)

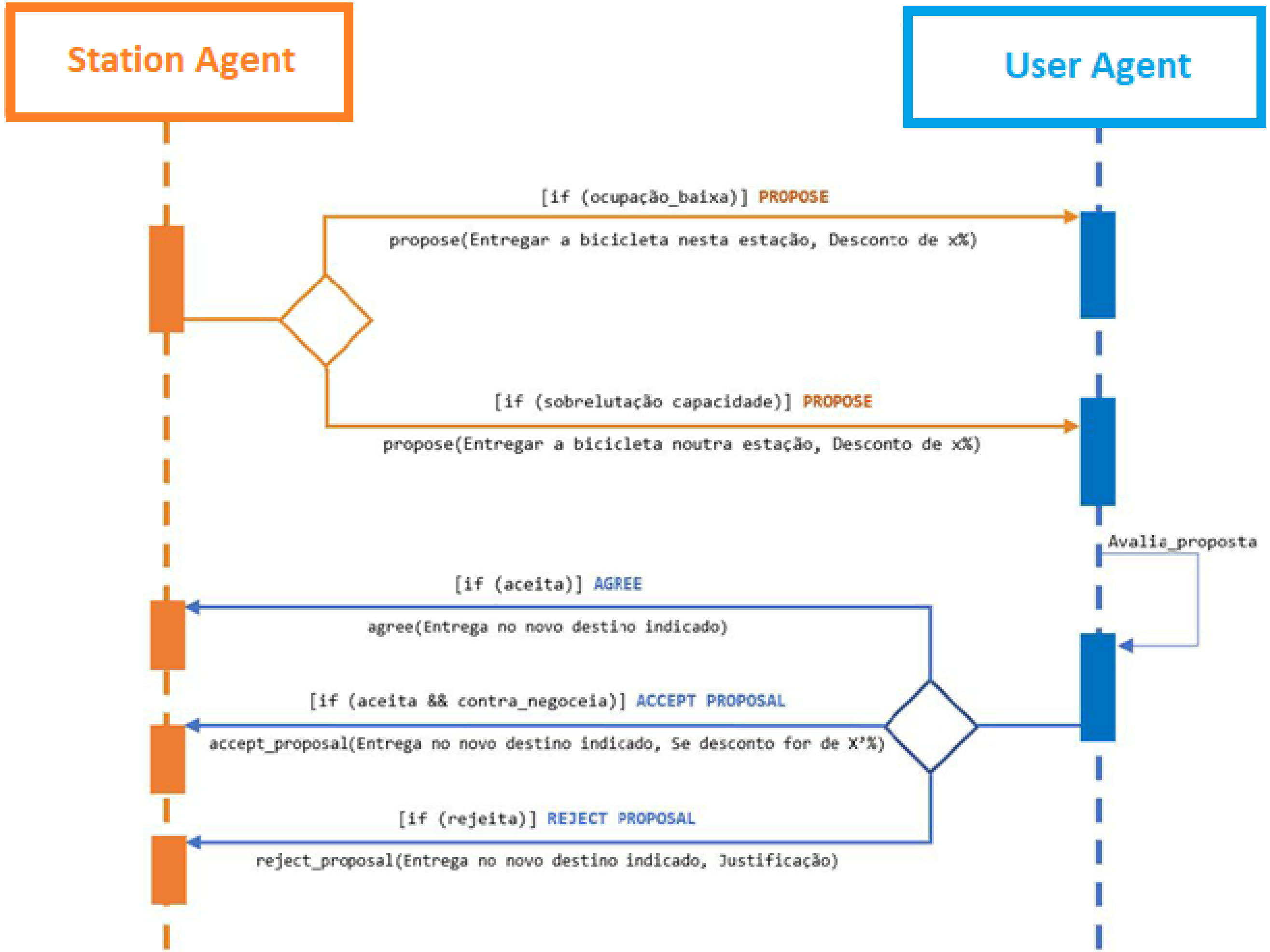
(d) Shift (synchronous)

Different Agent States



(a) Classification (b) Declassification (c) Reclassification

UML Sequence Diagram (Example)



Station Agent

User Agent

[if (ocupação_baixa)] PROPOSE

propose(Entregar a bicicleta nesta estação, Desconto de x%)

[if (sobrelutação capacidade)] PROPOSE

propose(Entregar a bicicleta noutra estação, Desconto de x%)

Avalia_proposta

[if (aceita)] AGREE

agree(Entrega no novo destino indicado)

