



## The tool

AESOP (Agents and Emergencies for Simulating Organizations in Python) uses actions, agents (acting and deciding people) and the scheduling of events into an agent based framework.

Agents may use fixed rules but they can also learn to improve these rules in a changing environment. They can also be modeled so as to be aware of the consequences of their behavior.

In the simulator, single agents can also be modeled as neural networks, so that the system appears as a "net of neural networks".



The simulator is currently based on Swarm ([www.swarm.org](http://www.swarm.org)) as basic layer, see <http://web.econ.unito.it/terna/jes>

The AESOP simulator is under implementation in Python ([www.python.org](http://www.python.org)), using **SLAPP, Swarm-Like Agent Protocol in Python**, <http://eco83.econ.unito.it/terna/slapp/>

[we want to join the easiness of Python, and its openness, to the clarity of the Swarm protocol; Python is connected in AESOP to the R statistical system (R is at <http://cran.r-project.org/>), via the rpy library, at <http://rpy.sourceforge.net/>)]

# The SWARM protocol and SLAPP



An **absolutely clear and rigorous** application of the SWARM protocol is contained in the original SimpleBug tutorial (1996?) with ObjectiveC code and text by the Swarm development team in Santa Fe (N. Minar, R. Burkhart, C. Langton and M. Askenazi), on line at

<http://ftp.swarm.org/pub/swarm/apps/objc/sdg/swarmapps-objc-2.2-3.tar.gz>

(into the folder “tutorial”, with the texts reported into the README files in the tutorial folder and in the internal subfolders)

The same tutorial has also been adapted to Java by Charles J. Staelin (*jSIMPLEBUG, a Swarm tutorial for Java*, 2000), at

<http://www.cse.nd.edu/courses/cse498j/www/Resources/jsimplebug11.pdf> (text) or

<http://eco83.econ.unito.it/swarm/materiale/jtutorial/JavaTutorial.zip> (text and code)

At <http://eco83.econ.unito.it/terna/slapp> you can find the same structure of files, now implementing the SWARM protocol using Python

So, the **SWARM protocol** as *lingua franca* in agent based simulation models

Existing code (a Java layer upon a JavaSwarm implementation, on  
line at <http://web.econ.unito.it/terna/jes/>)



---

## WD - DW - WDW simplified formalism

---

Existing code (a Java layer upon a JavaSwarm implementation, on  
line at <http://web.econ.unito.it/terna/jes/>)



- **WD side or formalism: What to Do**
- **DW side or formalism: which is Doing What**
- **WDW formalism: When Doing What**



**recipe** = a sequence of steps to be executed

WD, What to Do

**unit** = a productive structure within an organization or operating alone; a unit is able to perform one (or more) of the steps described in a recipe

