INFRASTRUCTURES FOR THE ENVIRONMENT OF MULTIAGENT SYSTEMS

Mirko Viroli - Tom Holvoet - Alessandro Ricci -
Kurt Schelfthout - Franco Zambonelli

Team Winter Slayers:
Trieu Hung Tran - Minal Khatri

Overview

- Introduction
- Background
- Supporting the MAS environment with infrastructures
- A survey of MAS infrastructures for environment
- Conclusions

Introduction

- Endorse a general viewpoint where the environment of a multiagent system is seen as a set of basic bricks called *environment abstractions*, which
    - Provide agents with services useful for achieving individual or social goals
    - Are supported by some underlying software infrastructure managing their creation and exploitation

- Focus the survey on the opportunities that environment infrastructures provide to system designers when developing multiagent applications.

Background

- The MASs approach to software development is based on the idea that the emerging characteristics of modern, complex software systems can be suitably tackled by modeling software as a composition of **autonomous** entities (agents) whose behaviour can be understood and designed in terms of **“achieving a goal”**.
- For agents to form a MAS they should be situated in a common computational and/or physical environment, where agent **interactions with each other** and with resources is enabled.
- A middleware software layer for environments is understood as an infrastructure providing some class of environment abstractions at run-time, making them available to engineers for developing the multiagent application at hand.

Supporting the MAS environment with infrastructures

- Environment abstractions
- Infrastructures for environment

Environment abstractions

- An environment design maps a portion of the system functionality to the software elements in which the environment is decomposed.

- The environment is not made by entities which are autonomous and goal-oriented like agents. Rather, they are function-oriented, in the sense that their main scope in the multiagent system is to provide a function to agents.
  - The environment constituents are used by the agents and never the opposite.

- The environment should feature reactivity, function-oriented design and behaviour, partial (or sometimes complete) controllability, and so on.

