# Agent-based Modeling for Biological Systems: Spacial and Mobility aspects

Emanuela Merelli

joint work with
Ezio Bartocci, Nicola Cannata, Luca Tesei and Flavio Corradini

University of Camerino

Stony Brook, 8th November 2007

# Outline

- Computational Systems Biology: the challenge 21st century
- Agent and multi-agent paradigm: an overview
- Agent-based modeling in systems biology: our experience
  - Orion: space, time e mobility and distributed coordination
  - Kalliope: space, time e mobility and centralized coordination

The challenge of the 21st century will be to understand how biological molecules in living systems **integrate** to complex systems, how these systems **function** and their **evolution**.

We are scaling up from computational molecular biology (i.e. genomes, transcriptomes, proteomes, metabolomes) to **computational systems biology**

[*Computational challenges of Systems Biology* A. Finkelstein et al. IEEE Computer 2004]

**Computational systems biology** is an emerging field in biological simulation that attempts to model or simulate intracellular and intercellular events using data gathered from genomic, transcriptomic, proteomic or metabolomic experiments.

## Why so important?

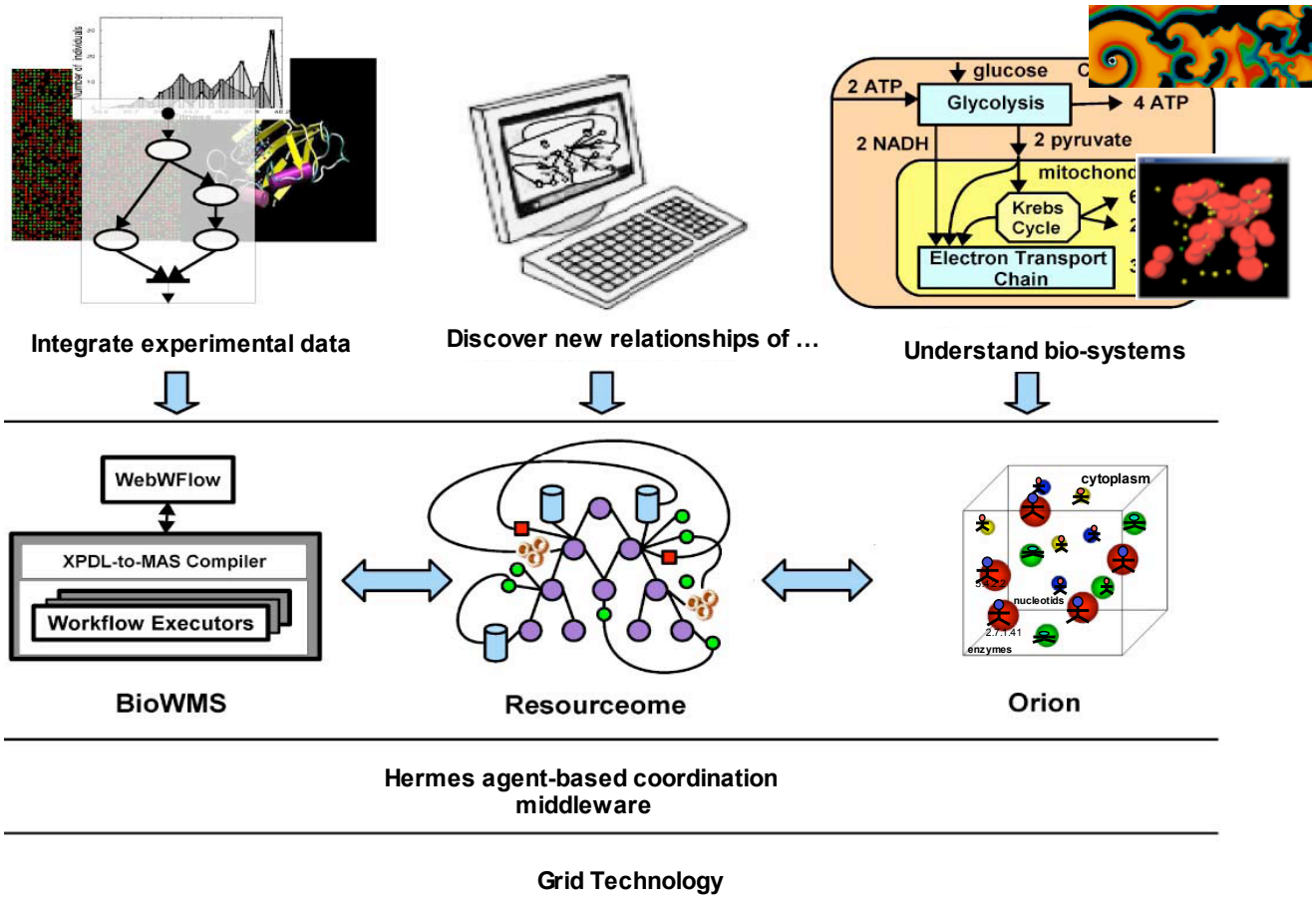
Models, from cells to organisms, will be used in **drugs research** for understanding their effects and side-effects

- avoiding to "torture" animals
- for building personalized medicines
- ...

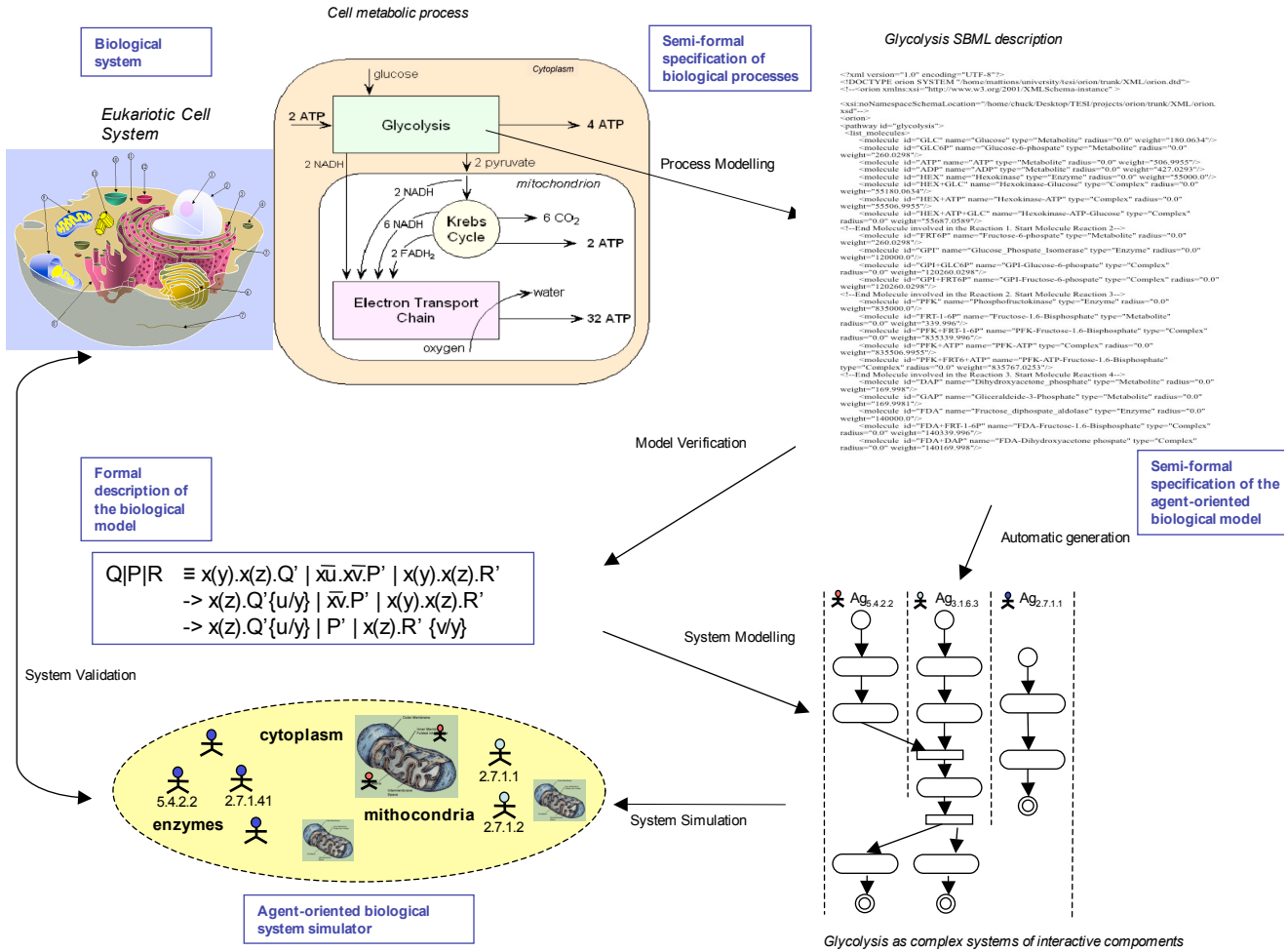
Computational Systems Biology typically requires tools to

