# Agents & MAS for Self-Organising Systems

## Autonomous Systems

Sistemi Autonomi

Andrea Omicini
*after Luca Gardelli*
andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

Academic Year 2013/2014

1. Multi-Agent Systems vs. Self-Organising Systems

2. Methodologies for Engineering SOS

3. A Principled Approach for Engineering SOS
   - The Case Study of Plain Diffusion

# MAS 4 SOS

- Is the agent paradigm the right choice for modelling and developing SOS?
- Are agents the right abstractions for SOS components?
- Are MAS the right way to put together components of a SOS?
- In order to answer this question we have to compare requirements for SOS with features of MAS

# SOS Requirements

- From our previous discussion on self-organisation and emergence, a possible basic requirements list can be given as follows:
  - *Autonomy* and *encapsulation* of behaviour
  - *Local actions* and *perceptions*
  - *Distributed environment* supporting *interactions*
  - Support for *organisation* and *cooperation* concepts

# MAS Checklist

- It is easy to recognise that the agent paradigm provides suitable abstractions for each aspect
  - Agents for autonomy and encapsulation of behaviour
  - Situated agents for local actions and perceptions
  - MAS distribution of components, and MAS environment supporting interactions through coordination
  - MAS support for organisation and cooperation concepts
- As a matter of fact, MAS are currently the reference for both self-organisation modelling and engineering
- In self-organisation literature not having a background in computer science, it is often the case that the term agent is used with a different meaning
- For instance, in biology and chemistry complex chemical compounds are often called *agents* without actually referring to the agent paradigm

# Current MAS Methodologies

- Most MAS methodologies were developed because of the need to address specific issues
- For instance Gaia was initially concerned more with intra-agent aspect, while SODA dealt with aspects at the society level
- Engineering methodologies are related to the paradigm in use
- Being interested in SOSs, we need a methodology that supports the basic requirements previously identified

# MAS Methodologies for SOS

- Unfortunately there are only a few methodologies soundly supporting organisation and environmental aspects [Molesini et al., 2007]
- The ADELFE methodology is a proposal for Adaptive MAS where properties emerges by self-organisation [Bernon et al., 2004]
- Although considering cooperation and environmental issues of self-organisation, in our opinion ADELFE provide no pragmatic approach for the engineering of emergence

# Designing Self-Organising Emergent Systems

- In developing artificial self-organising systems displaying emergent properties we identify two main issues [Gardelli et al., 2007a, Gardelli et al., 2007b]
  1. How do we design individual agent behaviour that collectively produce the target emergent property? : Due to non-linearities both in agent behaviour and environmental dynamics devising a strategy that eventually leads to the target property is a very difficult problem.
  2. How do we evaluate a specific solution and provide actual guarantees of its quality? : Because of dependability requirement, we cannot deploy a system without having profiled the possible evolutions and framed the working environmental conditions.

# Intro

- In the rest of the seminar we describe a possible approach for the engineering self-organising MAS with emergent properties
- In particular we consider issues related both to workflow and tools
- The material presented from now on is mostly based on [Gardelli et al., 2007a, Gardelli et al., 2007b]
- We now start considering the two previous issues, one at a time

# Issue 1: Forward vs. Reverse Engineering

- How do we design individual agent behaviour that collectively produce the target emergent property?
- It is generally acknowledged that forward engineering of emergent properties is feasible only for small/trivial problems
- Indeed, most of the artificial self-organising systems have been inspired by natural systems

# Issue 1: Inspiration

- Although pervasive, "inspiration" process is not a scientific approach and it is hardly reproducible
- We need a way to map computer science problems into successful natural strategies
- Only recently, it has been recognised the need of a more formal approach when designing SO MAS: a few proposal involve design patterns [Babaoglu et al., 2006, De Wolf and Holvoet, 2007, Gardelli et al., 2007c]