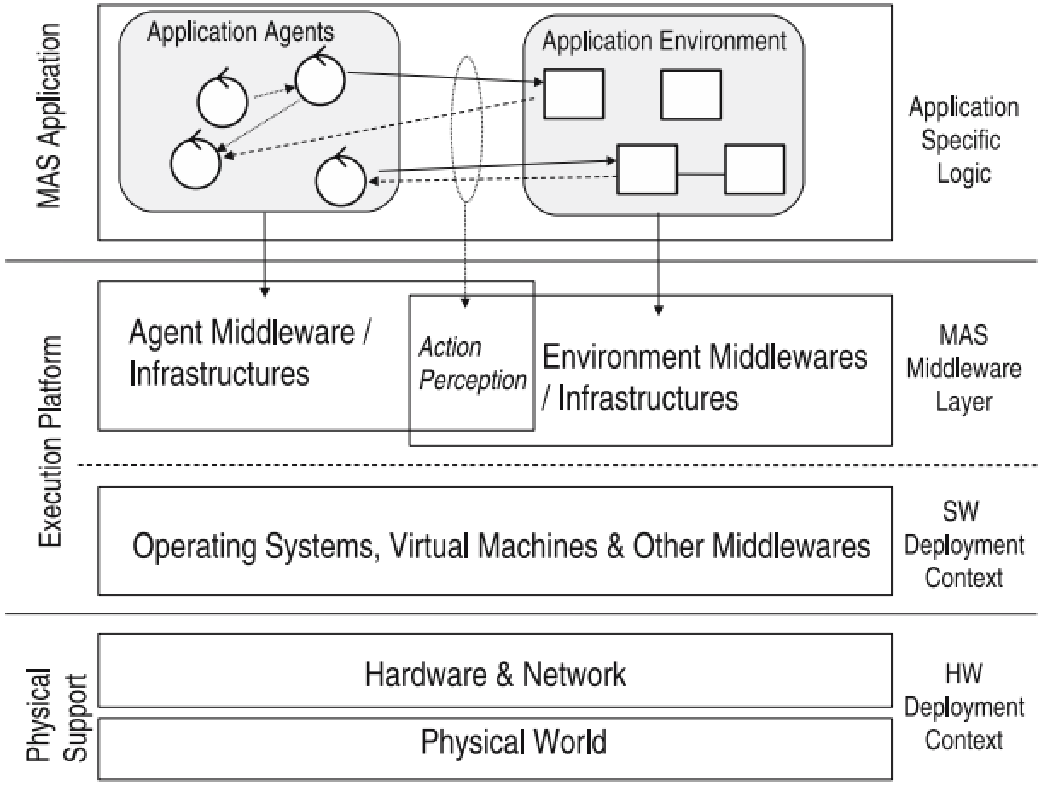
Environment abstractions

- An environment abstraction is an entity of the environment encapsulating some function or service for the agents.
- Environment abstractions are seen as loci where the designer can enforce rules, norms, and functions, regulating the agent social behaviour.

Infrastructures for environment

- It is almost mandatory to develop a middleware, that is a software layer handling the life-cycle of environment abstractions and their interaction with agents.
    - Building them from scratch each time is clearly not a viable approach.
- There can be two kinds of MAS middleware:
    - An infrastructure for agents providing agent life-cycle, management, and often some other core services like direct communication.
    - One (or more) environment infrastructures, each providing some class of environment abstractions.

Infrastructures for environment



A survey of MAS infrastructures for environment

- Tuple centres in TuCSoN
- Co-fields in TOTA
- Mobile tuple spaces in Lime
- Virtual environments in AGV application
- Digital pheromone infrastructure
- AMELI middleware for electronic institutions