- **integrate** the huge amount of experimental data
- **understand** the behaviour of biological systems
  - by creating the model
  - by simulating the model
- **discover** new relationships between the various parts of a biological system, such as organelles, cells, organs, organisms
  - by perturbing the model of a biological system (biologically, genetically, or chemically)
- ...
- **access** to a huge amount of computational-power

# UNICAM-LITBIO project architecture

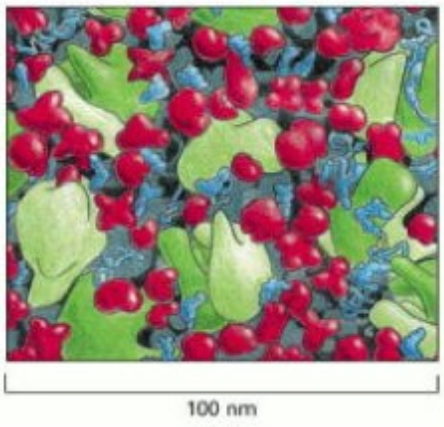


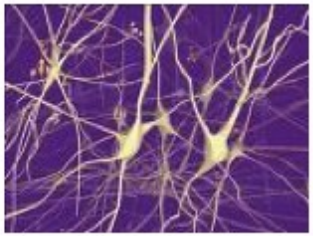
# UNICAM-LITBIO project methodology

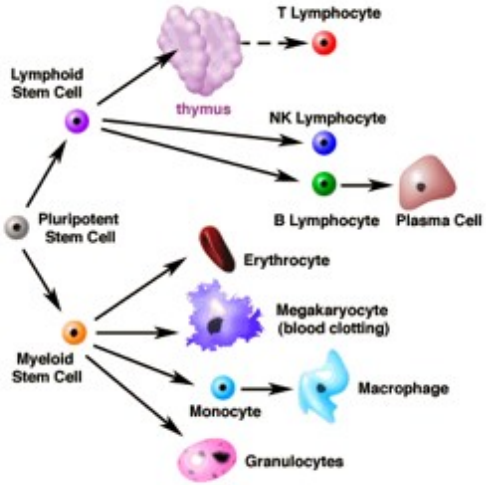


Bological artifacts exhibit the characteristics of **concurrent reactive systems** (Harel & Pnueli,86) on many levels: from the molecular, via the cellular, and all the way up to organs and full organisms  even entire populations.

**Orchestrated** and **coordinated** concurrency of systems from Nature can be remarkably beautiful and inspiring






The **collective behaviour** of the system emerges from **local interactions** among components

*Rencently from Luca Cardelli . . . about collective "stochastic" behaviour said:*

- A large set of interacting finite state automata
  - not quite **language automata** (large set)
  - not quite **cellular automata** (interacting, but not in a grid)
  - not quite **process algebra** (collective behaviour)
  - cf. **multiagent systems** and **swarm intelligence**

. . . **hybrid automata?**

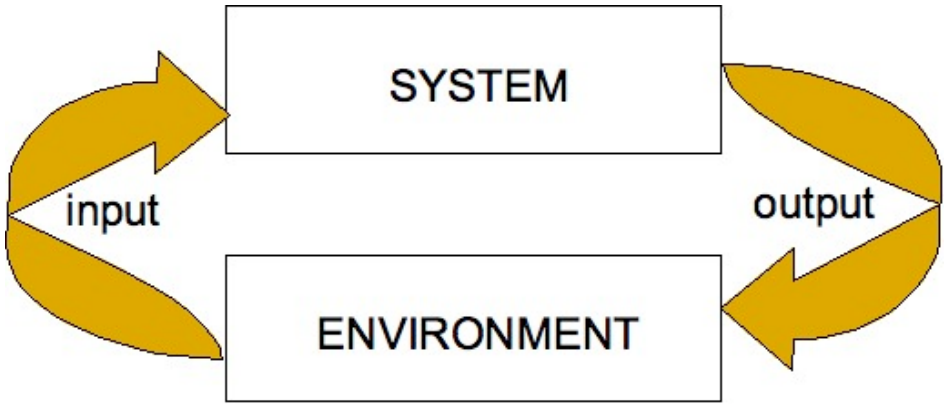
**A multiagent system** consists of a number of agents interacting one-another

- agents will be acting with their different **goals and motivations**;
- agents will require **the ability to cooperate, coordinate, and negotiate** with each other.

[*An Introduction to MultiAgent Systems* by Michael Wooldridge, John Wiley & Sons, 2002]

An **agent** is a computer system **situated** in some **environment** and capable of **autonomous**, **flexible** action in that environment in order to meet its design objectives

[M. Wooldridge, *Agent-based software engineering.* In IEE Proceedings of Software Engineering, 1997]



**An autonomous agent** is a computer system capable of **autonomous** (problem-solving), flexible action, situated in dynamic, likely open, unpredictable and typically multi-agent domains.

**The agent has the control over its internal state and over its own behaviour**

**A situated agent** is a computer system capable of flexible, autonomous (problem-solving) action, **situated** in dynamic, likely open, unpredictable and typically multiagent domains.

**The agent perceives the environment through sensors and acts through effectors**

**A flexible agent** is computer system capable of **flexible**, autonomous (problem-solving) action, situated in dynamic, eventually open, unpredictable and typically multi-agent domains.

*reactive:* **the agent responds in timely fashion to environmental change**

*adaptive:* **the agent responds to the environmental change according to its internal state**

*proactive:* **the agent acts in anticipation of future goals**

**Reactive agents** can be defined by a 6-tuple

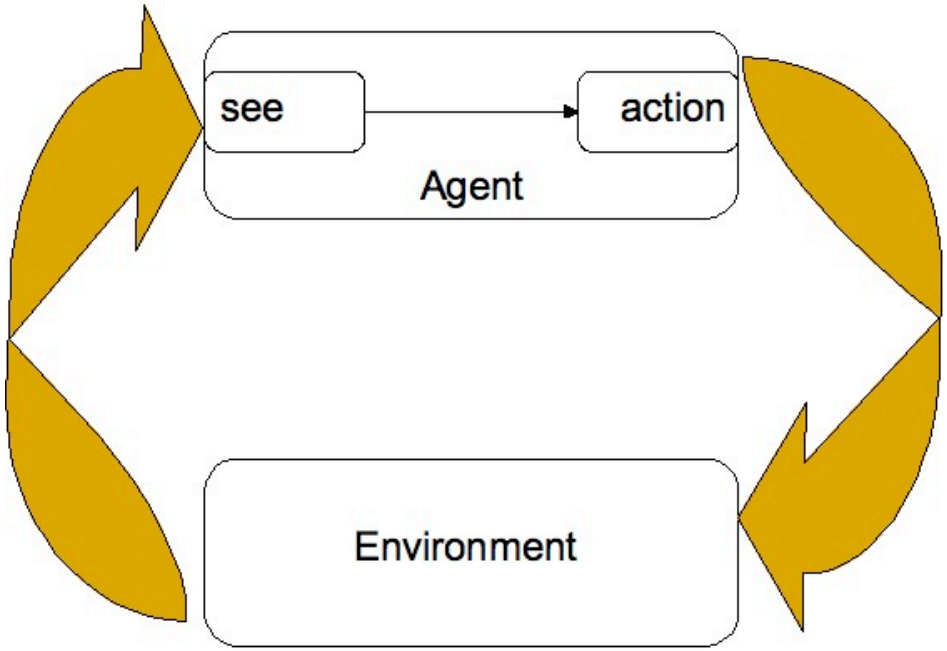
$$< E, P, A, see, do, action >$$

where

- $E$ is the set of states for the environment
- $P$ is a partition of $E$ (representing the perception of the environment from the agent's point of view)
- $A$ is a set of actions
- $see : E \rightarrow P$
- $action : P \rightarrow A$
- $do : A \times E \rightarrow E$

These agents observe the environment (*see*), find the appropriate action (*action*), and act (*do*) on the environment.

