Universidade do Minho
Escola de Engenharia
Departamento de Informática

Agent UML

Agentes e Sistemas Multiagente

Pedro Oliveira, Paulo Novais

Software Agent:

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

Multi-Agent System:

- Group of interacting agents that understand and coordinate with each other in global tasks involving cooperation or competition

Agents as **Active Object Extensions:**

- Passive Agents (Accept / Decline Orders)
- Proactive Agents (start activity without external invocation)

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
- Interaction Protocol Representation for Agents

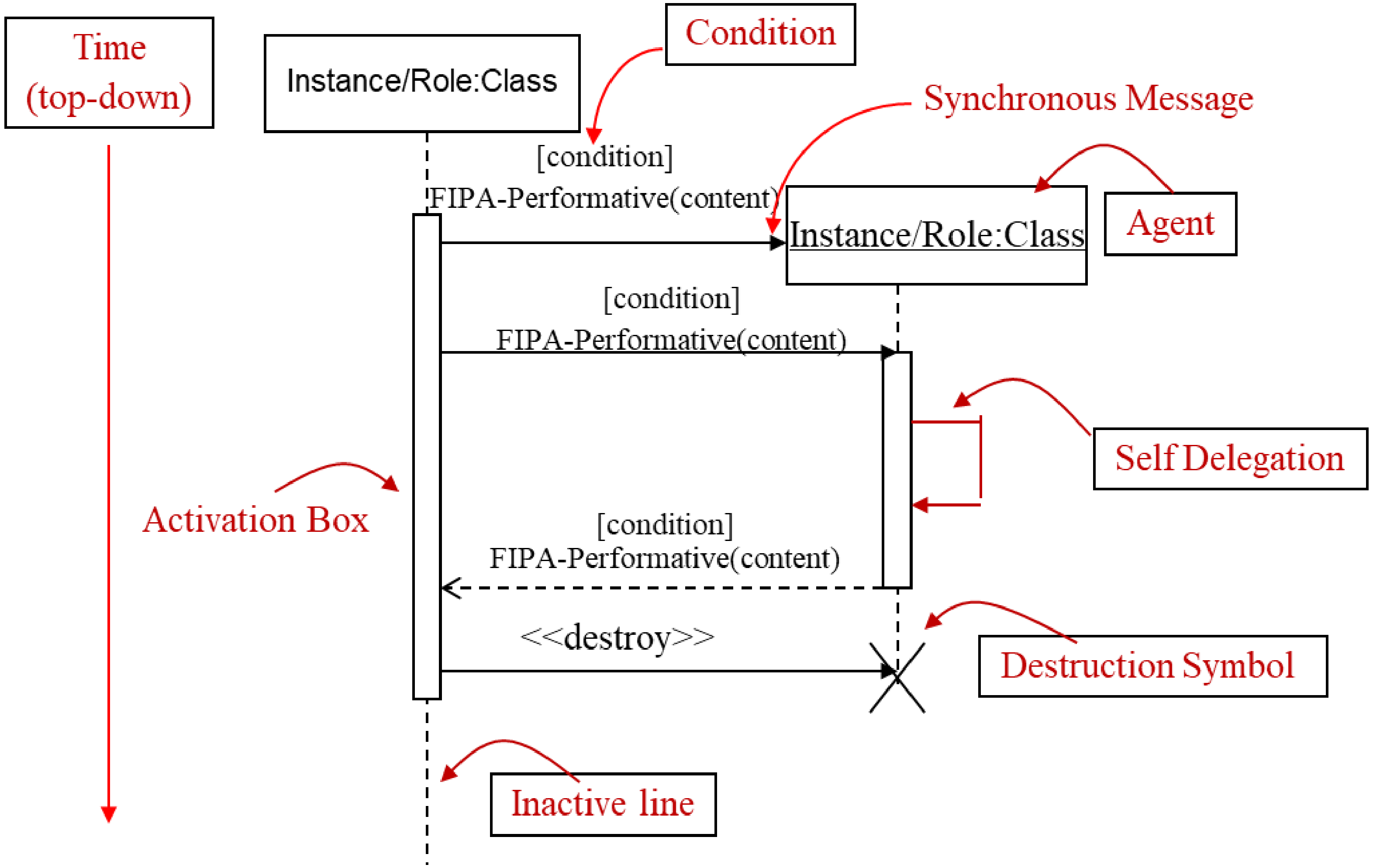
AUML models application:

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

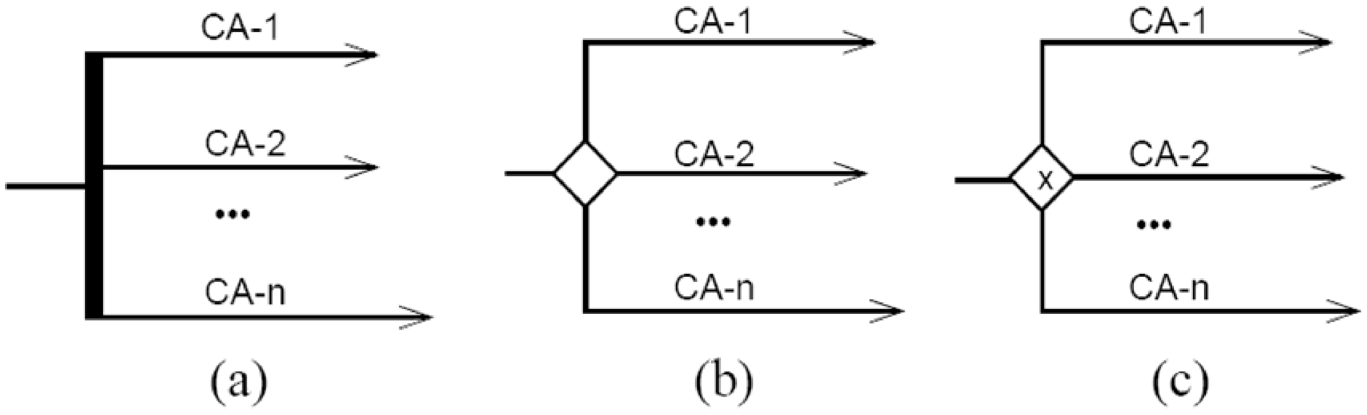
AUML takes a layered approach to protocols:

- **Level 1:** Represents the general protocol (tempaltes, modeling diagrams)
- **Level 2:** Represent agent interactions (sequence, collaboration, activity diagrams)
- **Level 3:** Represent internal agent processing (activity diagrams and statecharts)

General Diagram Structure



Message Transmission

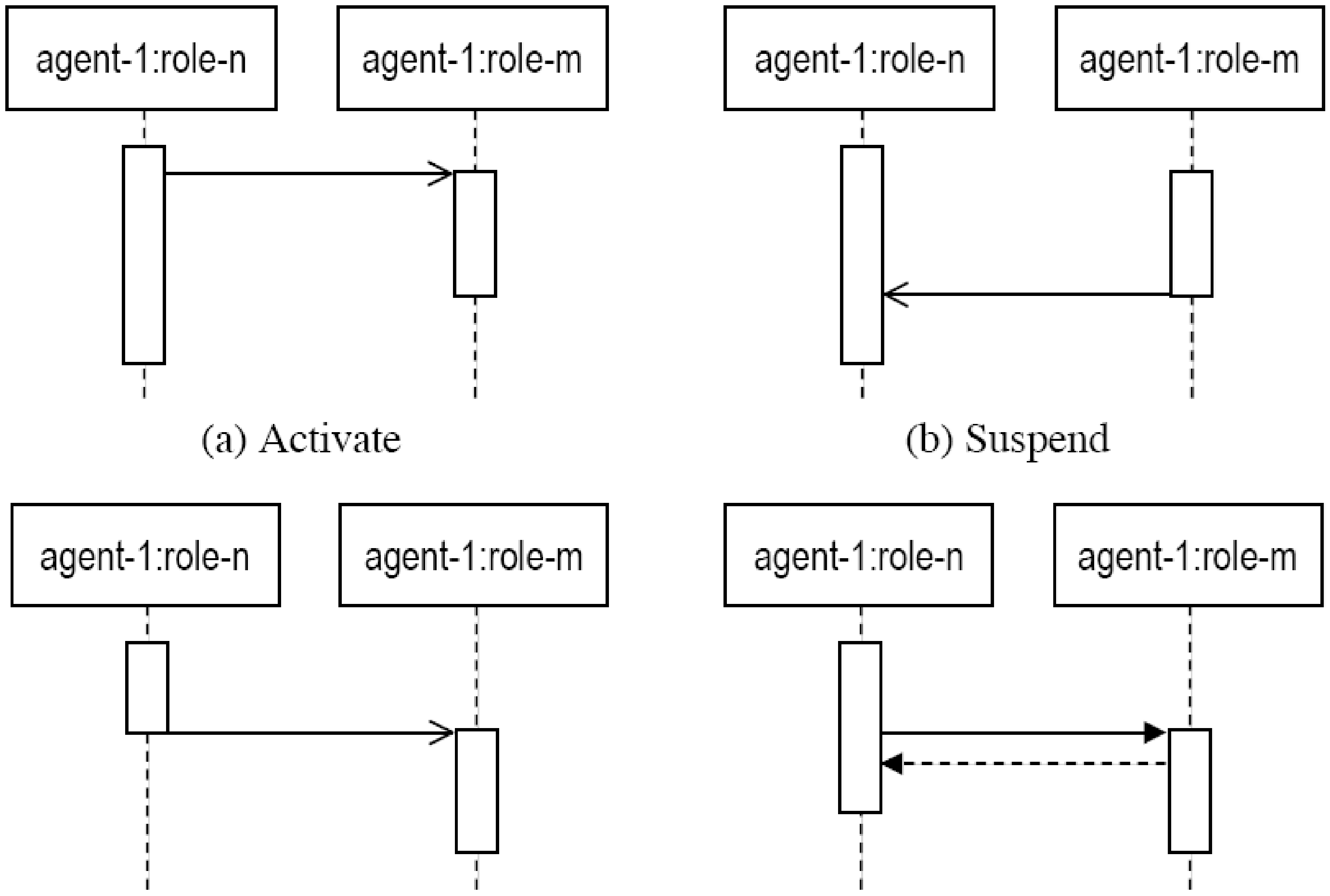