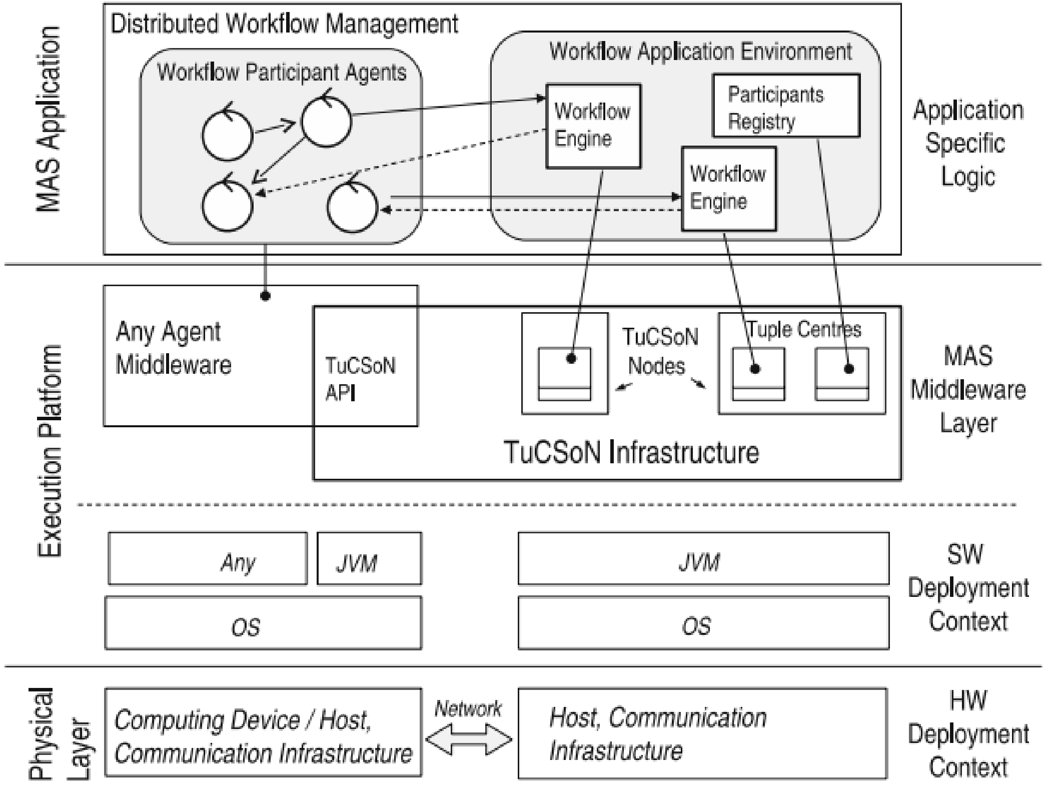
Tuple centres in TuCSoN

- Tuple spaces are seen as share repositories where agents can insert and retrieve information chunks (tuples), thus allowing for an indirect communication mechanism.
- **Tuple centres** are spaces of logic tuples and can be programmed to react to certain interaction events and accordingly modify the state of the space through a chain of internal transitions.
- TuCSoN infrastructure supports the creation, access and manipulation of tuple centres, distributed among the nodes of the multiagent system, independently of the underlying network.

Workflow Management System

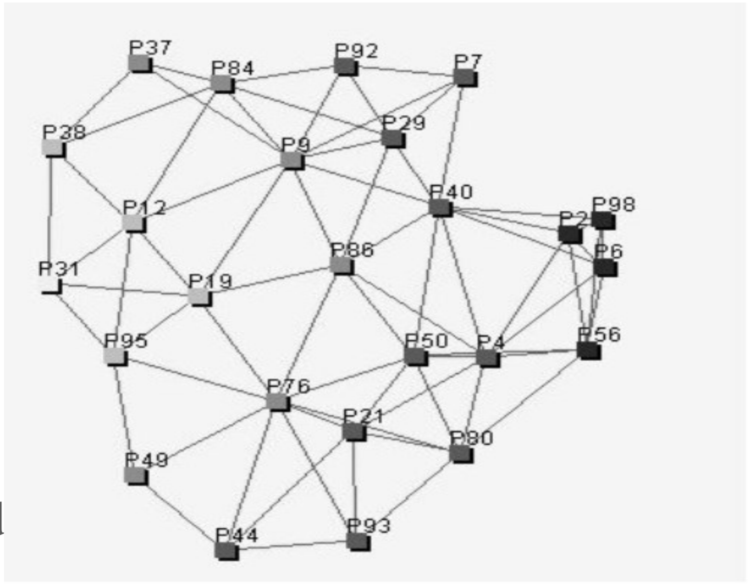
- Several workflow engine components embed and enact the workflow rules - defined by designers - for the coordination of the tasks assigned to participants.
  - Agents play the role of the participants
  - Workflow engines are designed and built as tuple centres **(environment abstraction)**.

- Designers of multiagent systems can mainly exploit tuple centres for encapsulating simple up to complex coordination rules, though they can be used also for knowledge sharing and indirect communication.