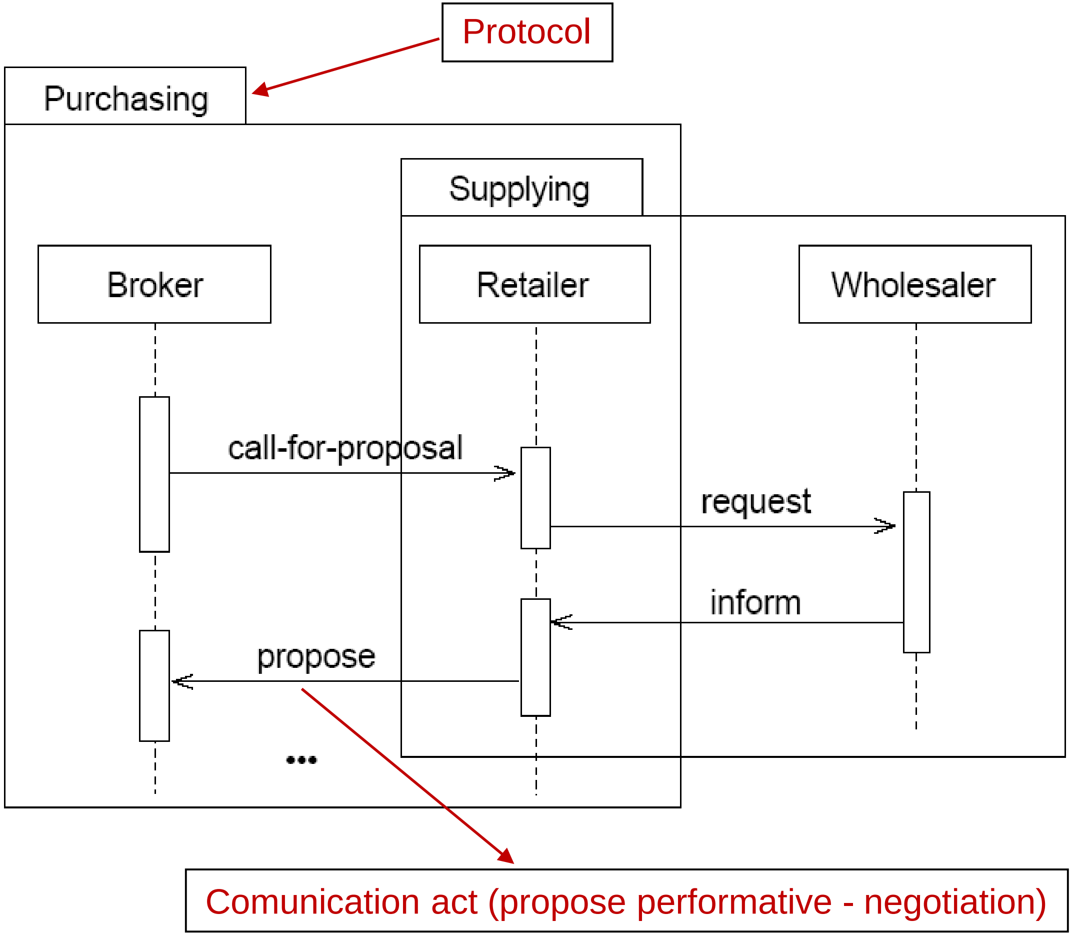
[if (aceita && contra_negoceia)] ACCEPT PROPOSAL

accept_proposal(Entrega no novo destino indicado, Se desconto for de X'%)

[if (rejeita)] REJECT PROPOSAL

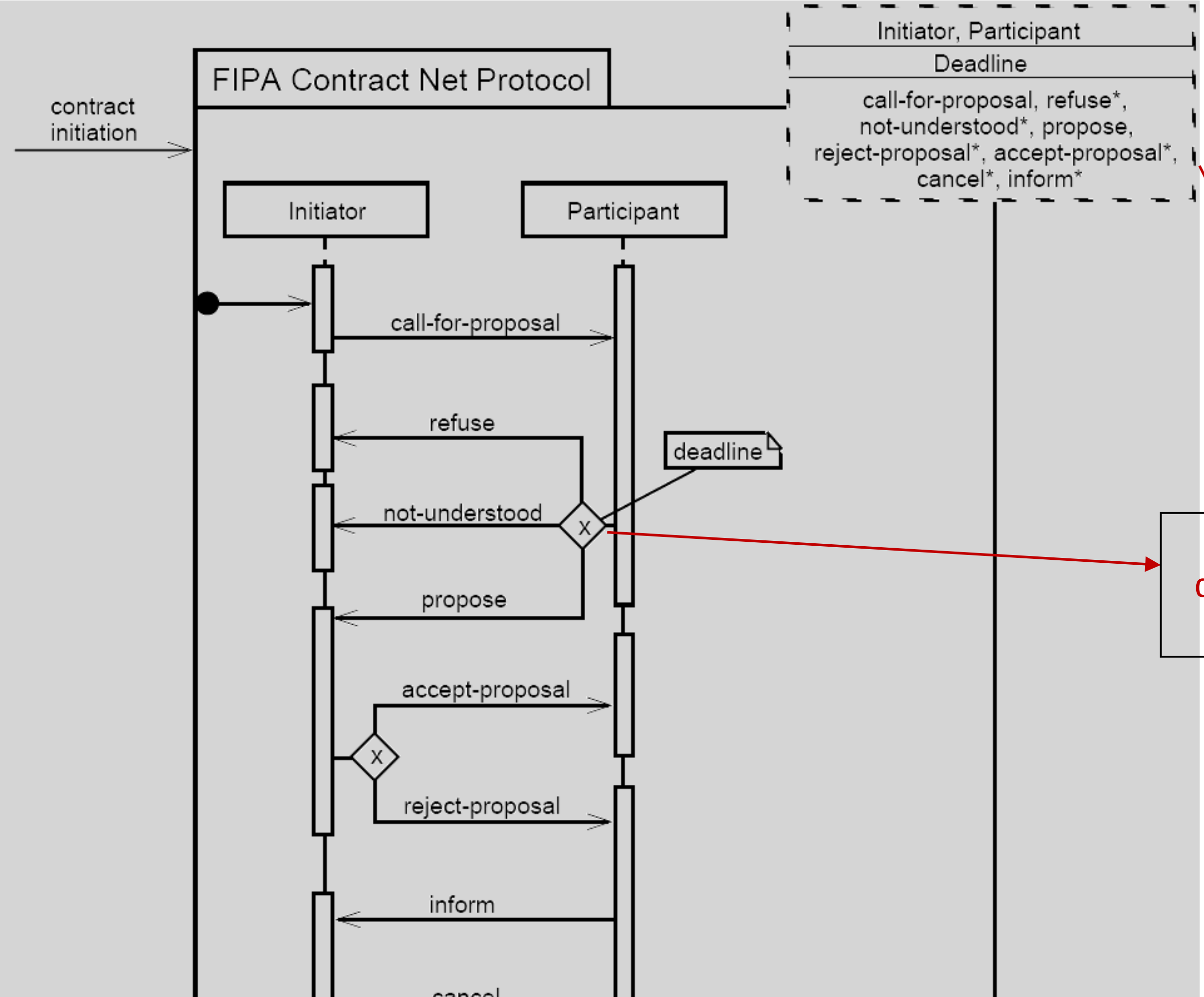
reject_proposal(Entrega no novo destino indicado, Justificação)