## Reactive agent



**Adaptive agents** can be defined by a 9-tuple

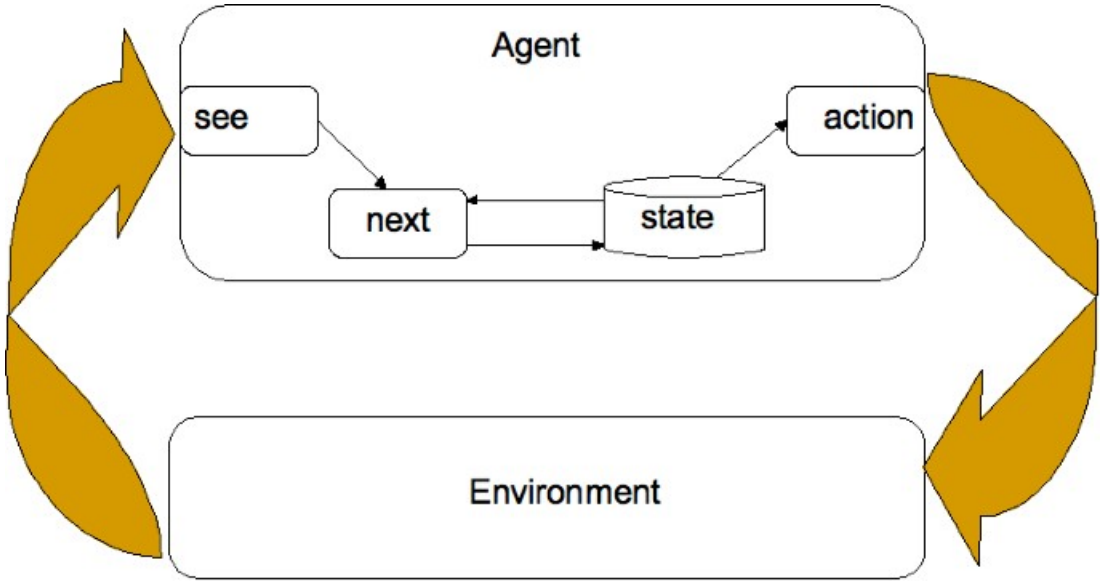
$$< I, E, P, A, i_0, see, next, do, action >$$

where

- $I$ is a set of internal states
- $E, P, A, see$ and $do$ are the same as for reactive agents
- $i_0$ is the initial internal state
- $action : I \times P \rightarrow A$
- $next : I \times P \rightarrow I$

These agents observe the environment ($see$), choose their next action according to their internal state ($action$), act ($do$), and update their internal state (next).

## Adaptive/proactive agent




Thus, a **simple adaptive agent** uses its memory to store observations of the state of the system and then it uses these observations to change its behavior.

**Proactive agents** can be defined by a 9-tuple

$$< D, E, P, A, d_0, see, update, do, action >$$

where

- $D$ is a set of predicate calculus (knowledge base)
- $E, P, A, see$ and $do$ are the same as for reactive agents
- $d_0$ is the initial predicate
- $action : D \times P \rightarrow A$
- $update : D \times P \rightarrow D$

These agents predicate observe the environment (*see*), choose their next action according to some kind of **reasoning** on their internal state (*action*), act (*do*), and update their knowledge base (*update*).

## Multiagent system

A multiagent system consists of a collection of **interactive agents** and a **set of coordination rules**.

An agent has a name and a set of behaviours. He can adopts **different behaviours** at differents times depending on the environmental conditions.

**A well-designed MAS is one that achieves a global task through the actions and interaction of its agents.**

Main design steps consist:

- decomposing a global task into distributable subcomponents, yielding tractable tasks for each agent;
- establishing communication channels that provide sufficient information to each agent to achieve its task; and
- **coordinating the agents** in a way they cooperate on the global task, or at the very least, does not allow them to pursue conflicting strategies in trying to achieve their tasks.

The fundamental problem, in analyzing/designing such MAS, is in determining **how the combined actions** of a large number of agents, leads to **coordinated behavior** on the global task.

"... Coordination is the process of building programs by gluing together active pieces"

[*Coordination languages and their significance,* Carriero and Gelernter, Communication of ACM, 1992]

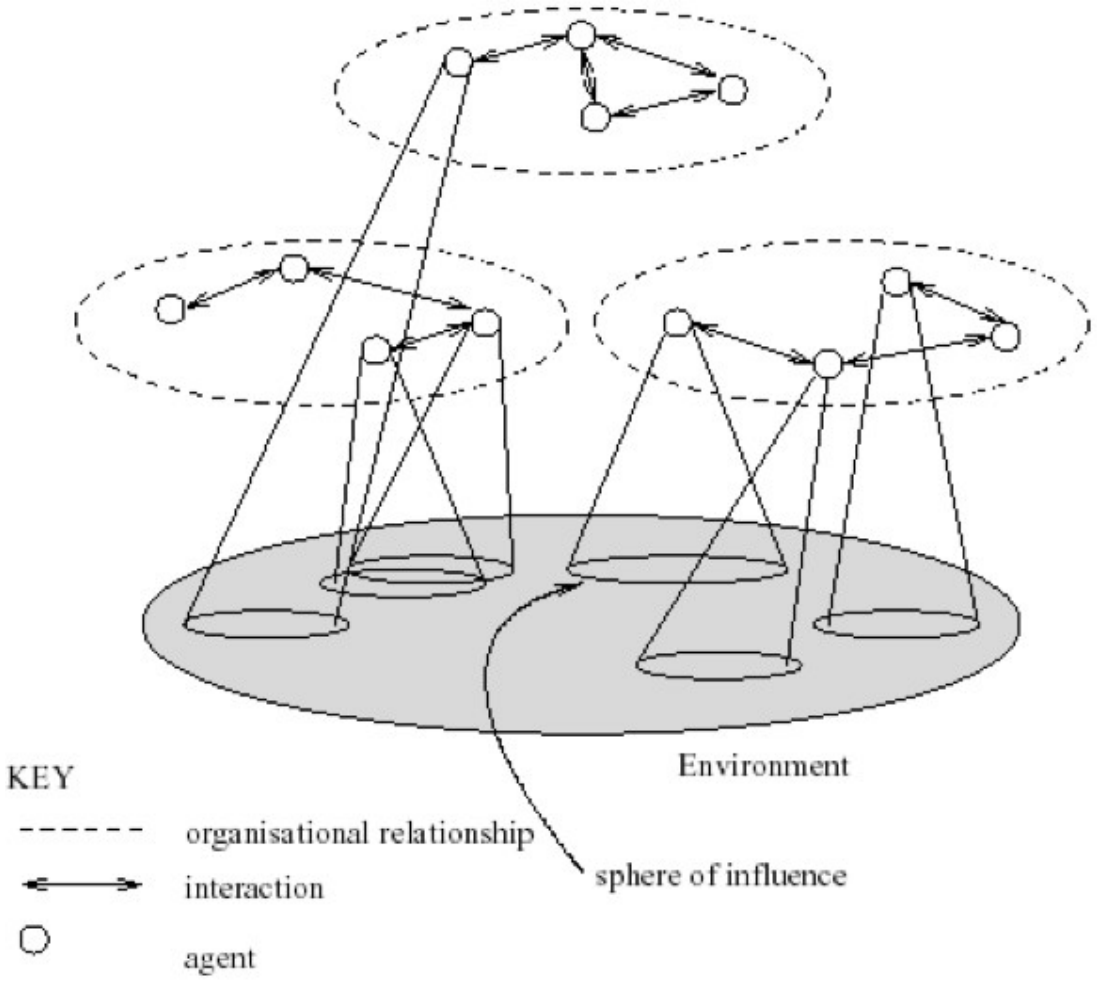
"... Coordination is managing dependencies between activities"

[*The interdisciplinary study of coordination,* Malone and Crowston, ACM Computing Survey, 1994]

"**A coordination model** is the glue that binds separate activities into an ensemble"

## Coordination models

- **Message passing models** allow to coordinate processes that can communicate with other processes through channels or ports on which messages are sent and received
- **Tuple space models** allow to coordinate processes through a common space used to communicate in an temporal and referential uncoupled manner