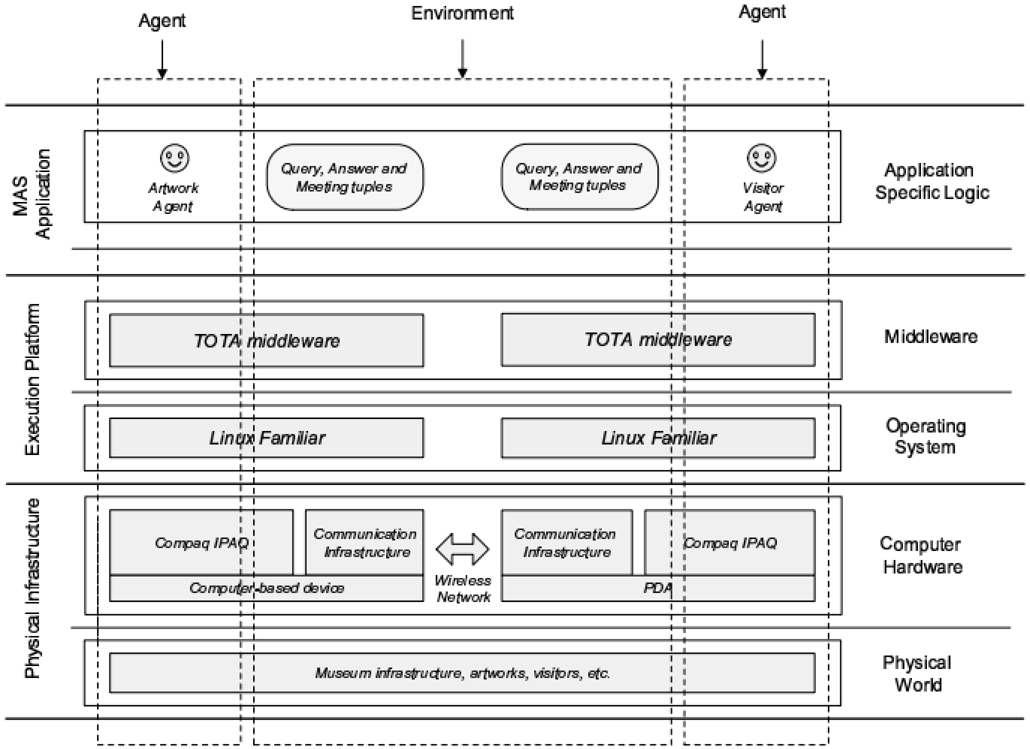
Co-fields in TOTA

- Each agent can **inject in the net some tuples** which encapsulate rules for their diffusion, aggregation and evaporation.
- The distribution of such tuples in time hence really models a field, which can be perceived by agents and navigated to retrieve resources and/or other agents.
- TOTA provides as **environment abstraction** the **TOTA tuple spaces** spread over the net topology, which implement the distributed space supporting co-fields.



Intelligent museum application

- Support the visitors of a museum in retrieving information about art pieces, to orientate inside the museum, and meet other people in the case of organised groups.
- This can be exploited by a designer in wireless and ad-hoc networks to implement services related to awareness and emergent coordination.



Lime

- A related infrastructure for mobile agents is LIME (Linda in Mobile Environment)
- Infrastructure for knowledge sharing and for indirect communication.
- Applications for mobile environment classified into two topologies

  (i) Data sharing(eg ROAMING JIGSAW)

  (ii) Transient interactions with other components as context changes(RED ROVER)

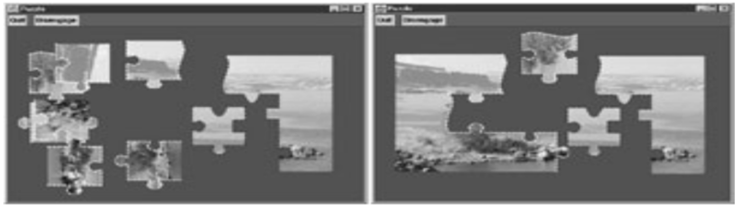


Figure 1: ROAMINGJIGSAW. When a player is disconnected (left), only the pieces previously selected can be assembled. Upon reconnection (right), the assemblies made meanwhile by other players become visible in the player's workspace.

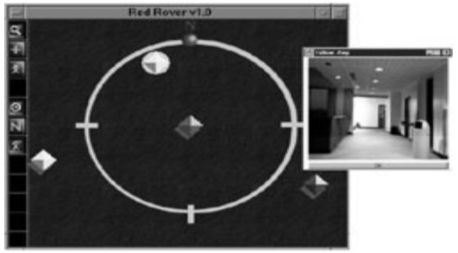


Figure 2: REDROVER. The main console of REDROVER, and the most recent camera image of a connected player.