DW, which is Doing What

**a clock**

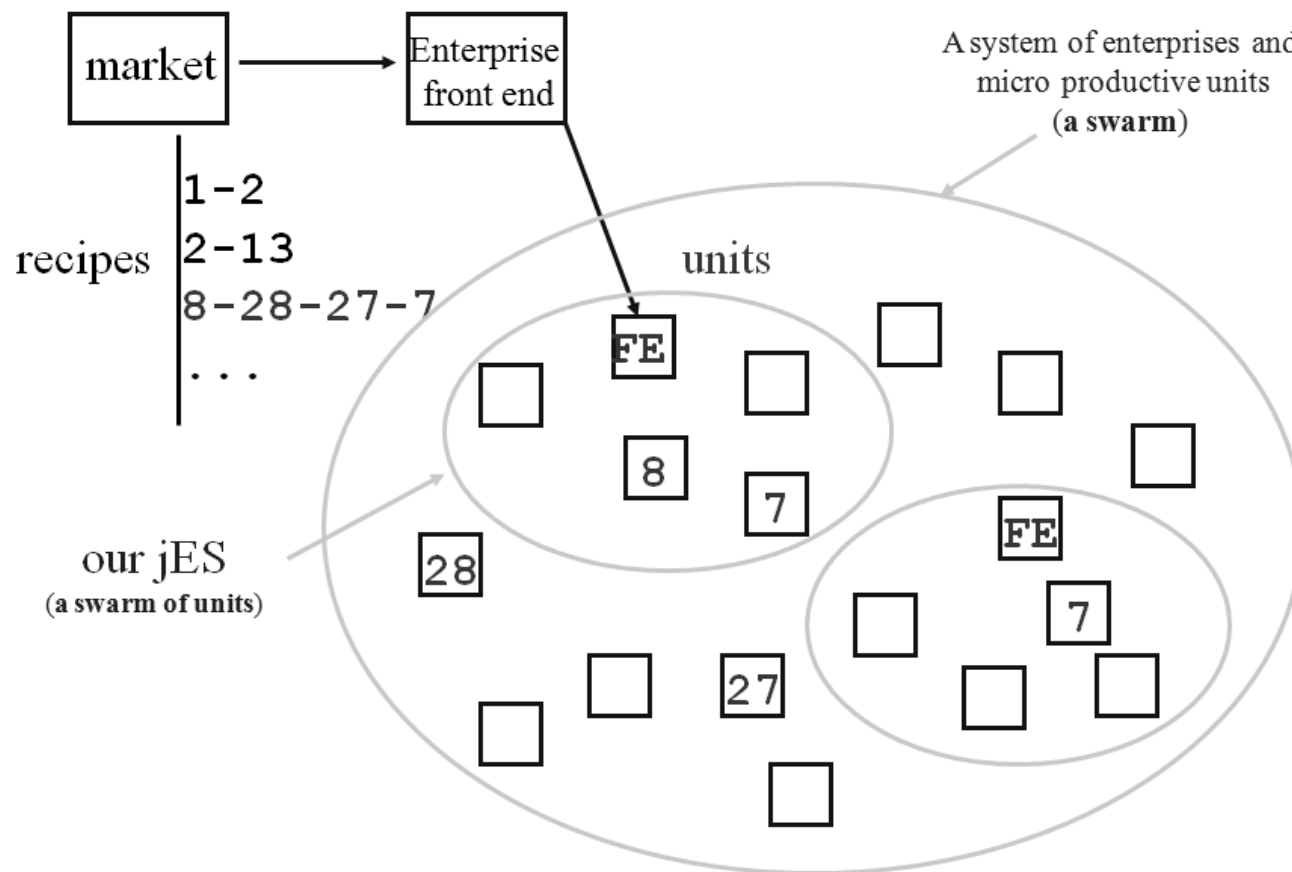
WDW, When Doing What

Existing code (a Java layer upon a JavaSwarm implementation, on line at <http://web.econ.unito.it/terna/jes/>)



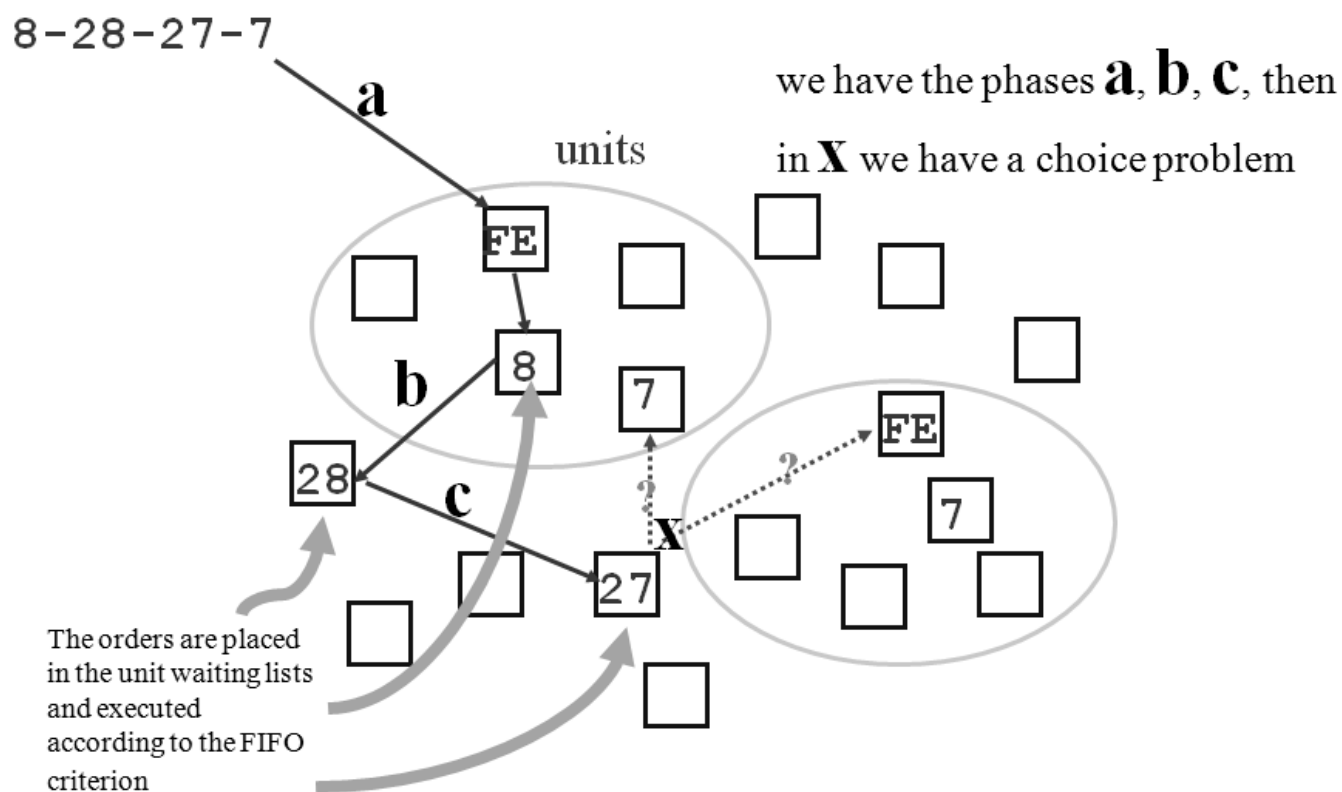
Existing code (a Java layer upon a JavaSwarm implementation, on line at <http://web.econ.unito.it/terna/jes/>)