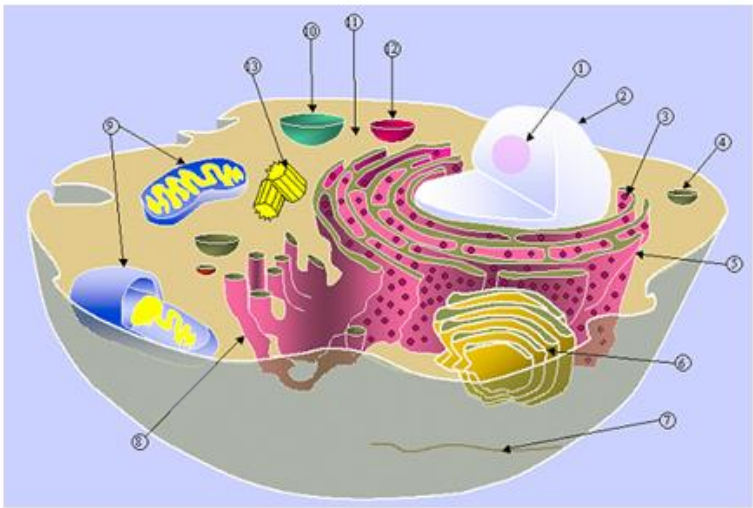

[*An Introduction to MultiAgent Systems* by Michael Wooldridge, John Wiley & Sons, 2002]

# the Cell

The Carbohydrate Oxidation is the energy production process performed by two active components of the Cell: Cytoplasm and Mitochondrion

Cell Components

- Cytoplasm
- Mithocondrion
  - Mithocondrial Matrix
  - Mithocondrial Inner Membrane

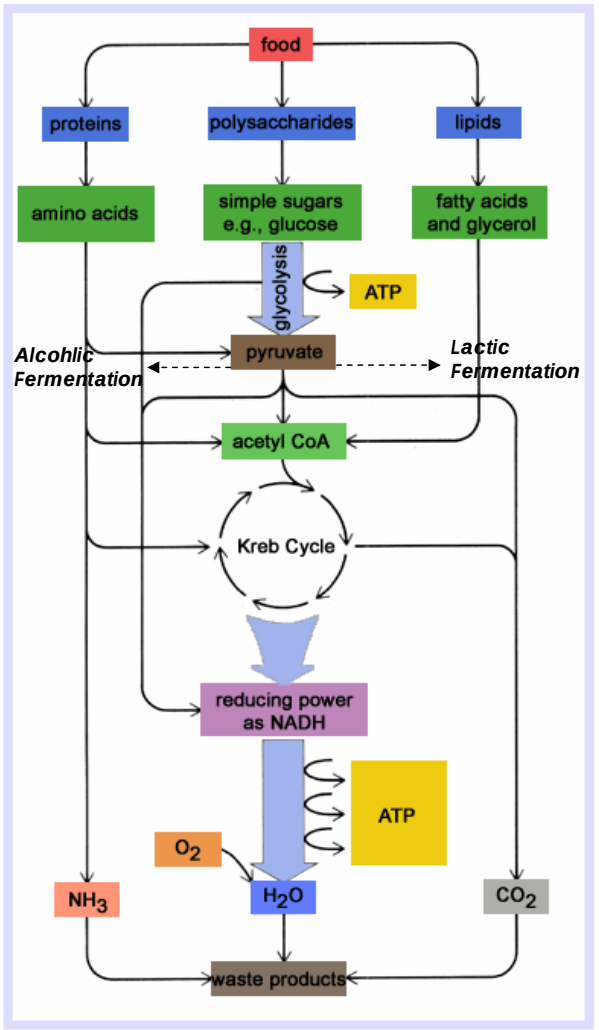


**Cell metabolism** is the process by which individual cells process nutrient molecules.

The series of chemical reactions within a cell that realise cell metabolism are called **metabolic pathways**.

**Carbohydrate oxidation is a set of metabolic pathways** by which a cell produces energy through chemical transformation of carbohydrates like fructose, glucose, mannose, maltose, lactose, saccharose, glycogen and starch

# Carbohydrate Oxidation Cellular Process




**Step 1:** Transformatrion of glucose into pyruvate

**Step 2:** Transformation of acetyl CoA into NADH

**Step 3:** Transportation of NADH into ATP

**Legend**

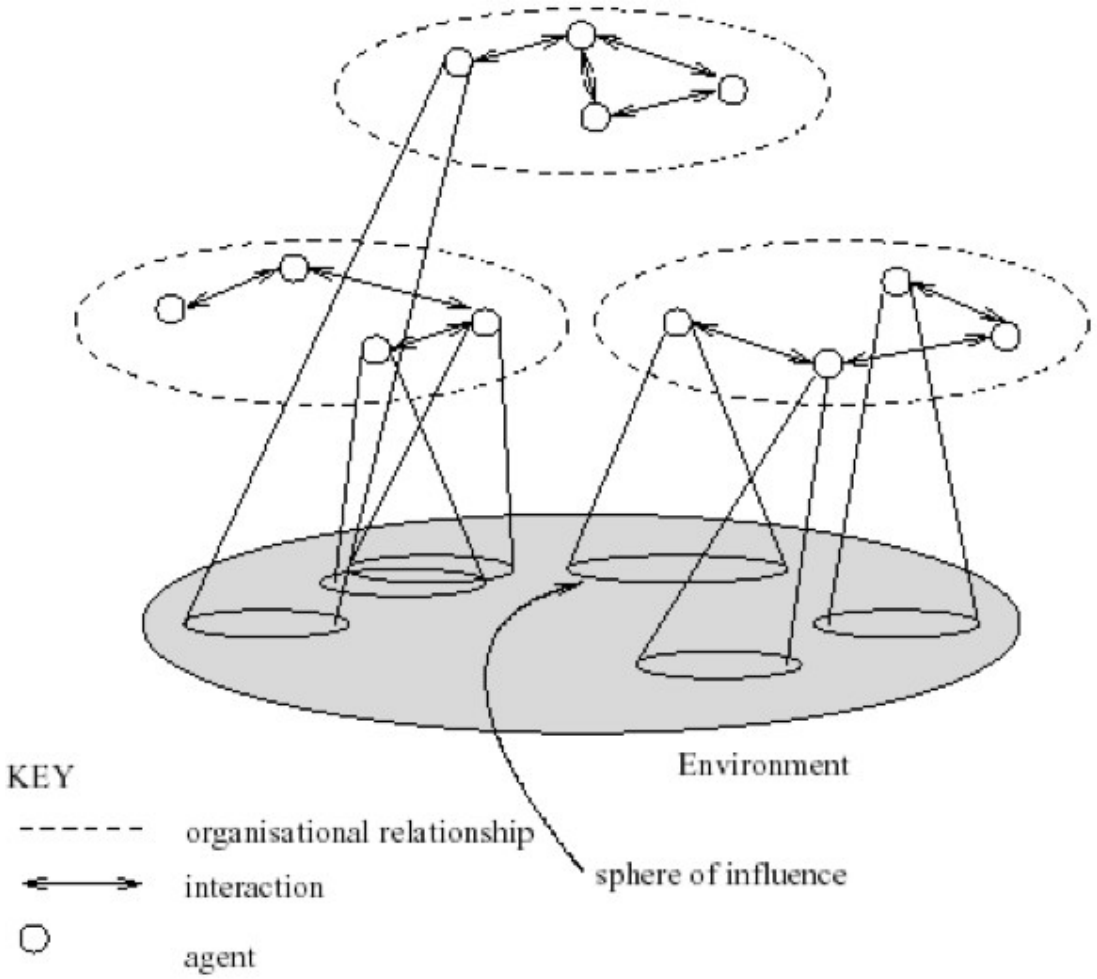
- Metabolic pathway
- Compounds reaction
- Molecules