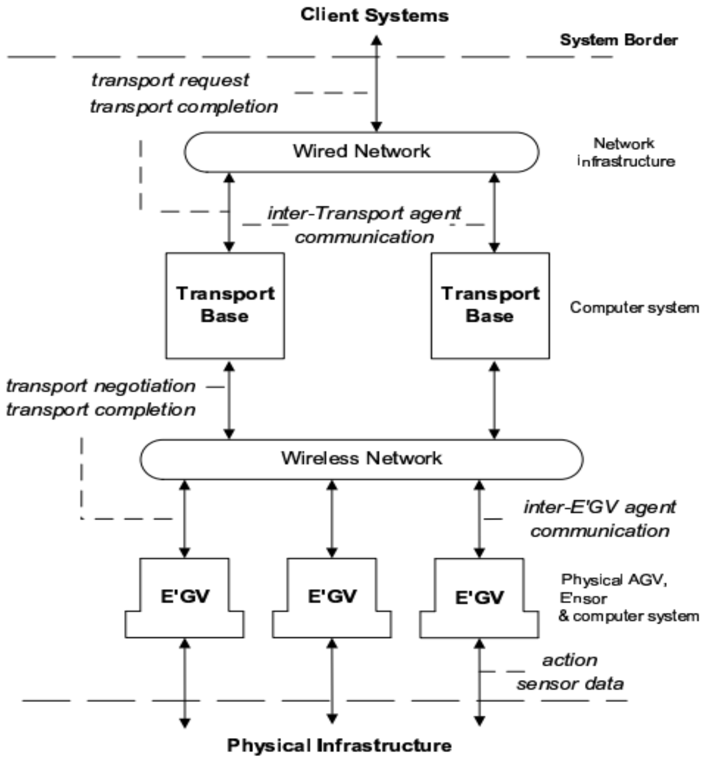
Mobile tuple spaces in Lime

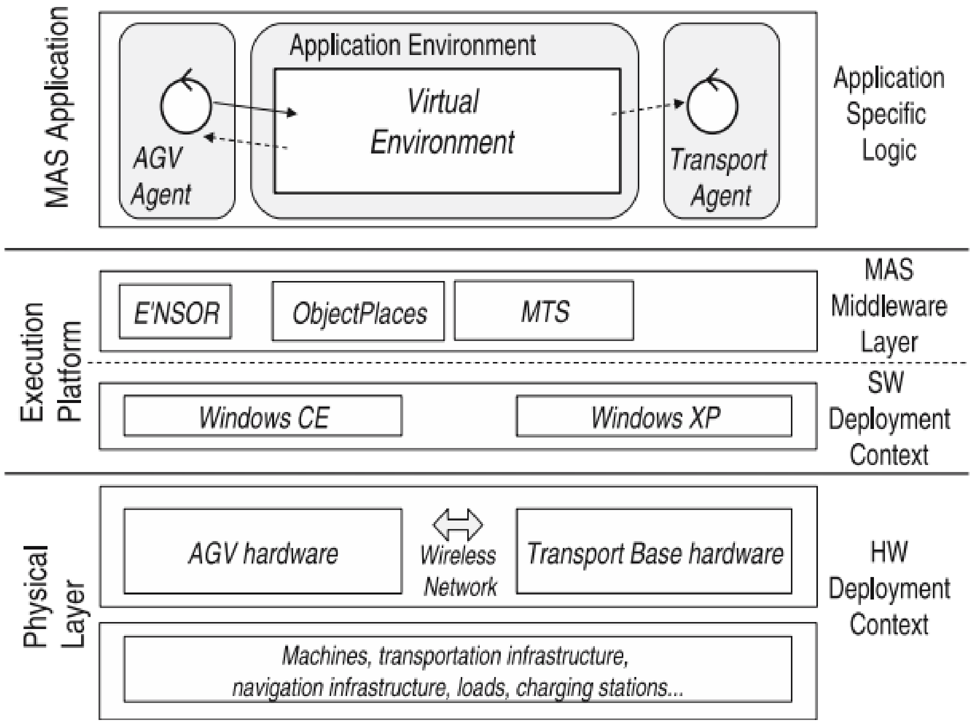
- Agents have **an individual interface tuple space (environment abstraction)** where they put and retrieve data, and which moves along with agents.
- When an agent resides in a host, its ITS actual content is not simply the result of its owner agent's interactions, but it is rather obtained by **dynamically joining it with the ITSs of the other agents in the same host.**
- Typical application scenarios of Lime feature mobile agents roaming an unknown environment, getting partial observations, and dynamically joining their knowledge.