AND /
Inclusive

OR /
Exclusive

XOR /
Exclusive or

Different Agent States



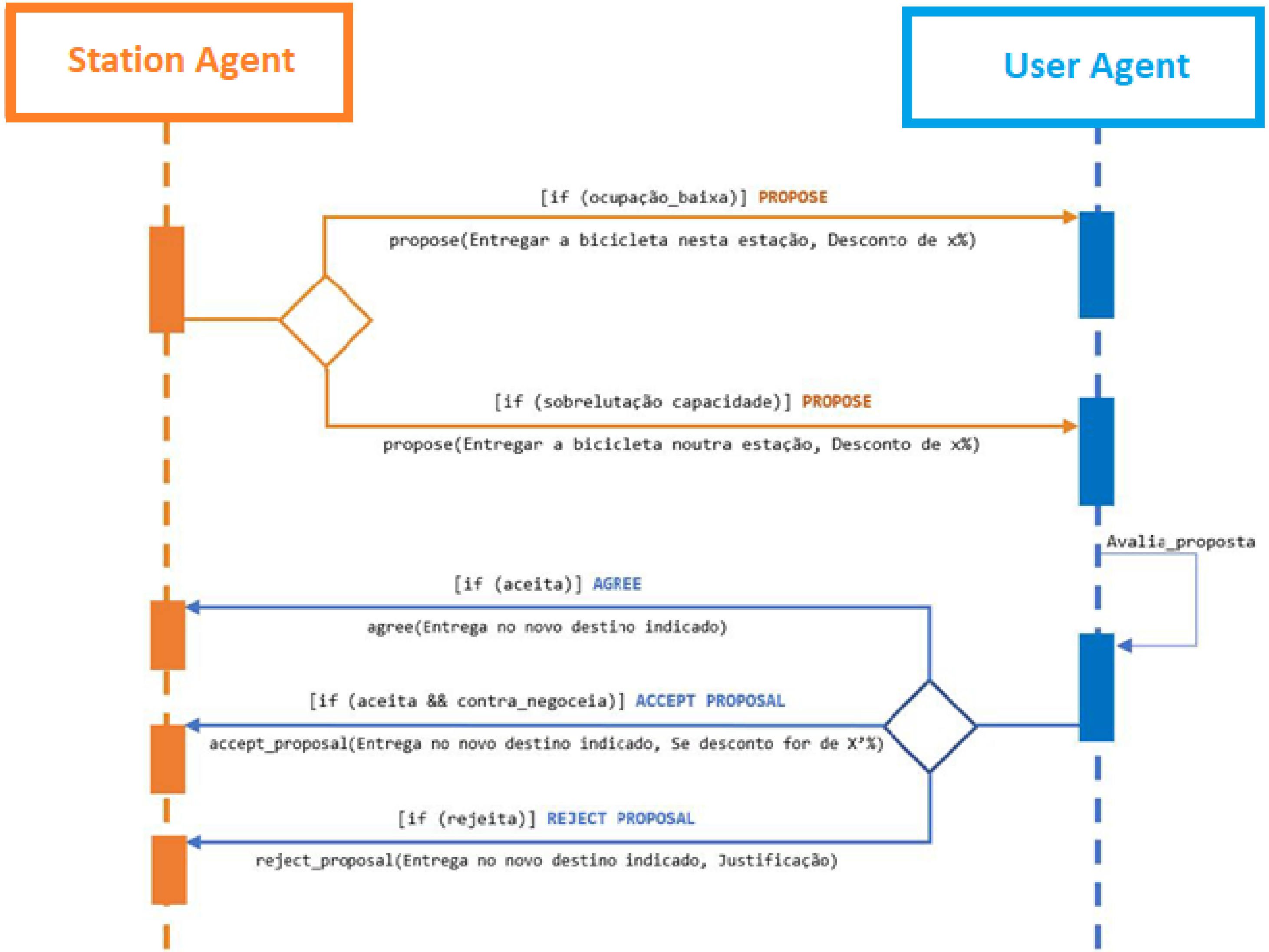
(a) Activate

(b) Suspend

(c) Shift (asynchronous)

(d) Shift (synchronous)

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)

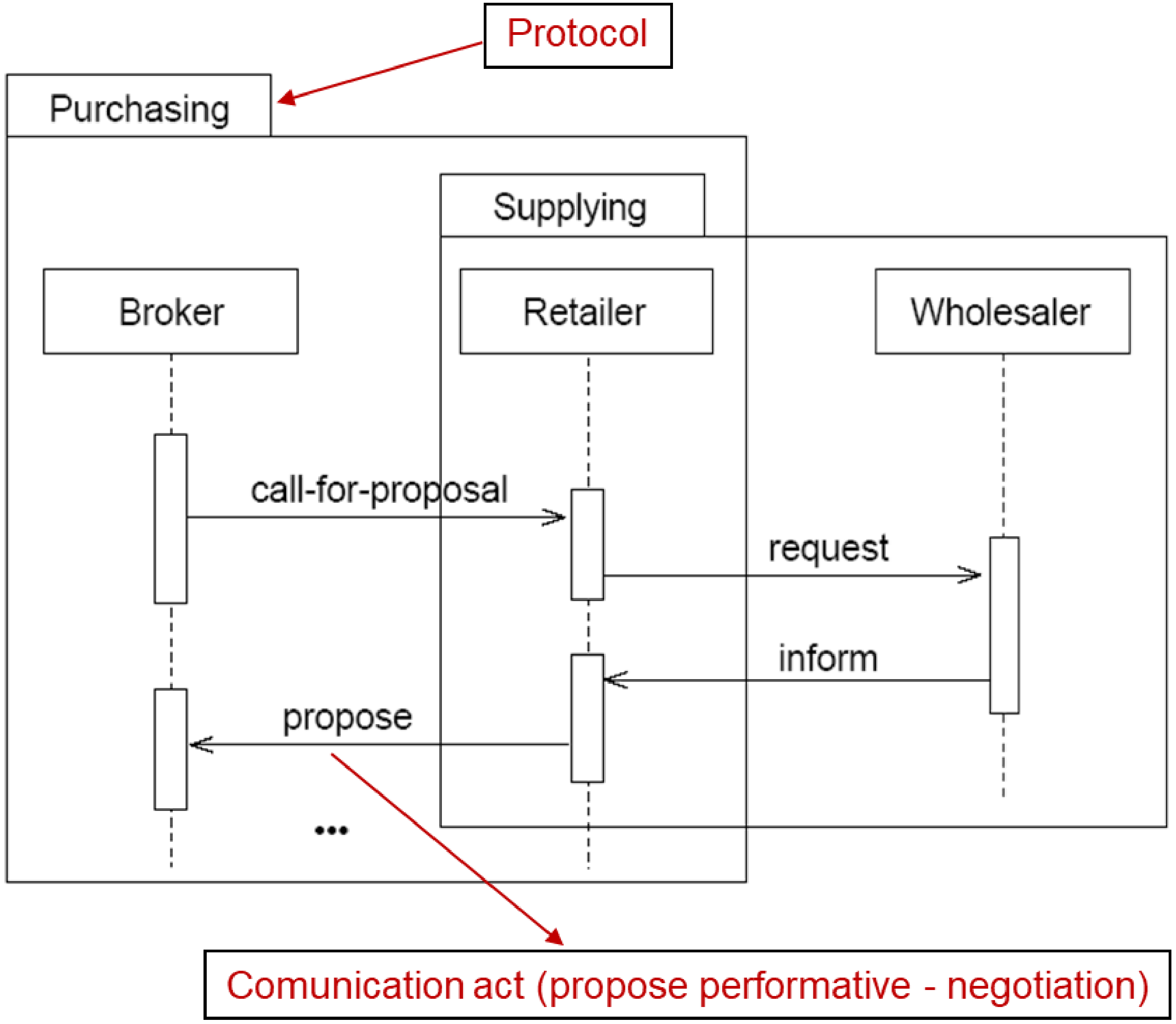
[if (aceita && contra_negoceia)] ACCEPT PROPOSAL