Protocol Modelling


Protocol

Comunication act (propose performative - negotiation)

Protocol Modelling (Packages)



Indicates that the protocol can be used as a template to create other protocols

Defines different forms of response

Level 2: Interaction between Agents

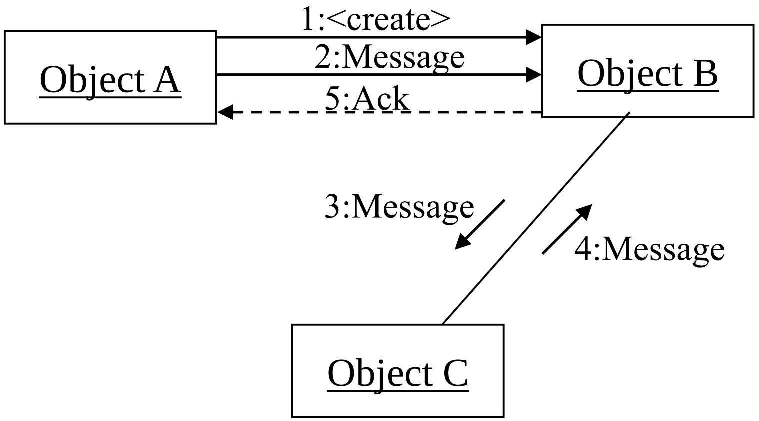
Diagrams:

- Extended Sequence Diagrams
- Collaboration Diagrams
- Activity Diagrams

However, greater system complexity requires more complex graphical presentation:

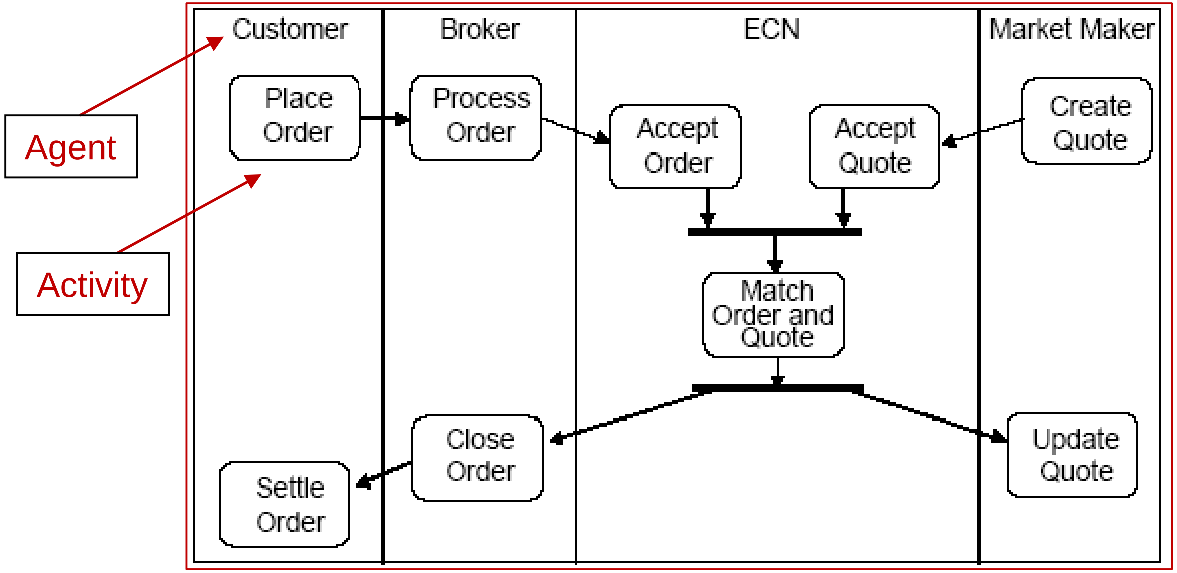
- We often need to express the role an agent plays in his interaction with other agents
- If the number of agents and functions increases, UML diagrams become graphically complex
- UML has no capacity to represent the agent's functions on interaction lines. **Solution: Messages Identify the Agent's Role Transition**