# Issue 1: Design Patterns

- Initially introduced in architectural engineering, design patterns have been popularised in computer science in the 1990s along with the object-oriented paradigm [Gamma et al., 1995]
- A design pattern provide a reusable solution to a recurrent problem in a specific domain
- In our context design patterns are a viable approach to encode successful solution provided by natural systems to computer science problems [Babaoglu et al., 2006, De Wolf and Holvoet, 2007, Gardelli et al., 2007c]
- Although there have been already proposed several patterns, we are confident that we will not find a suitable pattern for every computer science problem: we will discuss it later when dealing with Issue 2

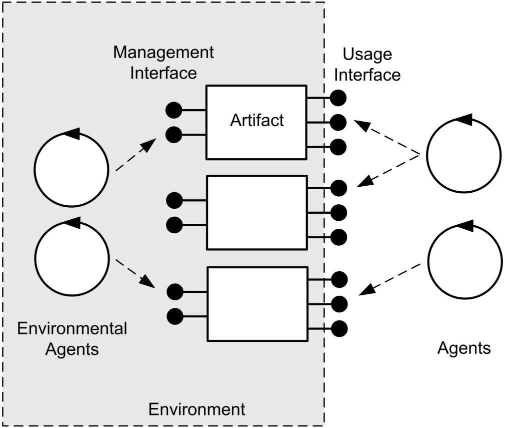
# Issue 1: Feedback Loop

- Self-organisation and Emergence involve the existence of a feedback loop
- Such feedback loop is often produced by a functional coupling between agents and the environment
- E.g. consider the ants depositing pheromone while the environment evaporates it

# Issue 1: Architectural Pattern I

- When designing a SO MAS according to the Agents & Artifacts metamodel [Ricci et al., 2006] we identify a recurrent architectural solution
- Since, it is often the case that the agent environment is partially or completely given, such as in case of legacy resources, we do not have complete control over the environment
- Hence, being difficult to embed self-organisation into artifacts, we introduce environmental agents whose role is to close the feedback loop between agents properly managing artifacts behaviour
- Furthermore, environmental agents allow a finer control isolating normal behaviour from the one responsible of emergent properties

# Issue 1: Architectural Pattern II




The architectural pattern featuring environmental agents encapsulating self-organising behaviour and managing artifacts.

# Issue 1: Summarising

- Forward engineering of emergent properties is not feasible, hence we rely on the existence of a natural system providing a suitable solution
- Such solution should be encoded as a design pattern eventually leading to the creation of a coherent pattern catalogue
- In particular the design pattern should provide behaviours for the three roles identifies in the architectural pattern: agents, artifacts and environmental agents

# Issue 2: Towards a Workflow

- How do we evaluate a specific solution and provide actual guarantees of its quality?
- In order to fulfill this issue we promote the following iterative engineering process
  1. Modelling
  2. Simulation
  3. Verification
  4. Tuning (if needed then back to step 2)

# Issue 2: Exploiting Formal Tools

- Since we are going to perform several tasks on a given model we promote the use of formal tools
- Formal languages allow the specification of selective and unambiguous models and provide a solid basis for automatic processing
- Hence, having a model expressed in a suitable formal language we can

    1. run simulations by specifying only operating parameters
    2. verify the system by model-checking just providing the properties in a suitable temporal logic

# Workflow: Modelling

- During the modelling phase we have, according to the architectural pattern, identify the roles of each entity, namely agents, artifacts and environmental agents
- The individual behaviour is to be found within the design pattern catalogue
- Modifications to the pattern may be required to fit the actual requirements: this is a non-trivial step and requires expertise in the domain
- In this phase the model should not be too detailed, rather reflect the abstract architecture of the system: indeed a fine-grained model can prevent further automatic processing

# Workflow: Simulation

- Simulation allows us to qualitatively preview the global system dynamics
- Before running the simulation we have to provide working parameters for agents and artifacts, while parameters set for environmental agents is our unknown variable
- Needless to say that in order for the simulation results to be valid parameters should reflect the actual deployment conditions
- Although the use of simulation is a common practice in system engineering, it is almost unused in software development
- In self-organisation literature, the need for simulation has been recognised only recently [Gardelli et al., 2006] [Gardelli et al., 2007a] [Bernon et al., 2006] [De Wolf et al., 2006]