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

[if (rejeita)] REJECT PROPOSAL

reject_proposal(Entrega no novo destino indicado, Justificação)

Packages Modelling



Interaction

Interaction between Agents

Diagrams:

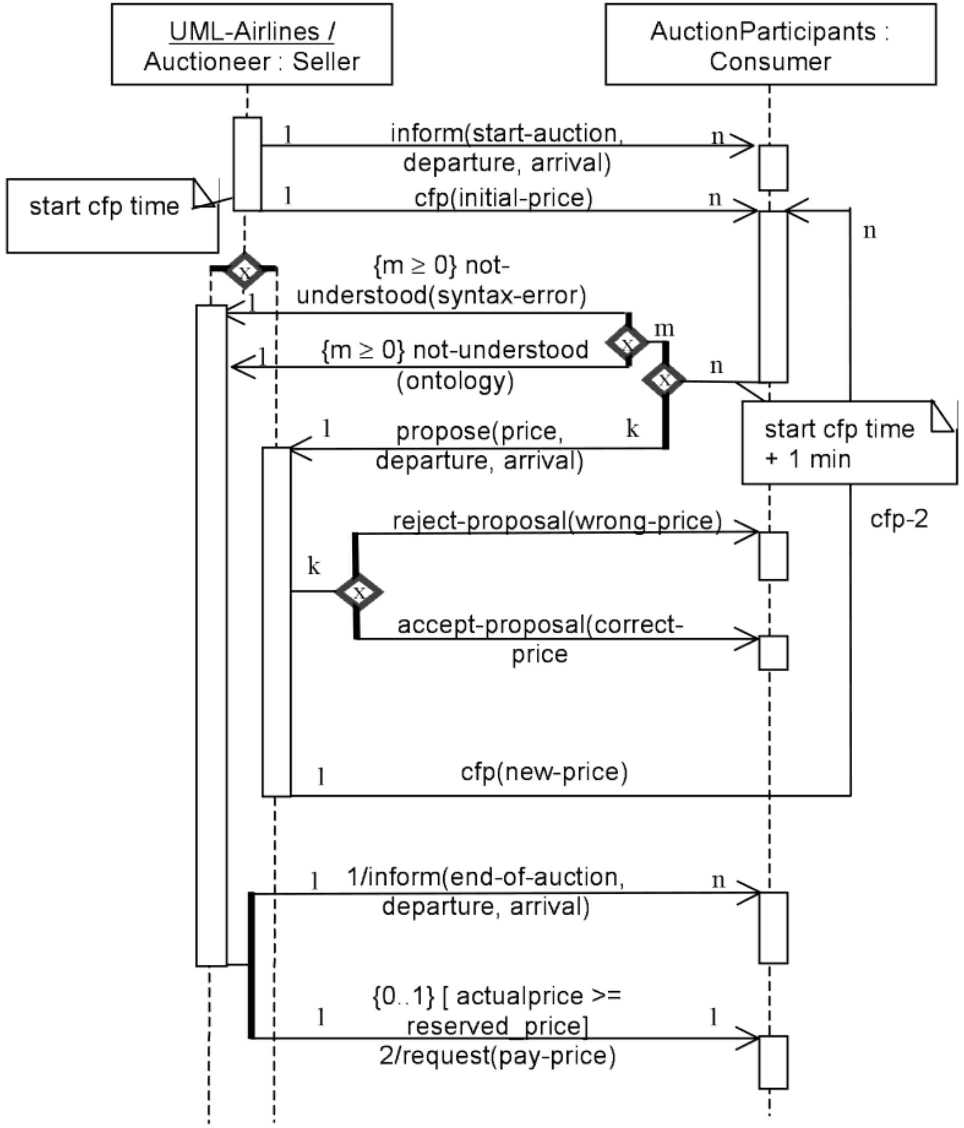
- Sequence Diagrams
- Activity Diagrams
- Collaboration Diagrams