UML Collaboration Diagram (Example)

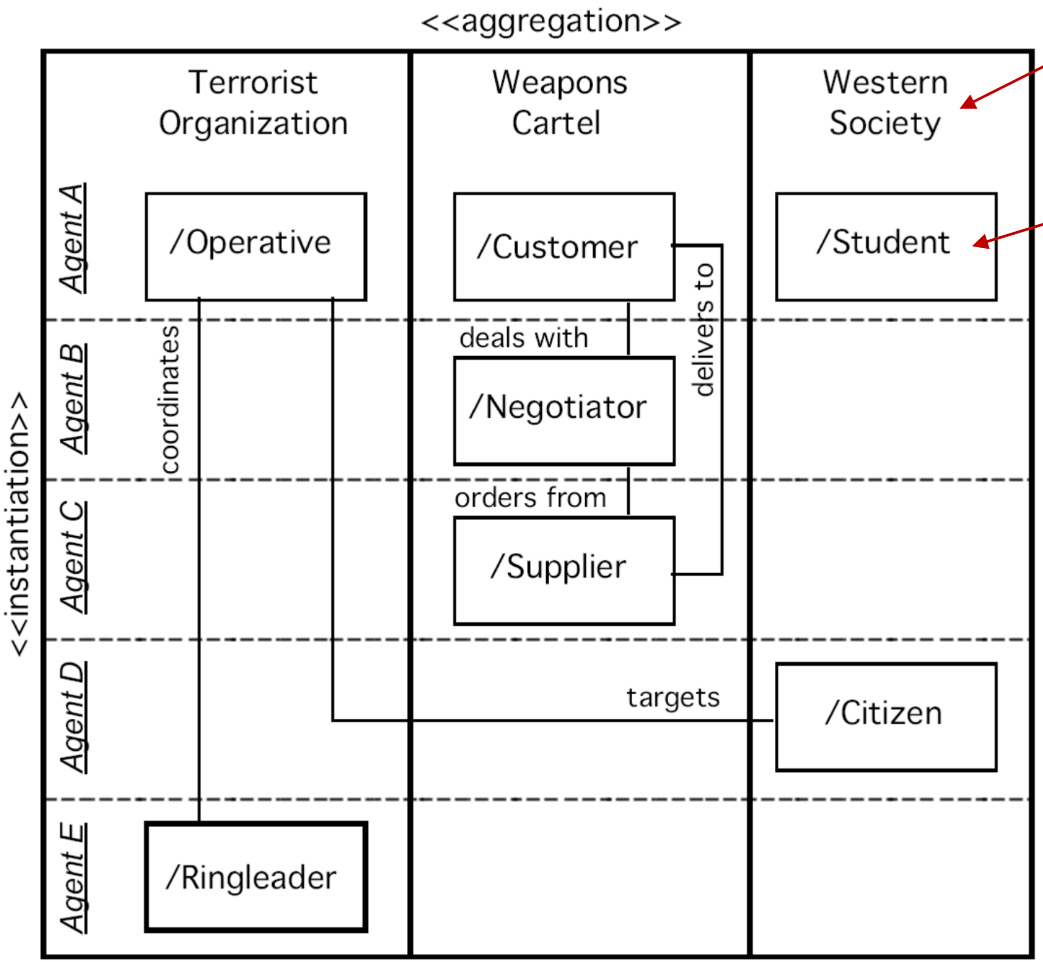


UML Activity Diagram

- Applied to represent the activities associated to a protocol or an agent’s activity
- Useful to plan complex interaction protocols that involve parallel processing



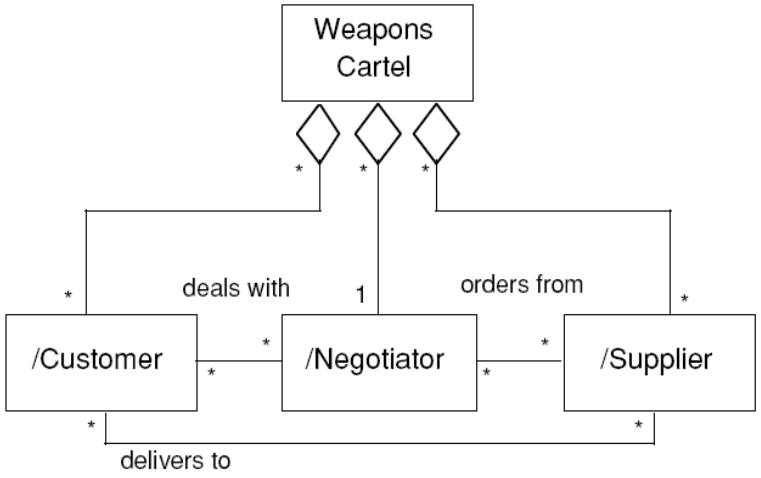
AUML Activity Diagram (Example)