### Recipes and production units





Existing code (a Java layer upon a JavaSwarm implementation, on line at <http://web.econ.unito.it/terna/jes/>)  
Recipes on move





Existing code (a Java layer upon a JavaSwarm implementation, on line at <http://web.econ.unito.it/terna/jes/>)  
Recipes on move

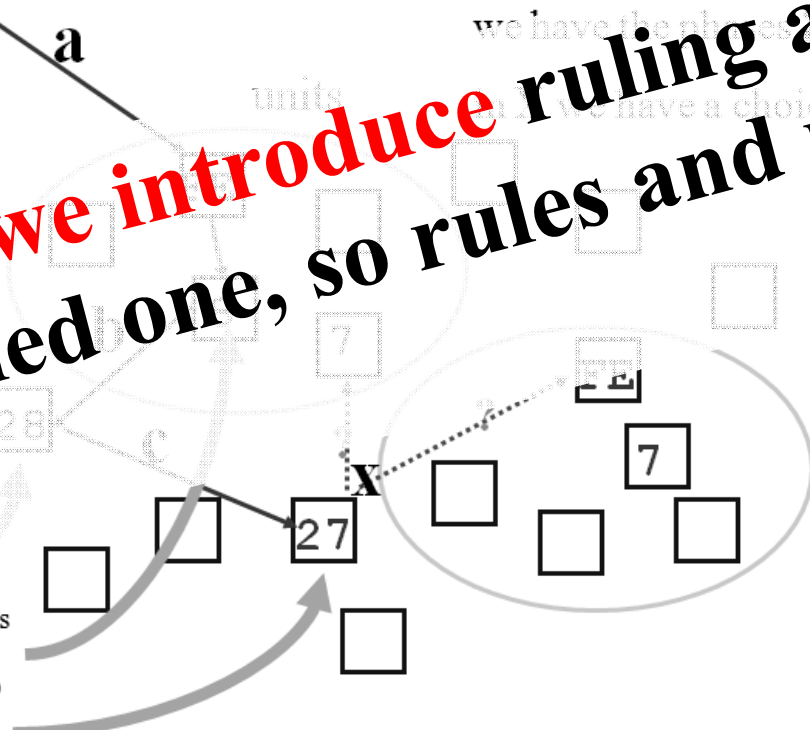
**what if we introduce ruling agents and ruled one, so rules and meta-rules?**

8-28-27-7

a

rules?

The orders are placed in the unit waiting lists and executed according to the FIFO criterion





The existing jES (Java Enterprise Simulator, with applications) tool is introduced in **E. Mollona (ed.), Computational analysis of firms' organization and strategic behavior, Routledge, forthcoming (2010)**

\* \* \*

The main idea for the evolution of the simulator framework is that of introducing:

- **rule modifications** (**adaptive** WD side of the model) and
- **agent adaption to changing rule** (**adaptive** DW side of the model)