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**

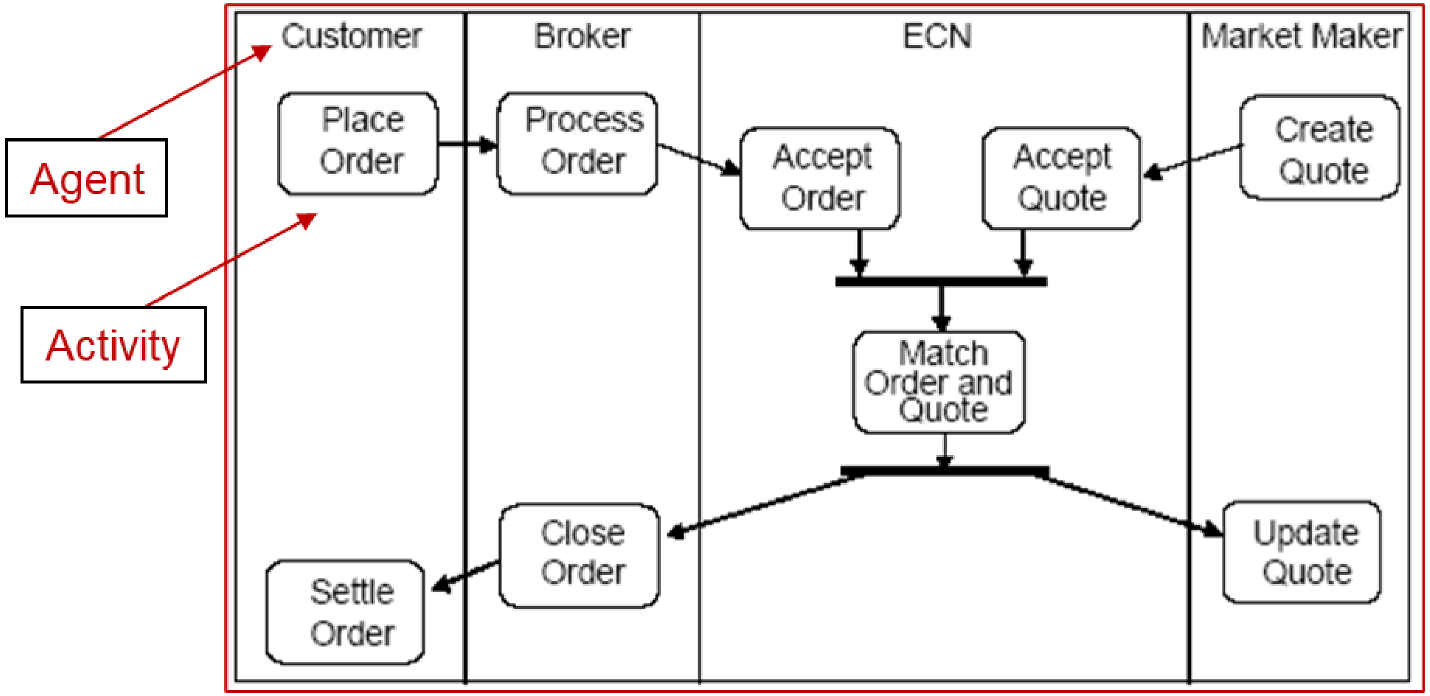
Sequence Diagram

- 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

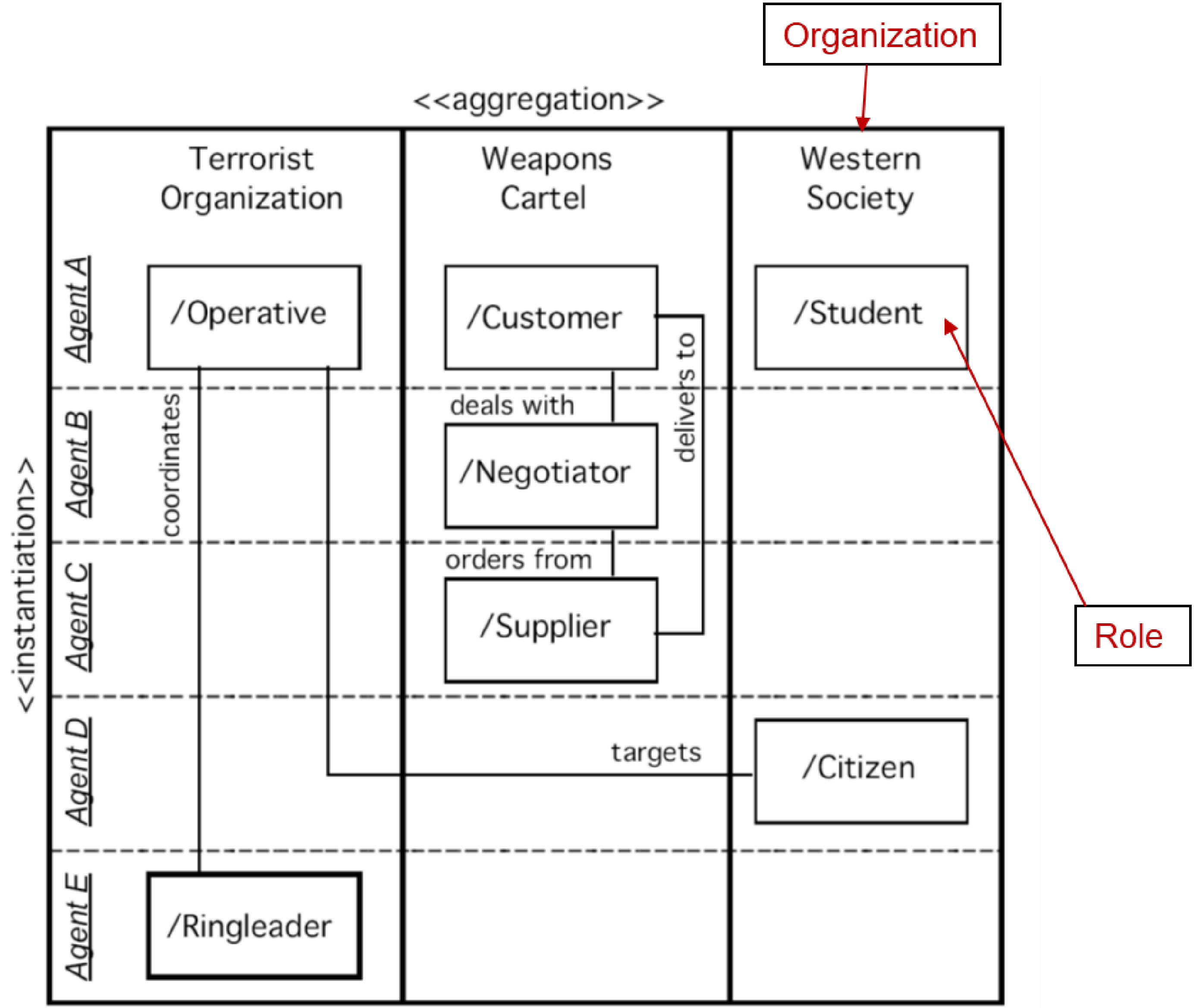

UML-Airlines / Auctioneer : Seller
AuctionParticipants : Consumer
1
inform(start-auction, departure, arrival)
n
start cfp time
1
cfp(initial-price)
n
n
{m ≥ 0} not-understood(syntax-error)
1
m
{m ≥ 0} not-understood (ontology)
1
n
start cfp time + 1 min
1
propose(price, departure, arrival)
k
reject-proposal(wrong-price)
cfp-2
k
accept-proposal(correct-price
1
cfp(new-price)
1
1/inform(end-of-auction, departure, arrival)
n
{0..1} [actualprice >= reserved_price]
1
2/request(pay-price)
1