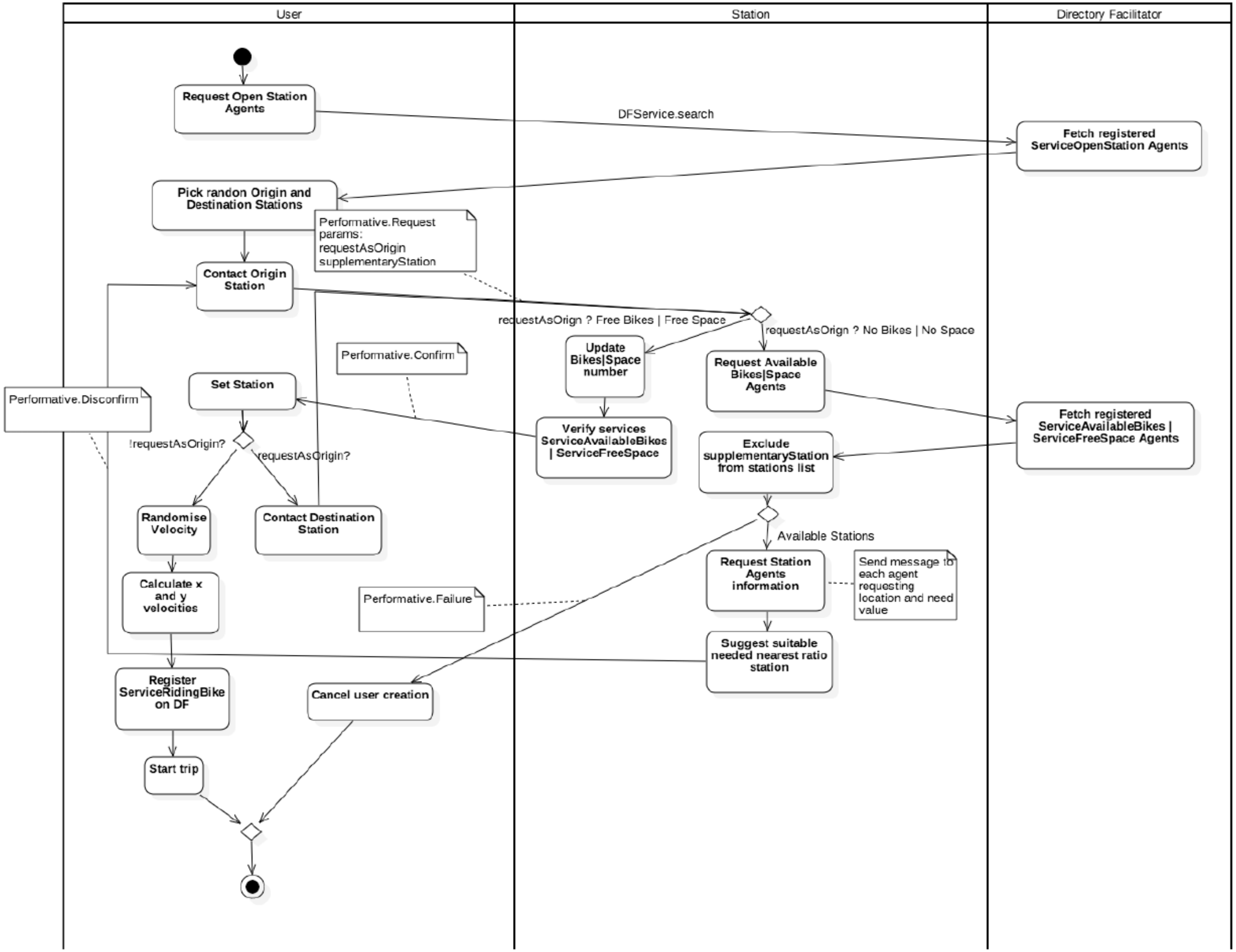
Organization

Role

Object Diagram



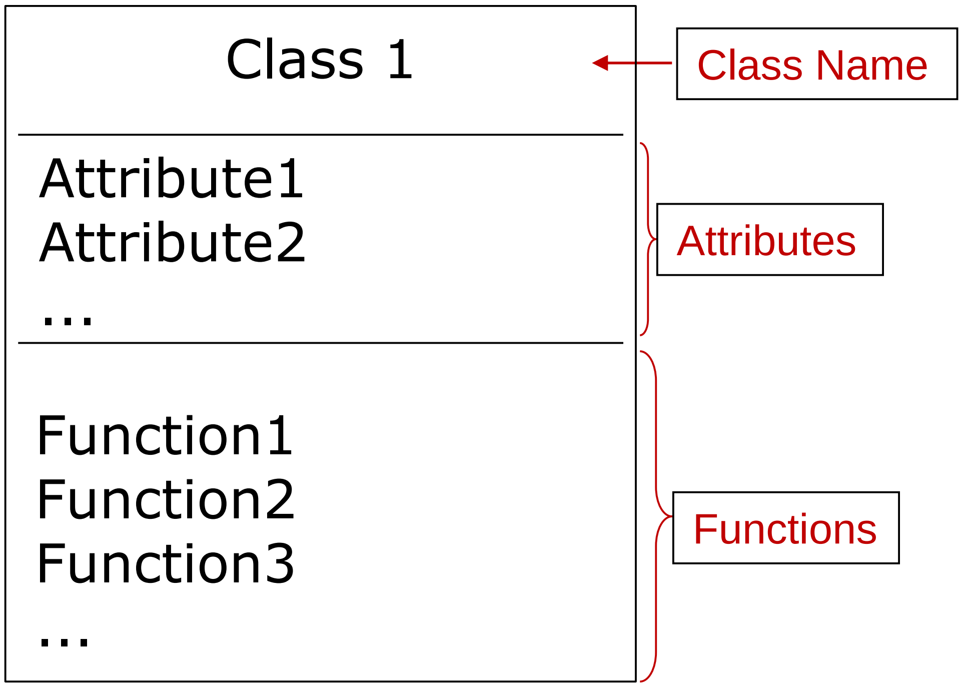
AUML Activity Diagram (Example)


User
Station
Directory Facilitator
Request Open Station Agents
DFService.search
Fetch registered ServiceOpenStation Agents
Pick randon Origin and Destination Stations
Performative.Request params: requestAsOrigin supplementaryStation
Contact Origin Station
requestAsOrign ? Free Bikes | Free Space
requestAsOrign ? No Bikes | No Space
Update Bikes|Space number
Request Available Bikes|Space Agents
Performative.Confirm
Set Station
Performative.Disconfirm
Verify services ServiceAvailableBikes | ServiceFreeSpace