Virtual environments in the AGV application

- **Automatic guided vehicles (AGV)**: unmanned vehicles controlled by agents transport various kinds of loads through a warehouse.
- **“Virtual environment” (VE)**: **(environment abstraction)** keep a consistent and updated map of the physical environment (including vehicles’ location and status, such as whether they are executing a job).
  - Each AGV keeps a local VE used as the information to share.
- **Transport bases**: where orders are generated and assigned to AGV agents by a number of Transport Agents.
- **Transport Agent**: responsible for assigning one transport to an AGV, and for making sure it is executed.
  - Each Transport Agent uses a local VE to get an up to date view on the AGVs position and status.

Virtual environments in the AGV application





Digital pheromone infrastructure

- “Command and Control” of multiple robotic entities, exploited e.g. in infrastructures for military air operations.
- The **digital pheremone infrastructure** provides services for injecting and perceiving digital pheromones in different sites of the physical/logical distributed environment, with the inner ability of aggregating, maintaining and diffusing information according to spatial and temporal criteria.
  - Inspired by stigmergy.

Digital pheromone infrastructure

- **Place agents** are stationary entities corresponding to regions of the problem space, forming a graph structure and supporting the environment infrastructure.
- **Walker agents** have the goal of finding roots in the system, and accordingly communicate with place agents asking for putting and perceiving pheromones of different kinds.
- **Avatar agents** are created to represent other entities in the environment (either enemies, friendly, or neutral), and can put and perceive pheromones as well.
- In this application, the set of walker agents and avatars exploit **environment abstractions** resembling repositories of digital pheromones, placed in the nodes of a network.

Digital pheromone infrastructure

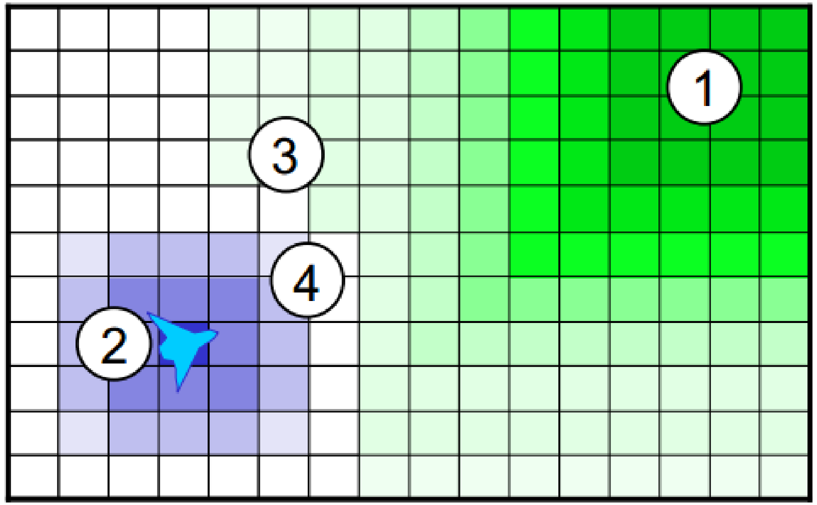


Figure 1. Attractive And Repulsive Pheromones For Surveillance, 1. Surveillance area deposits attractive pheromone, 2. ASV deposits repulsive pheromone, 3. Pheromone infrastructure propagates both attractive and repulsive pheromone to form gradient, 4. ASV climbs net gradient, withdrawing attractive pheromone.