**Compartment 1:** Cytoplasm

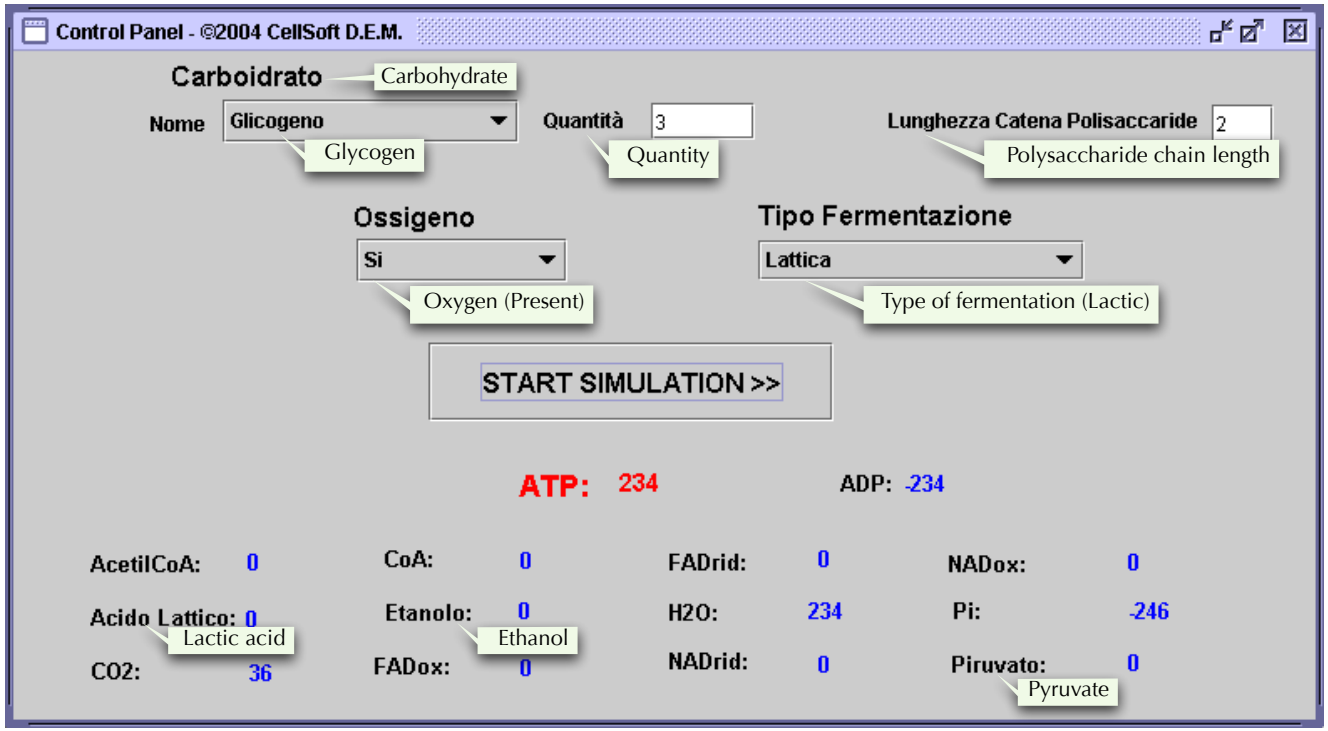
**Compartment 2:** Mitochondrion

**Compartment 2.1:** Mitochondrial Matrix

**Compartment 2.2:** Inner Mitochondrial Membrane


KEY
organisational relationship
interaction
agent
Environment
sphere of influence

# First Simulator: Carbohydrate Oxidation pathway

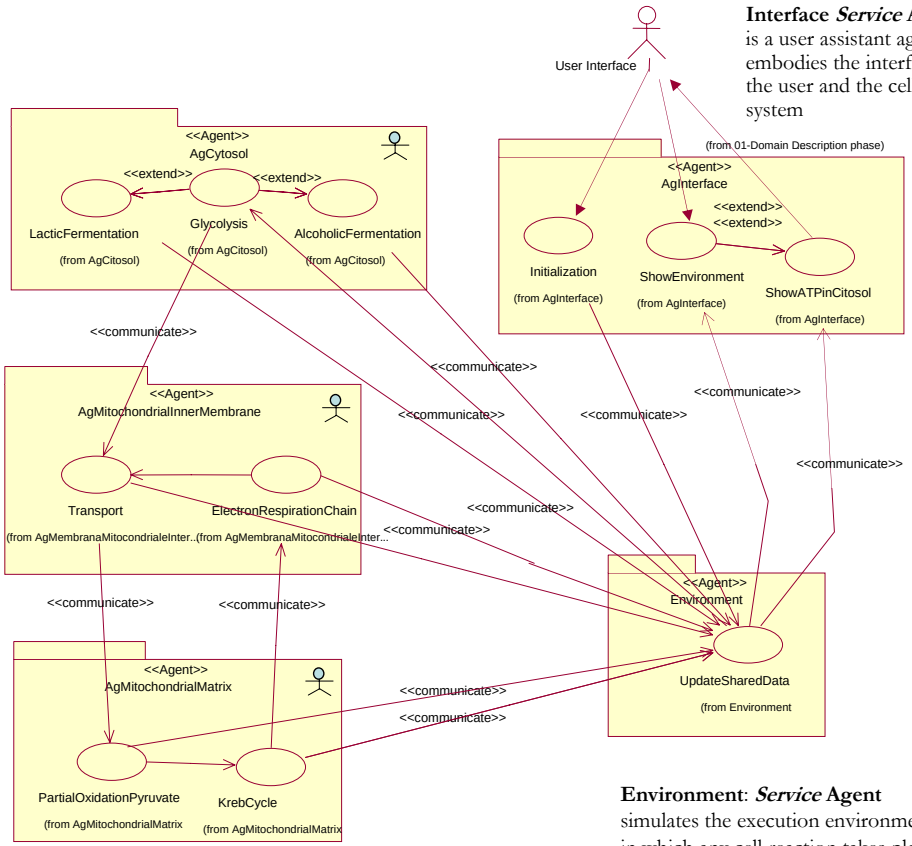


# Carbohydrate oxidation: the agents identification UML diagram

**Agent Citosol** simulates the role of Cytoplasm in the Glycolysis, Lactic and Alcoholic Fermentations functions

**Agent Inner Mithocondrial Membrane** simulates the role of IMM in the Transport and Electron Respiration Chain functions

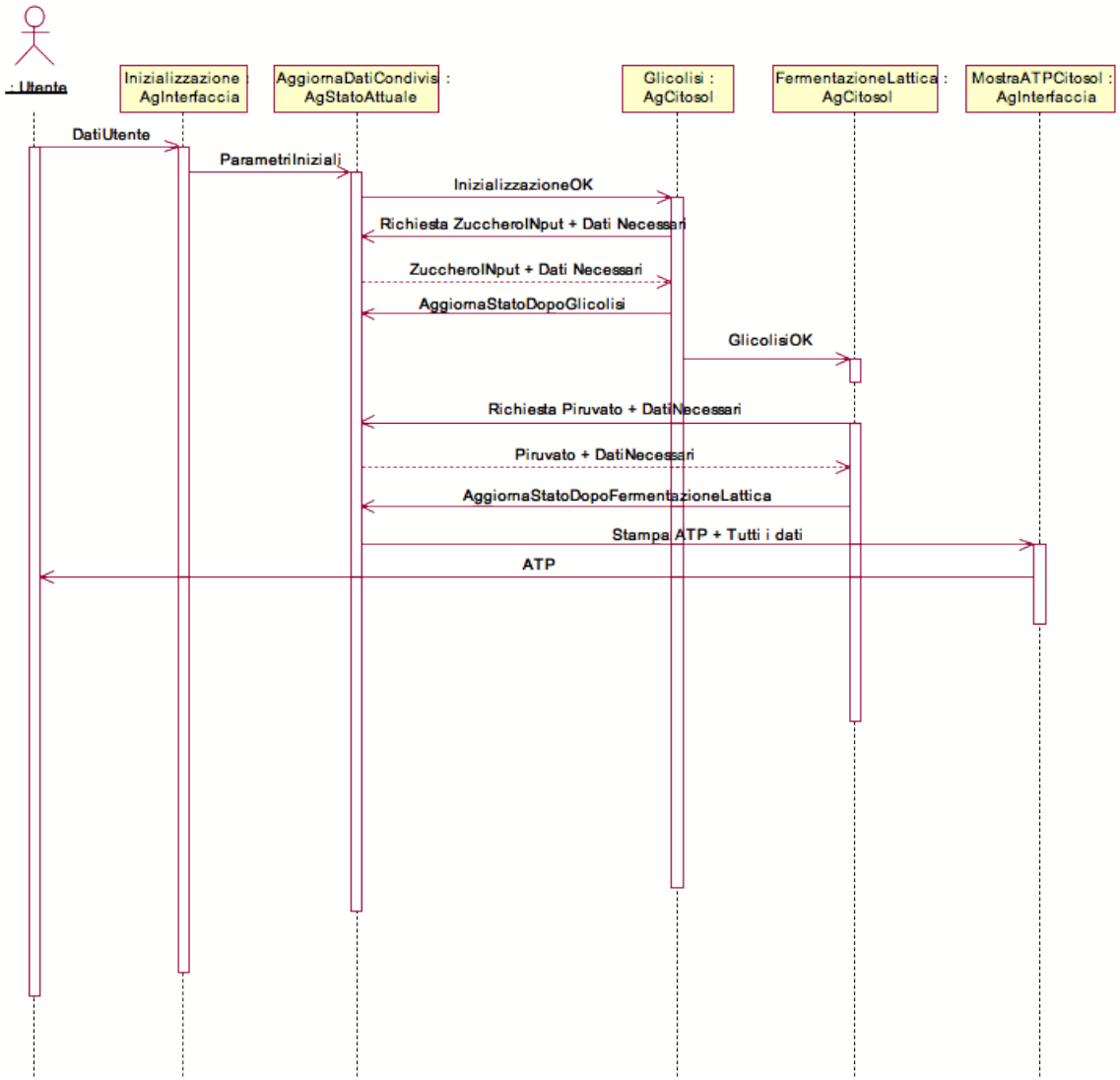
**Agent Mithocondrial Matrix**: simulates the role of MM in the Partial Oxidation of Pyruvate and Kreb's Cycle functions…