Fetch registered ServiceAvailableBikes | ServiceFreeSpace Agents
Exclude supplementaryStation from stations list
!requestAsOrigin?
requestAsOrigin?
Randomise Velocity
Contact Destination Station
Available Stations
Request Station Agents information
Send message to each agent requesting location and need value
Calculate x and y velocities
Performative.Failure
Suggest suitable needed nearest ratio station
Register ServiceRidingBike on DF
Cancel user creation
Start trip

Class Diagram

Class Diagrams are used to:

- Model the problem's dominion
- Model the classes implementation



Class 1

Attribute1
Attribute2
...

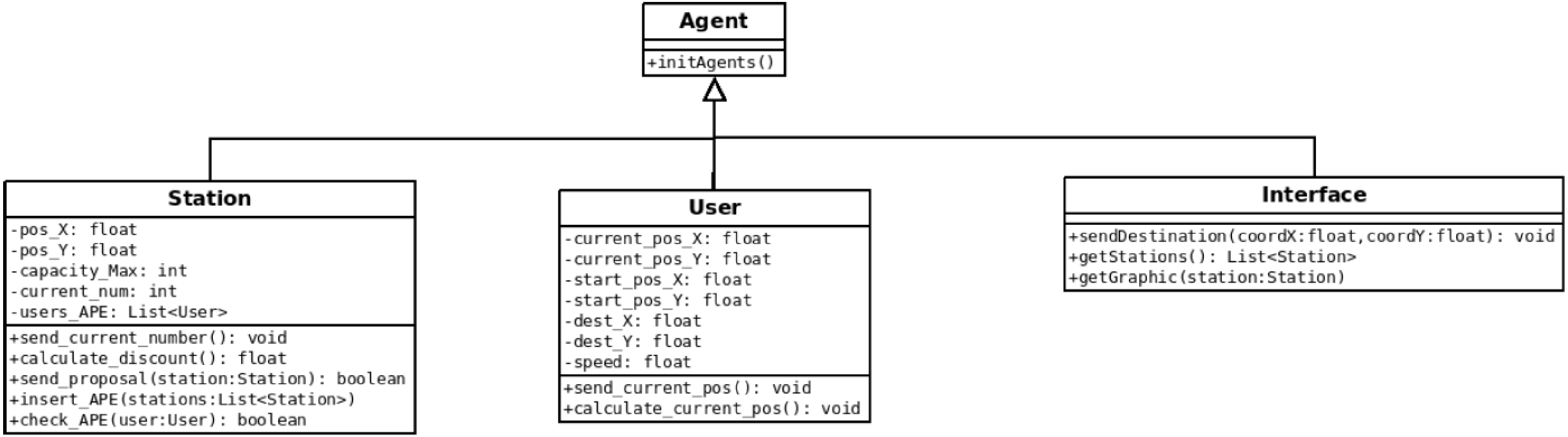
Function1
Function2
Function3
...