# Workflow: Verification

- Simulation alone does not provide sound guarantees because of incompleteness
- Conversely, model checking [Edmund M. Clarke et al., 1999] is a formal technique for verifying automatically the properties of a target system against its model
- The model to be verified is expressed in a formal language, typically in a transition system fashion
- Then, properties to be verified are formalised using a variant of temporal logic depending on the current model
- The main drawback of model checking is dependence upon model state space which grows very quickly, becoming unfeasible

# Workflow: Tuning

- If the current system model does not meet requirements we have to tune its parameters
- This implies a further cycle, of simulation-verification-tuning
- If the results display discrepancies with requirements we may consider also altering the model

# Workflow: Tools

- In order to ease the workflow we need a tool supporting the whole process
- The tool must meet the following requirements
  - provide a formal modelling language allowing to express stochastic aspects
  - provide a built-in stochastic simulator able to run directly from the specified model
  - provide a built-in probabilistic model checker and support the specifications of temporal logic properties

# Tools: PRISM

- Among the various available tools we selected PRISM – Probabilistic Symbolic Model Checker developed at University of Birmingham [PRISM, 2007]
- PRISM language allows the specification of models in a transition-system fashion
- The built-in stochastic simulator is very simple but has plotting and exporting capabilities, although more sophisticated tools would have been appreciated
- The built-in probabilistic model checker is very robust: it provides alternative engines and allows the specification of properties both in PCTL – Probabilistic Computational Tree Logic and CSL – Continuous Stochastic Logic

# Outline

1 Multi-Agent Systems vs. Self-Organising Systems

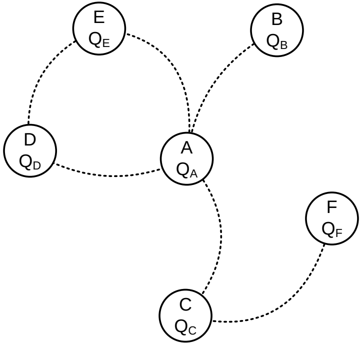
2 Methodologies for Engineering SOS

3 A Principled Approach for Engineering SOS
- The Case Study of Plain Diffusion

# Problem Statement

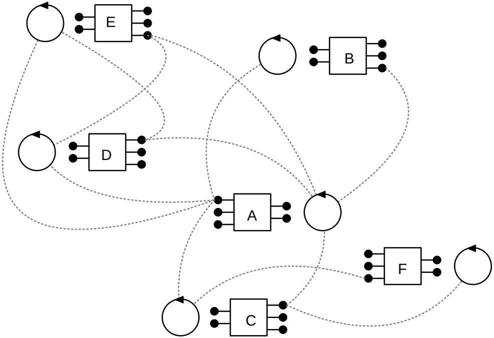
- Provided a networked set of nodes not fully connected where each node hosts a certain amount of data items
- Given that each node knows only (i) the number of local items, and (ii) the neighbouring nodes, while has no information about network size and total amount of items
- Devise a self-organising strategy for implementing a plain diffusion strategy that eventually leads the system to a state where each node has the same amount of items

# Reference Network Topology




The reference topology: starting from state $A = 36, B = C = D = E = F = 0$ the system must evolve into $A = B = C = D = E = F = 6$.

# Equivalent A&A Topology




Notice that this topology is equivalent to the previous one.

# Modelling

- We have to provide a strategy for environmental agents that exchanging items with neighbouring artifacts based on local information eventually produce the desired dynamics
- The key is the dynamical equilibrium established by agents exchanging items at different rates: if the exchange rates are identical the situation remains statistically unchanged
- Agents have to exchange items proportionally to the local number of items, i.e. working faster when having large number of items and slower in the other case
- Furthermore, agents should exchange items proportionally to the number of neighbouring nodes: hubs have to work faster to avoid congestion!