**Interface *Service* Agent** is a user assistant agent. It embodies the interface between the user and the cell simulation system

**Environment**: ***Service*** **Agent** simulates the execution environment in which any cell reaction takes place

# Agent roles and interaction in the Lactic Fermentation function



## Model validation

The **validation of the model**, based on **mutation testing**, imposes a new requirement on the model:

*the model simulation should give reasonable results when the model is altered in ways that correspond to known or* **plausible mutations**

## MAS Valitation by mutation

Carbohydrate oxidation can take place within two different environmental conditions:

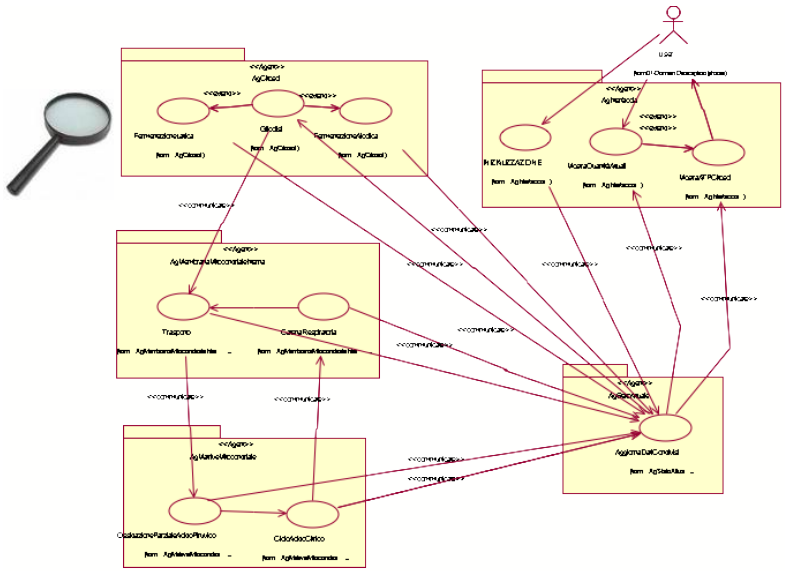
- in presence of oxygen (aerobic) or
- in its absence (anaerobic)

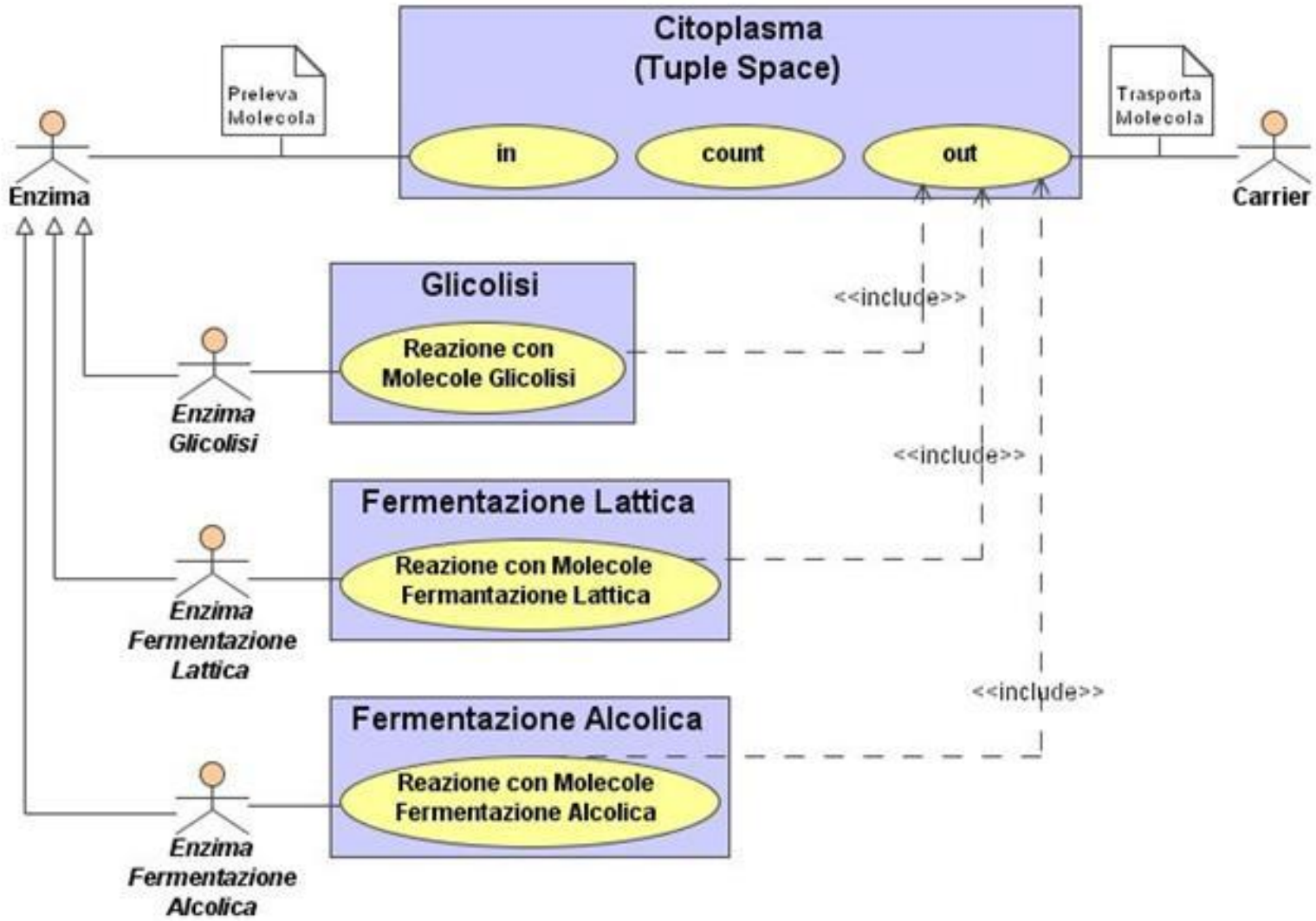
The former takes place in the mitochondria, the latter in the cytoplasm.

There are some cases where as consequence of malfunction due to **DNA mutation in the mitochondrion** the aerobic pathway is blocked and metabolism is forced to change behavior with respect to new condition.