Digital pheromone infrastructure

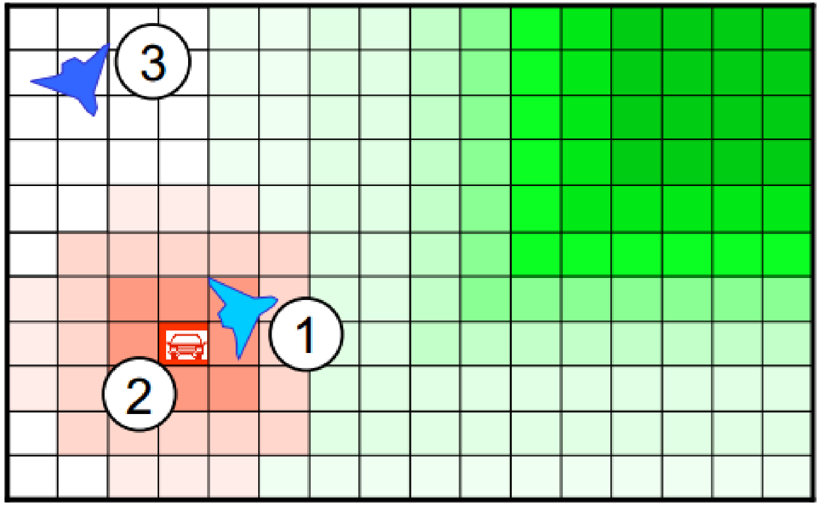


Figure 2. Pheromones Attracting Confirming Sensors – 1. ASVdet detects target and Red avatar is created, 2. Red avatar deposits "NeedsID" pheromone, 3. ASVid is more attracted to NeedsID pheromone than lawn pheromone and climbs gradient to ID target.

Digital pheromone infrastructure

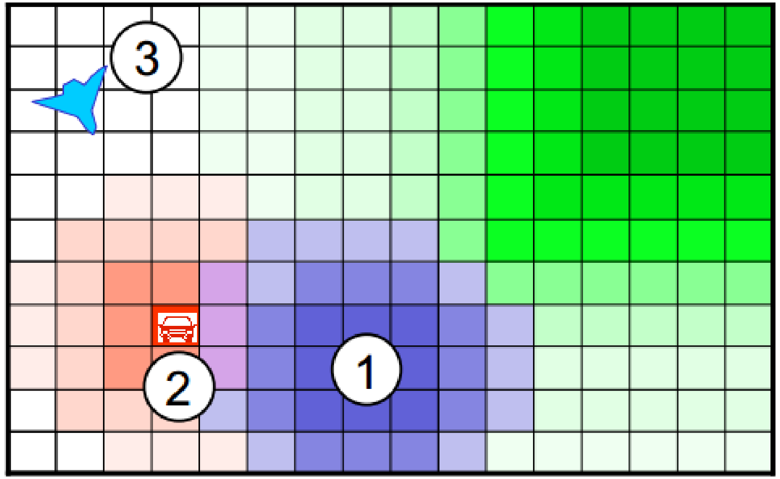


Figure 3. Pheromone Tracking Algorithm – 1. ASV acquires target and deposits Visited pheromone repelling other ASVs, 2. Red avatar estimates movement, deposits Tracking pheromone, 3. As avatar moves away from Visited deposit, nearby ASV is more attracted to its Tracking pheromone than repelled by Visited pheromone and climbs gradient to reacquire the target.