# PRISM Model

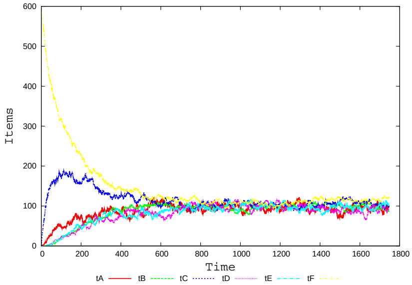
- We describe the model using the PRISM language in order to allow further automatic elaborations
- PRISM language define a transition system

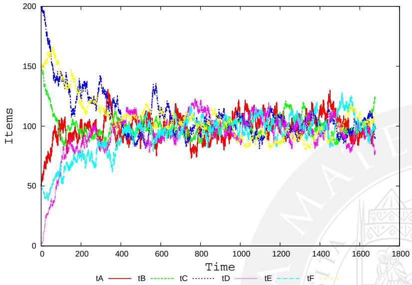
```
module agentA
[] tA > 0 & tB < MAX & tC < MAX & tD < MAX ->
rA : (tA'=tA-1) & (tB'=tB+1) +
rA : (tA'=tA-1) & (tC'=tC+1) +
rA : (tA'=tA-1) & (tD'=tD+1) +
rA : (tA'=tA-1) & (tE'=tE+1);
endmodule
```

The code snippet show the description of the agent hosted by the hub, node A.

# Simulation

- Providing values for system parameters we can run simulations directly from PRISM

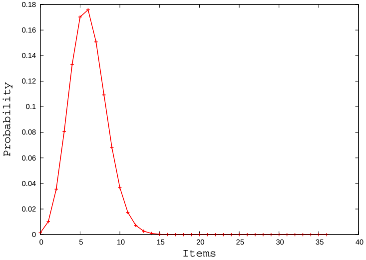



Two sample simulations from different initial states (left) all items in one node (right) almost sorted.

# PRISM Model Checking

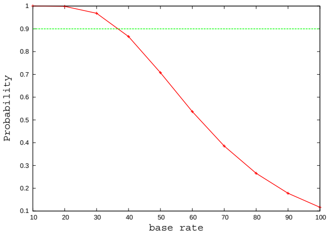
- Which is the steady-state probability for the node X to contain Y items?: using the PRISM syntax for CSL properties $S =? [tA = Y]$




The chart displays the distribution of the probability for a node to contain a specific number of items: further experiments show that the chart is the same for each node.

# Tuning

- Is the probability of reaching the dynamic equilibrium condition within 200 time units greater or equals to 90% ?: using the PRISM syntax for PCTL properties $P >= 0.9\ [true\ U <= 200\ tB = 6]$ for the node tB




The chart displays the probability values for the node tB varying base rate parameter: we can guess that the desired value is within the range 30..40.

1. Multi-Agent Systems vs. Self-Organising Systems
2. Methodologies for Engineering SOS
3. A Principled Approach for Engineering SOS
    - The Case Study of Plain Diffusion

# Bibliography I


Babaoglu, O., Canright, G., Deutsch, A., Di Caro, G. A., Ducatelle, F., Gambardella, L. M., Ganguly, N., Jelasity, M., Montemanni, R., Montresor, A., and Urnes, T. (2006).
Design patterns from biology for distributed computing.
*Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):26–66.

Bernon, C., Camps, V., Gleizes, M.-P., and Picard, G. (2004).
Designing agents' behaviors and interactions within the framework of ADELFE methodology.
In *Engineering Societies in the Agents World*, volume 3071 of *LNCS (LNAI)*, pages 311–327. Springer.
4th International Workshops, ESAW 2003, London, UK, October 29-31, 2003, Revised Selected and Invited Papers.