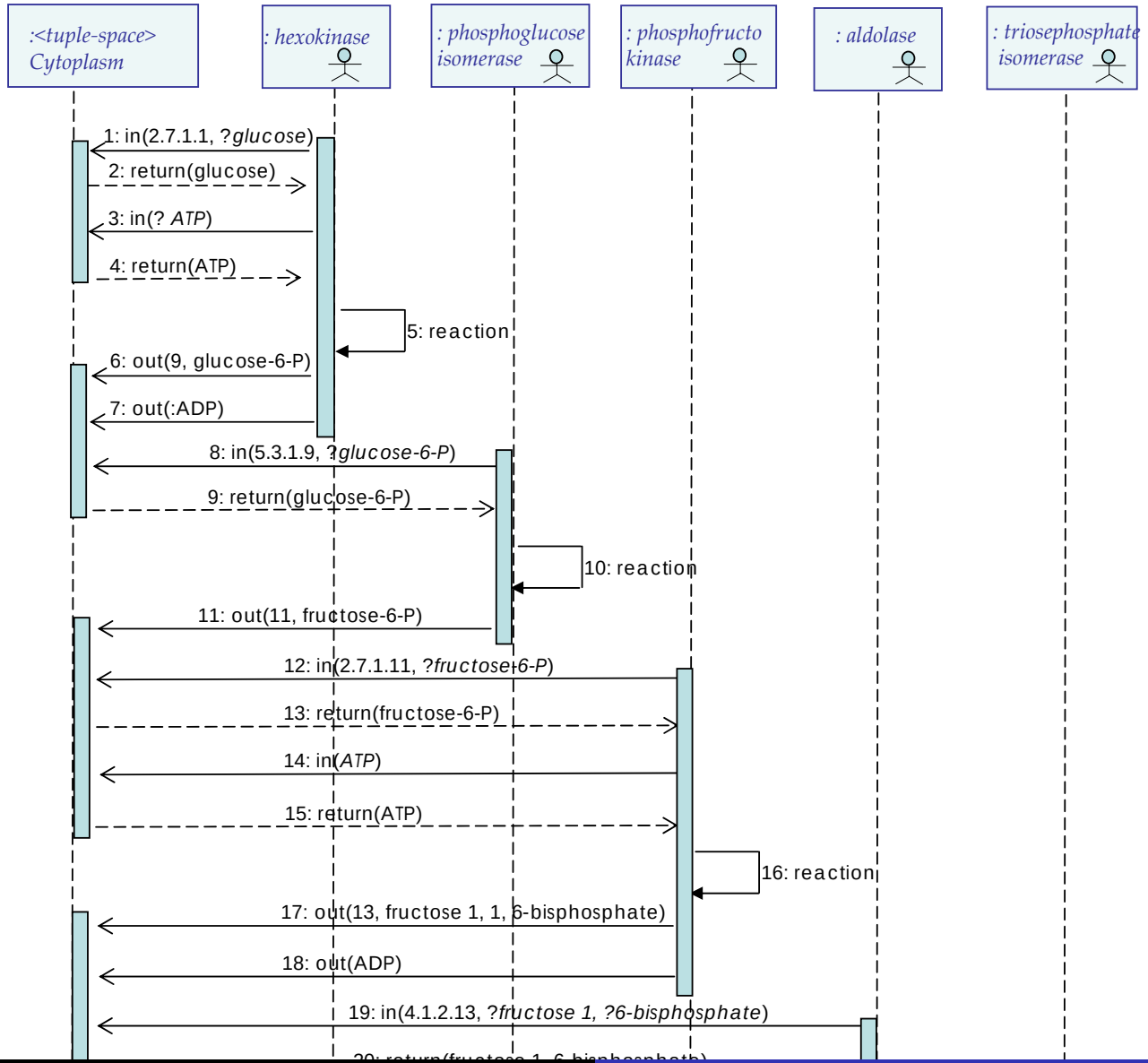
We need to refine the model at a fine grain abstraction!

. . . thus we concentrate on Cytoplasm metabolic processes by zooming-in the Cytoplasm

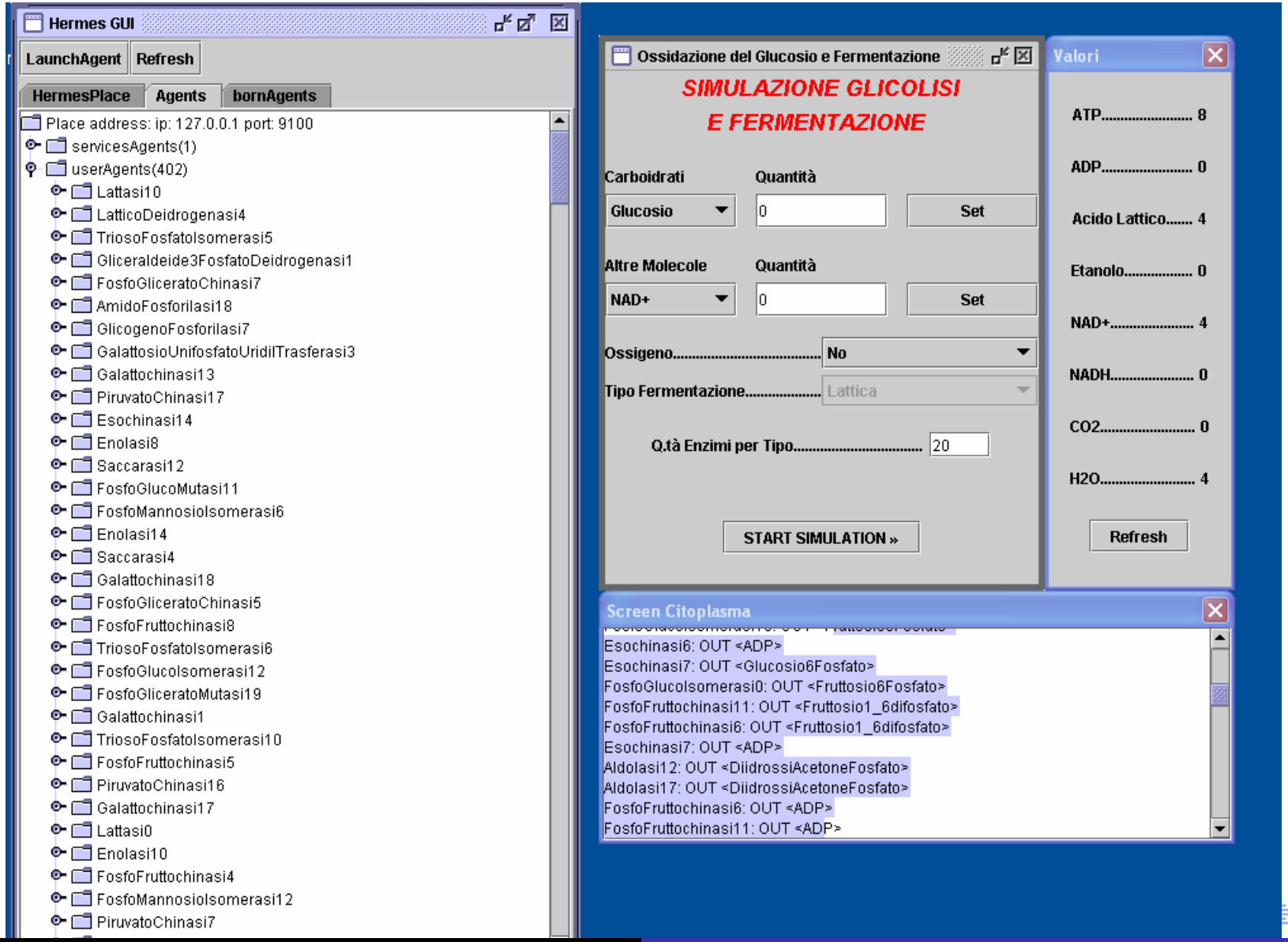



Citoplasma
(Tuple Space)
in
count
out
Preleva
Molecola
Trasporta
Molecola
Enzima
Carrier
<<include>>
Glicolisi
Reazione con
Molecole Glicolisi
Enzima
Glicolisi
<<include>>
Fermentazione Lattica
Reazione con Molecole
Fermantazione Lattica
Enzima
Fermentazione
Lattica
<<include>>
Fermentazione Alcolica
Reazione con Molecole
Fermentazione Alcolica
Enzima
Fermentazione
Alcolica