Activity Diagram

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

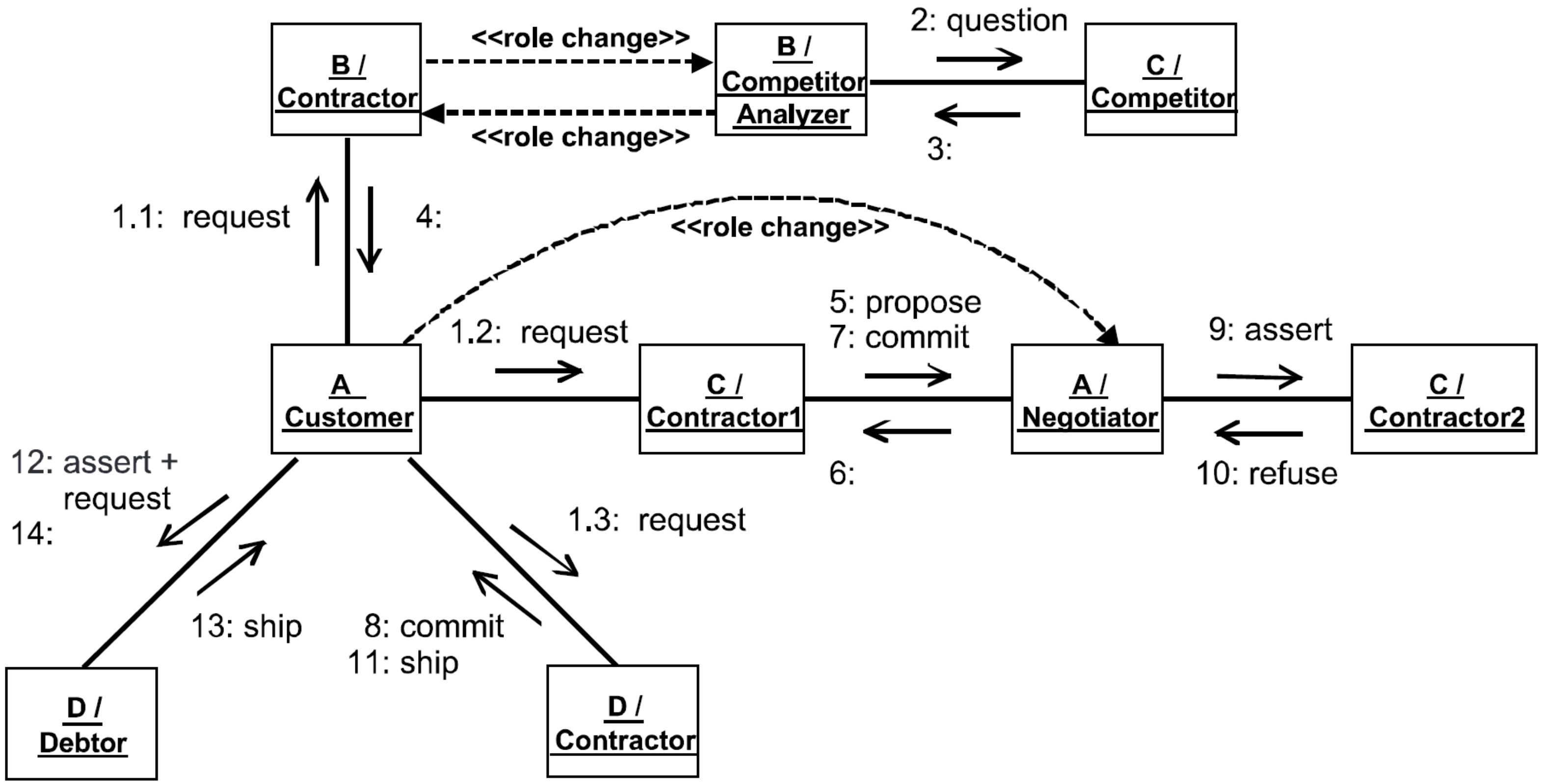


Activity Diagram



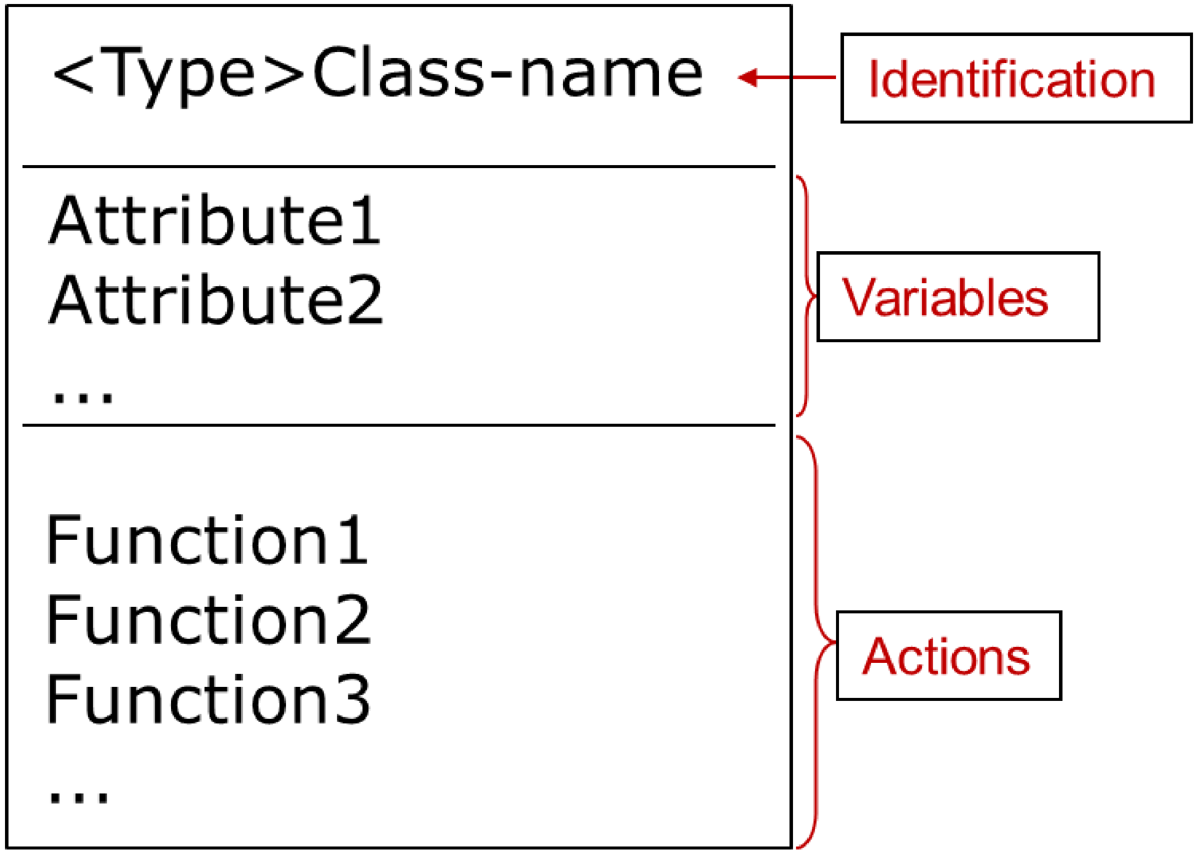
Collaboration/ Communication Diagram

- Similar to sequence diagram
- Presents a clear and understandable representation of the system
- Sequence of interactions are numbered on the collaboration diagram