AMELI middleware for electronic institutions

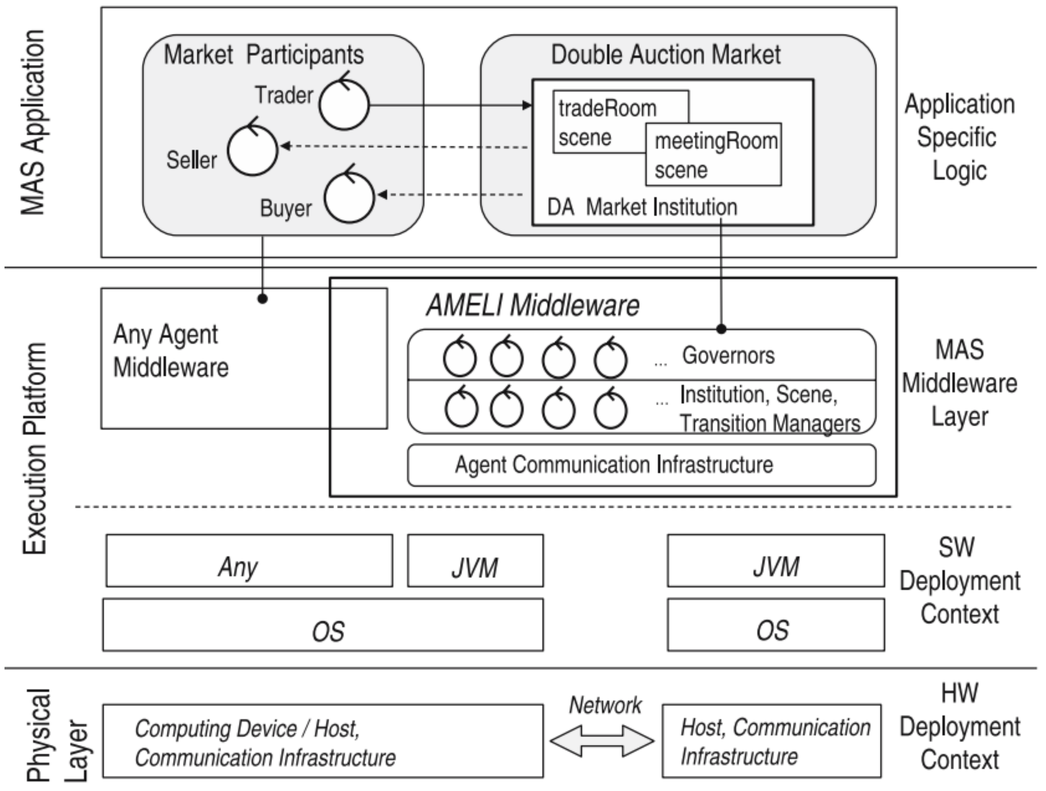
- One of the most challenging issues concerning the engineering of agent-based systems is balancing agent autonomy and social order.
- A normative system: a regulatory structure establishing permission and denials for agents.
- AMELI: a computational environment for possibly heterogeneous agents.
- Each institution represents a normative system, and includes *scenes* that agents can enter and exploit in order to exchange messages.

AMELI middleware for electronic institutions

- An application example is given by markets, where traders (buyers and sellers) meet to trade their goods under the supervision of the institution, and where auctions are used as trading protocols.
- The institution as computational environment **encapsulates** and **enforces the norms** that **characterise the trade**.

AMELI middleware for electronic institutions

- The **environment abstraction** provided by AMELI is hence **an institution of scenes**, which is a “context” where agents can meet and exchange messages in a regulated way, and where prohibited interactions are disallowed by the infrastructure.
- Designers exploiting AMELI are able to organise agents cooperation, regulating their access to the institution and enforcing the necessary rules and norms.



Conclusions

- Supporting environment abstractions through an infrastructure is a useful interpretation for many existing works on environments for multiagent systems.
- Identifying environment abstractions as building blocks of a multiagent system can foster the introduction of a methodology that tackles in a more systematic way the issue of environment engineering.
    - Identifying the proper environment infrastructure
    - Designing the environment abstractions to be included in the specific multiagent system
    - Taking into account environment abstractions in agent development

Conclusions

- The choice of infrastructure - and thus of environment abstractions - is an important early design decision that has a significant influence on the architecture of the application, and it is difficult to reverse later.
- Exploration on identifying a general model for environment abstractions could help streamlining research on theory and practice of environment in multiagent systems.

Our conclusions

- The environment infrastructure is a new focus in multiagent system design.
- The environment infrastructure design is as important as agent design.
- Good choice of environment infrastructure can provides opportunities to system designers when developing multiagent applications while assuring agent behaviours like autonomy, proactivity, social attitude, and goal-driven.
- It is efficient to obtain environment infrastructures as it can be reused for multiple related applications (reducing the workload).
- Environment abstraction makes software implementation easier and efficient.
- Environment can serve as robust, self revising shared memory and an excellent medium for indirect coordination of agents

Question?