Class Name

Attributes

Functions

Class Diagram (Example)



Agent

+initAgents()

Station

-pos_X: float
-pos_Y: float
-capacity_Max: int
-current_num: int
-users_APE: List<User>

+send_current_number(): void
+calculate_discount(): float
+send_proposal(station:Station): boolean
+insert_APE(stations:List<Station>)
+check_APE(user:User): boolean

User

-current_pos_X: float
-current_pos_Y: float
-start_pos_X: float
-start_pos_Y: float
-dest_X: float
-dest_Y: float
-speed: float

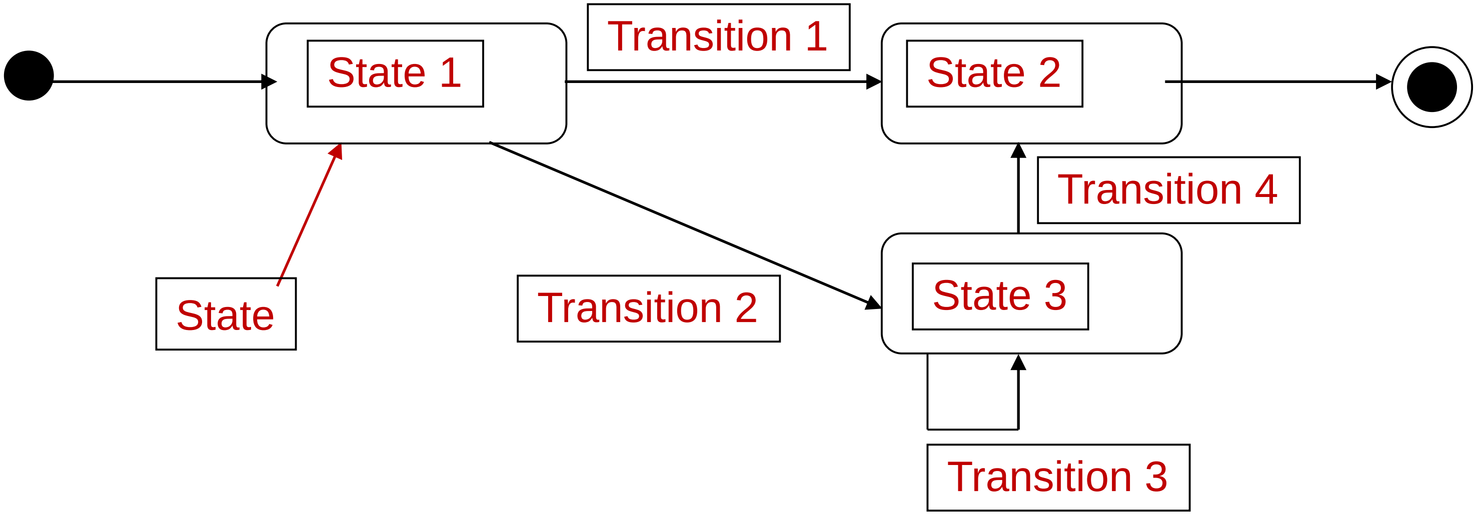
+send_current_pos(): void
+calculate_current_pos(): void

Interface

+sendDestination(coordX:float,coordY:float): void
+getStations(): List<Station>
+getGraphic(station:Station)

UML State Diagram

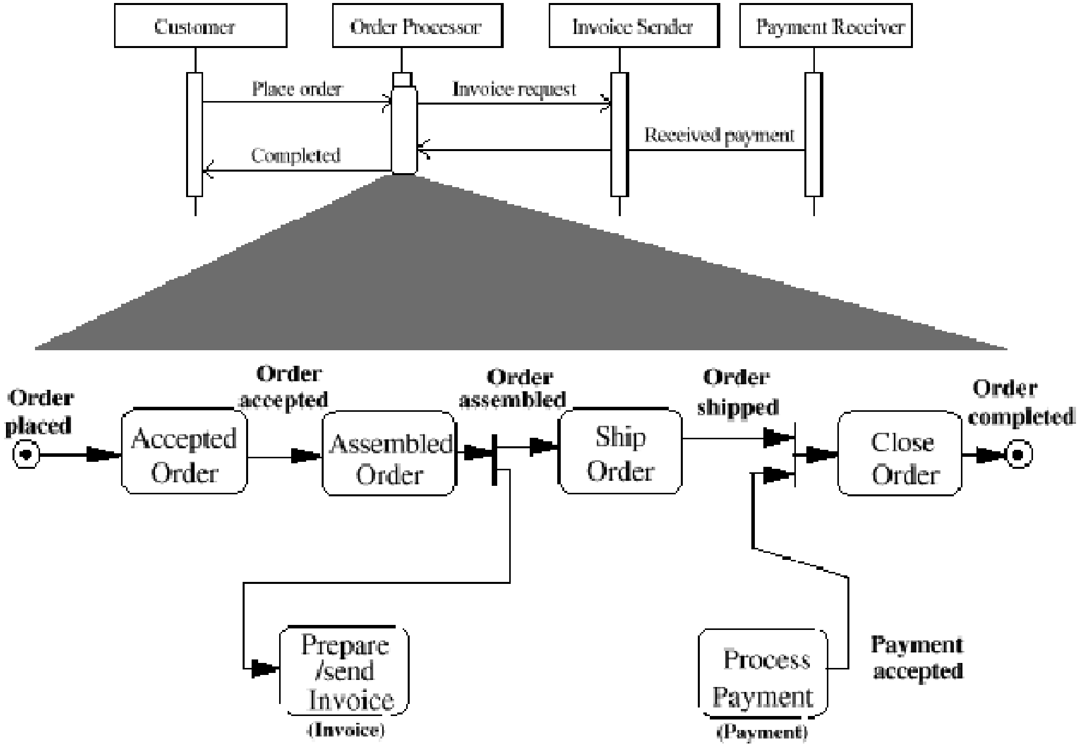
- Applied to represent the different states of a system and its transactions