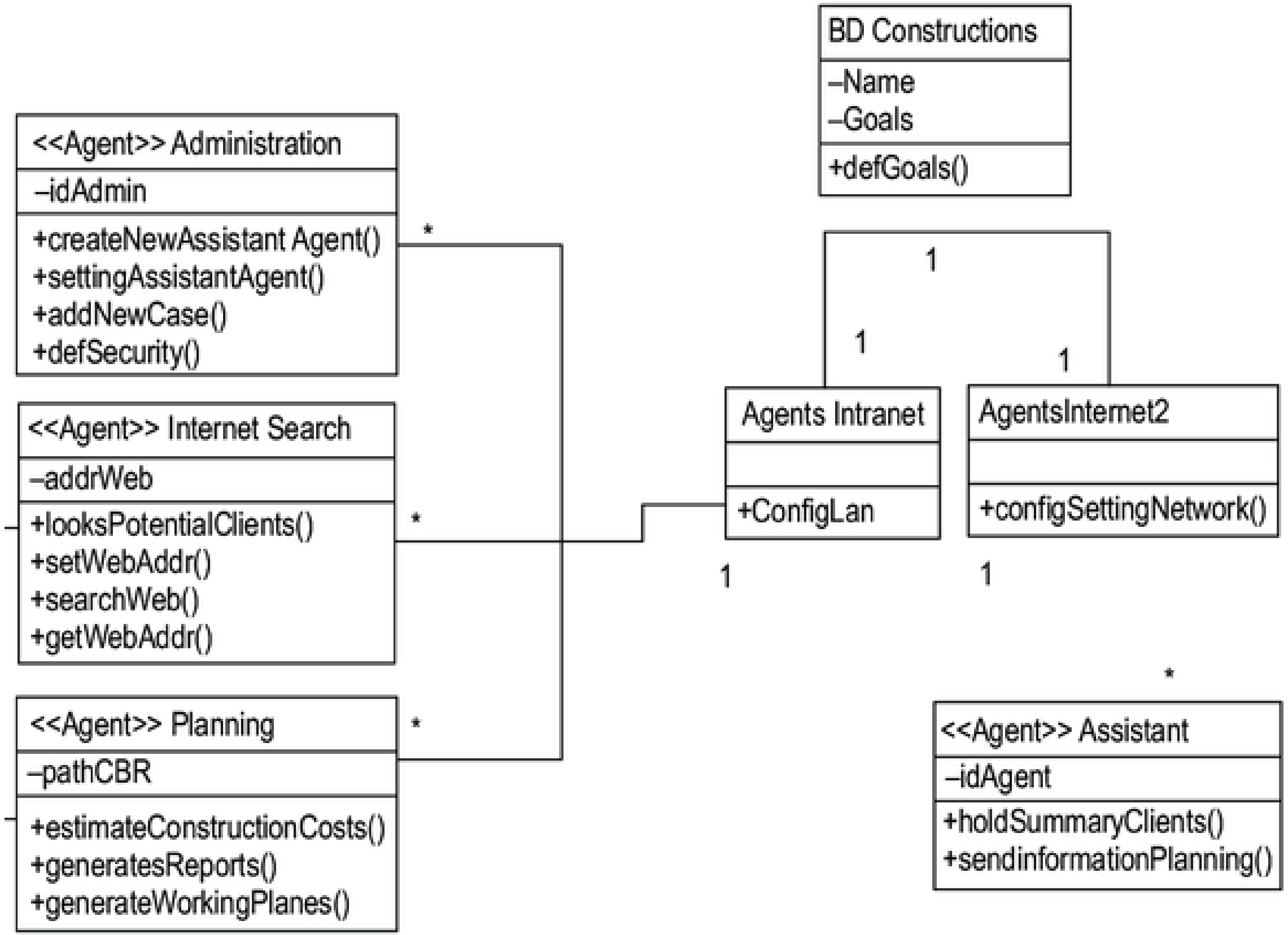


Class Diagram

Class Diagrams are used to model the problem's dominion and agent-class implementation



Class Diagram



<<Agent>> Administration
–idAdmin
+createNewAssistant Agent()
+settingAssistantAgent()
+addNewCase()
+defSecurity()

<<Agent>> Internet Search
–addrWeb
+looksPotentialClients()
+setWebAddr()
+searchWeb()
+getWebAddr()

<<Agent>> Planning
–pathCBR
+estimateConstructionCosts()
+generatesReports()
+generateWorkingPlanes()

BD Constructions
–Name
–Goals
+defGoals()