Bernon, C., Gleizes, M.-P., and Picard, G. (2006).
Enhancing self-organising emergent systems design with simulation.
In *Seventh International Workshop on Engineering Societies in the Agents World (ESAW'06)*, Dublin, Ireland. University College Dublin (UCD).

# Bibliography II

De Wolf, T. and Holvoet, T. (2007).
Design patterns for decentralised coordination in self-organising emergent systems.
In Brueckner, S., Hassas, S., Jelasity, M., and Yamins, D., editors, *Engineering Self-Organising Systems*, volume 4335 of *LNCS*, pages 28–49. Springer.
Fourth International Workshop, ESOA 2006, Future University-Hakodate, Japan, 2006, Revised Selected Papers.

De Wolf, T., Holvoet, T., and Samaey, G. (2006).
Development of self-organising emergent applications with simulation-based numerical analysis.
In Brueckne, S. A., Di Marzo Serugendo, G., Hales, D., and Zambonelli, F., editors, *Engineering Self-Organising Systems*, volume 3910 of *LNCS*, pages 138–152. Springer.
Third International Workshop, ESOA 2005, Utrecht, The Netherlands, July 2005, Revised Selected Papers.

Edmund M. Clarke, J., Grumberg, O., and Peled, D. A. (1999).
*Model Checking*.
The MIT Press.

# Bibliography III


Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995).
*Design patterns: elements of reusable object-oriented software*.
Professional Computing. Addison-Wesley, One Lake Street, Upper Saddle River, NJ, 07458, USA.

Gardelli, L., Viroli, M., Casadei, M., and Omicini, A. (2007a).
Designing self-organising environments with agents and artifacts: A simulation-driven approach.
*International Journal of Agent-Oriented Software Engineering*, 2(2).
In Press.

Gardelli, L., Viroli, M., Casadei, M., and Omicini, A. (2007b).
Designing self-organising MAS environments: The collective sort case.
In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for Multi-Agent Systems III*, volume 4389 of *LNAI*, pages 254–271. Springer.
3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers.

# Bibliography IV


Gardelli, L., Viroli, M., and Omicini, A. (2006).
On the role of simulations in engineering self-organising mas: The case of an intrusion detection system in tucson.
In Brueckner, S. A., Di Marzo Serugendo, G., Hales, D., and Zambonelli, F., editors, *Engineering Self-Organising Systems*, volume 3910 of *LNAI*, pages 153 – 166. Springer Berlin / Heidelberg.
Third International Workshop, ESOA 2005, Utrecht, The Netherlands, July 25, 2005, Revised Selected Papers.

Gardelli, L., Viroli, M., and Omicini, A. (2007c).
Design patterns for self-organising systems.
In Burkhard, H.-D., Lindemann, G., Verbrugge, R., and Varga, L. Z., editors, *Multi-Agent Systems and Applications V*, volume 4696 of *LNCS (LNAI)*, pages 123–132. Springer, Heidelberg.
5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007, Leipzig, Germany, September 25–27, 2007.

Molesini, A., Omicini, A., and Viroli, M. (2007).
Environment in agent-oriented software engineering methodologies.
*International Journal on Multiagent and Grid Systems*.
In Press. Special Issue on Engineering Environments for Multiagent Systems.

# Bibliography V

PRISM (2007).
PRISM: Probabilistic symbolic model checker.
Developed at University of Birmingham. Version 3.1.1 available online at http://www.prismmodelchecker.org/.

Ricci, A., Viroli, M., and Omicini, A. (2006).
Programming MAS with artifacts.
In Bordini, R. P., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors, *Programming Multi-Agent Systems*, volume 3862 of *LNAI*, pages 206–221. Springer.
3rd International Workshop (PROMAS 2005), AAMAS 2005, Utrecht, The Netherlands, July 26, 2005. Revised and Invited Papers.

# Agents & MAS for Self-Organising Systems

## Autonomous Systems

Sistemi Autonomi

Andrea Omicini
*after Luca Gardelli*
andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

Academic Year 2013/2014