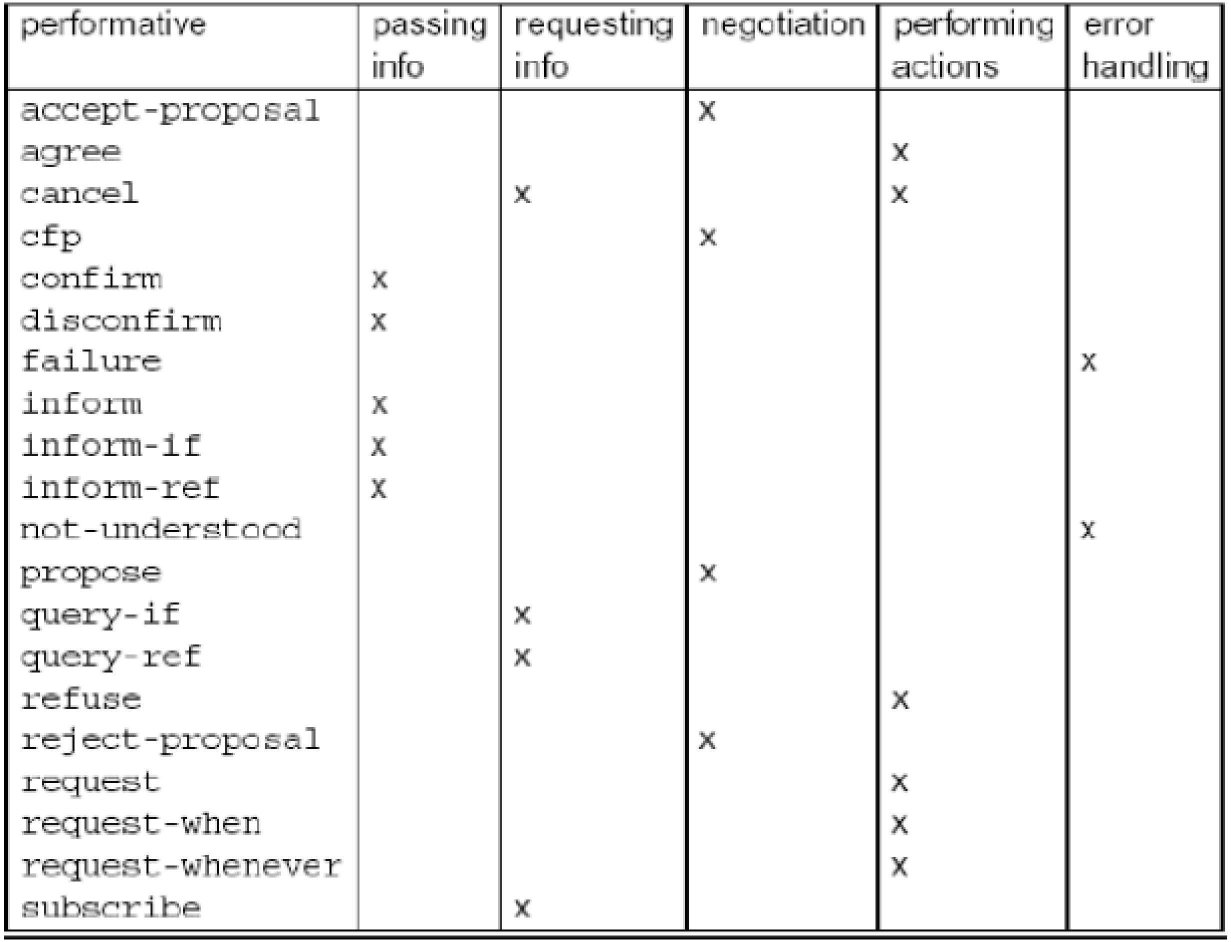
Level 3: Agent’s Internal Process

Level 3: Representation of the agent’s internal processing

- **Example:** Internal processing of Agent’s “Order Processor”



Extra (Performatives)



performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

Conclusions

UML extension mechanisms provide formalisms to specify Agents interaction to several levels:

- Specify protocols as a whole
- Express interaction patterns between Agents
- Express the internal behaviour of an Agent
- Formalization of Agents requirements and APIs important for the development & implementation of Multi-agent Systems

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Agentes Inteligentes
2019/2020

Paulo Novais, Filipe Gonçalves