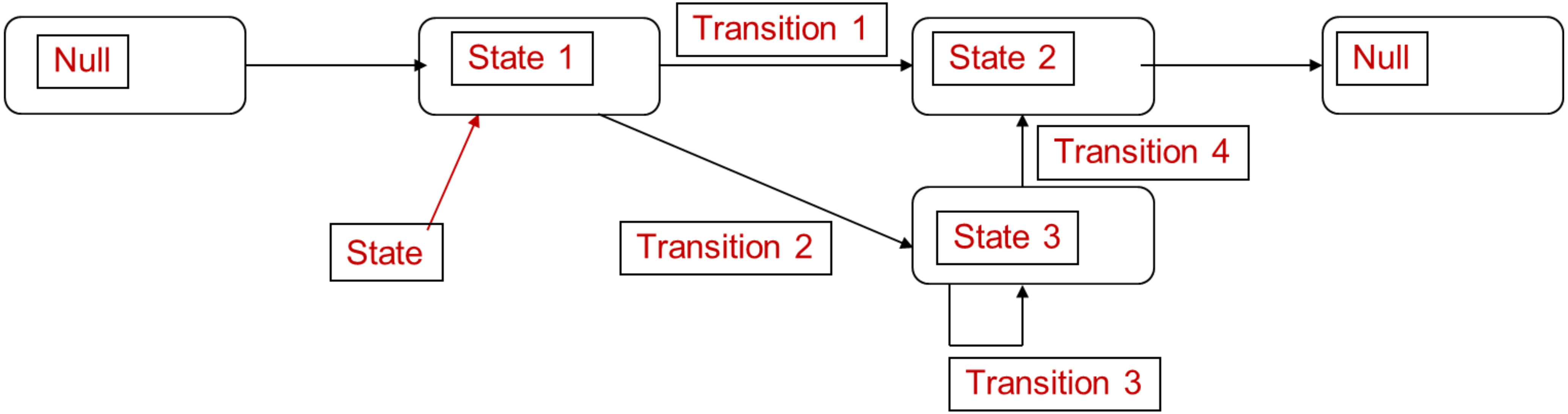
Agents Intranet
+ConfigLan

AgentsInternet2
+configSettingNetwork()

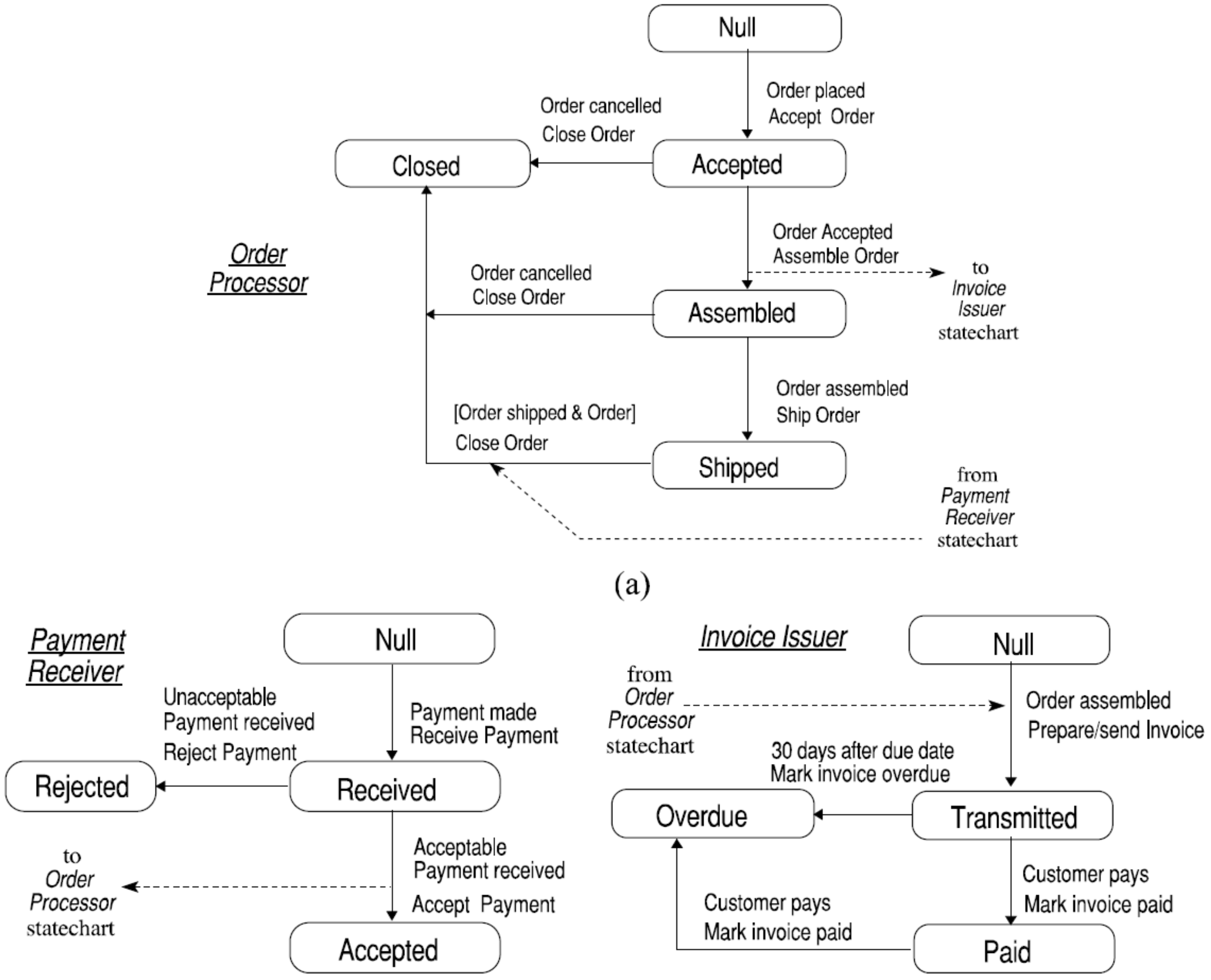
<<Agent>> Assistant
–idAgent
+holdSummaryClients()
+sendinformationPlanning()

Statecharts

- The internal processing of a single agent can be expressed as statecharts
- Statecharts specify order processing behaviour for the different agents



Statecharts



(a)

(b)

(c)

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

References and Electronic Resources

- J. Palanca, A. Terrasa, V. Julian and C. Carrascosa, "SPADE 3: Supporting the New Generation of Multi-Agent Systems," in IEEE Access, vol. 8, pp. 182537-182549, 2020

- AUML Manual
  - http://www.jamesodell.com/ExtendingUML.pdf

- Smart Python Agent Development Environment (SPADE)
  - https://spade-mas.readthedocs.io/en/latest/

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Agent UML

Agentes e Sistemas Multiagente

Pedro Oliveira, Paulo Novais