# UML Sequence diagram of the preliminary phase of glycolysis

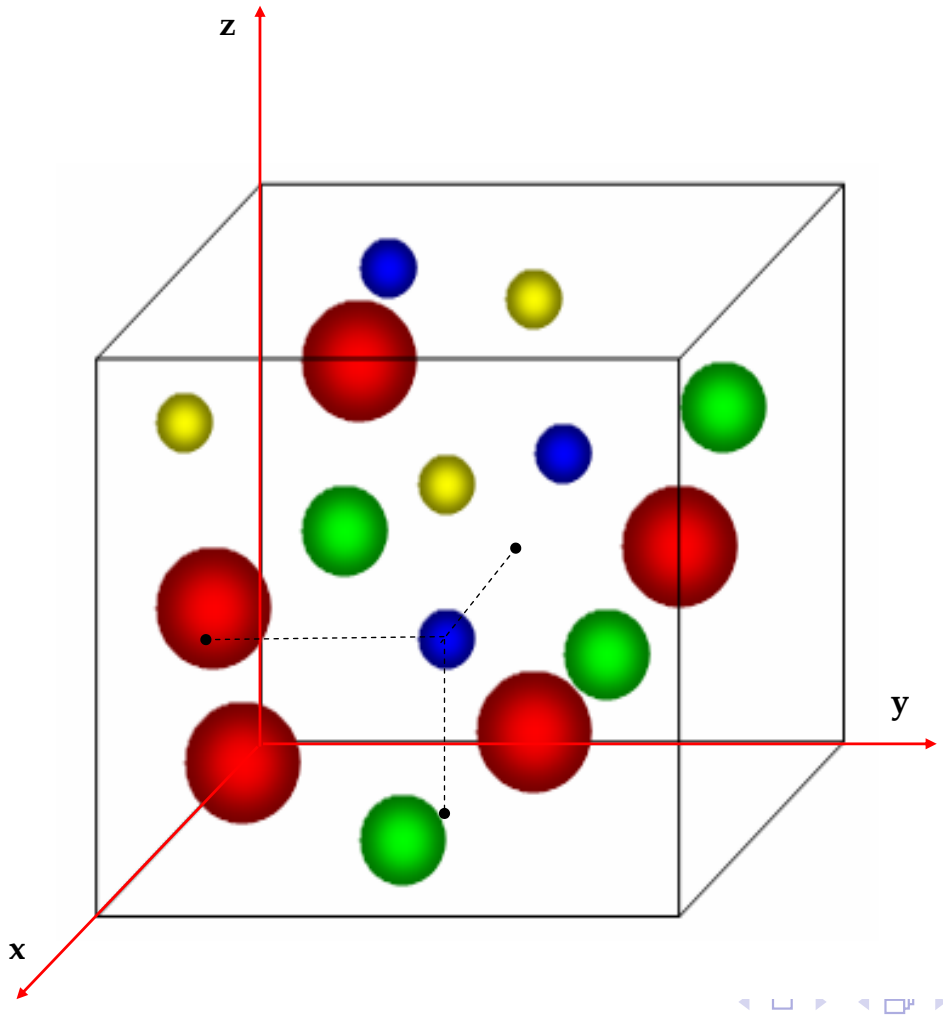


# Second Simulator: Glycolysis pathway

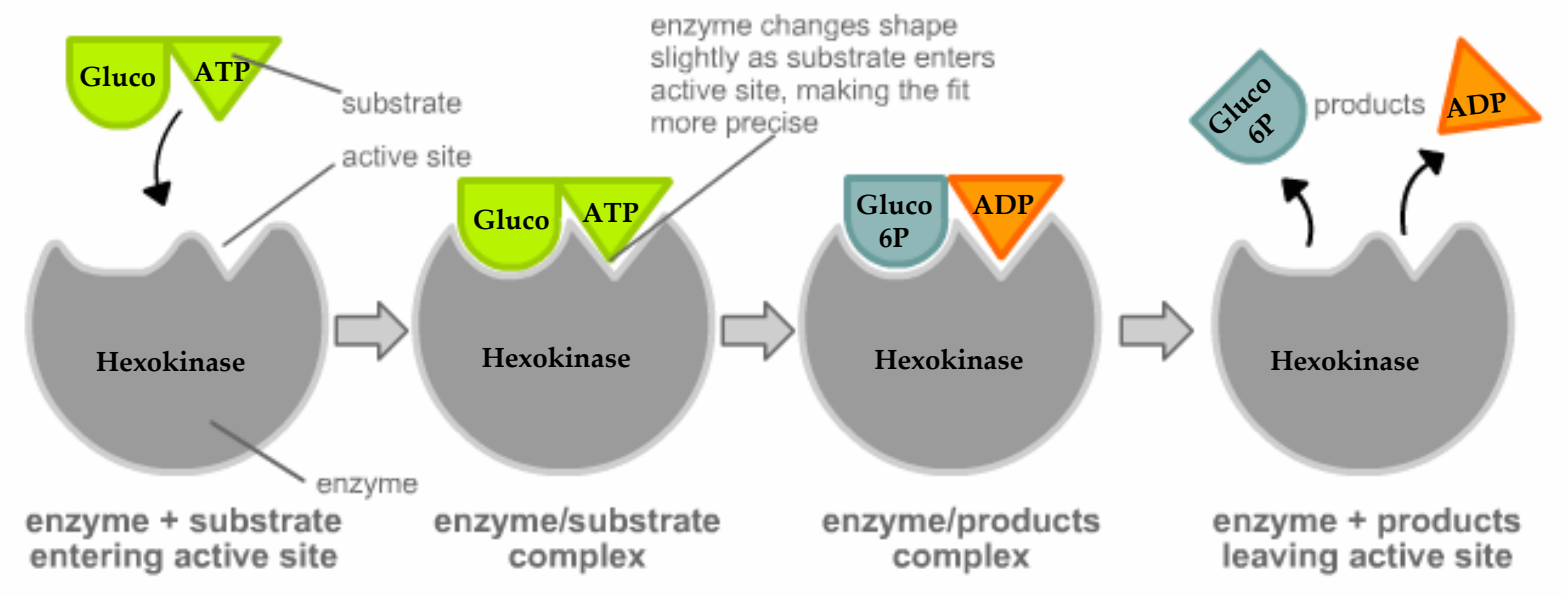


# Model validation

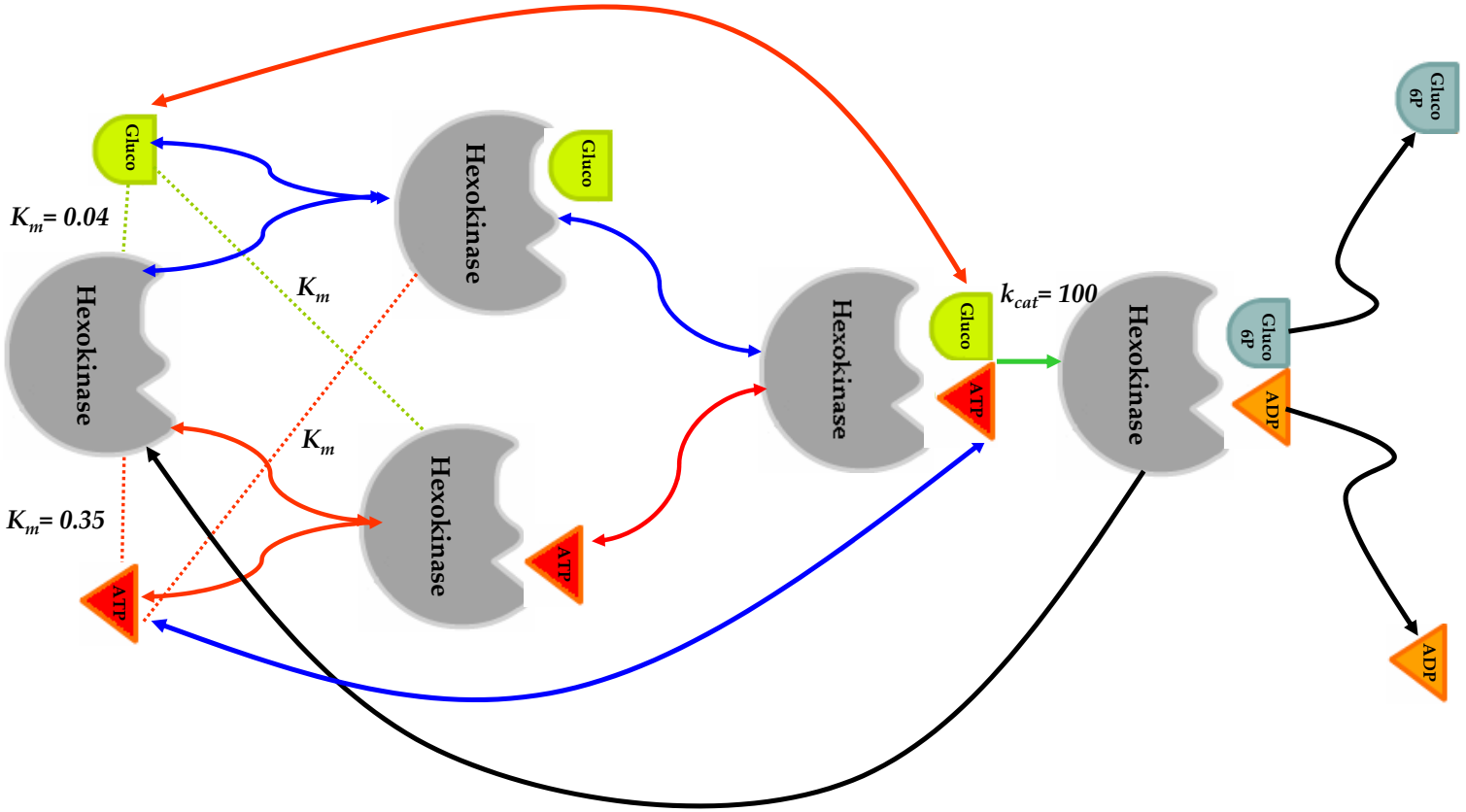
But ... molecules move



# ... and then interact by vicinity

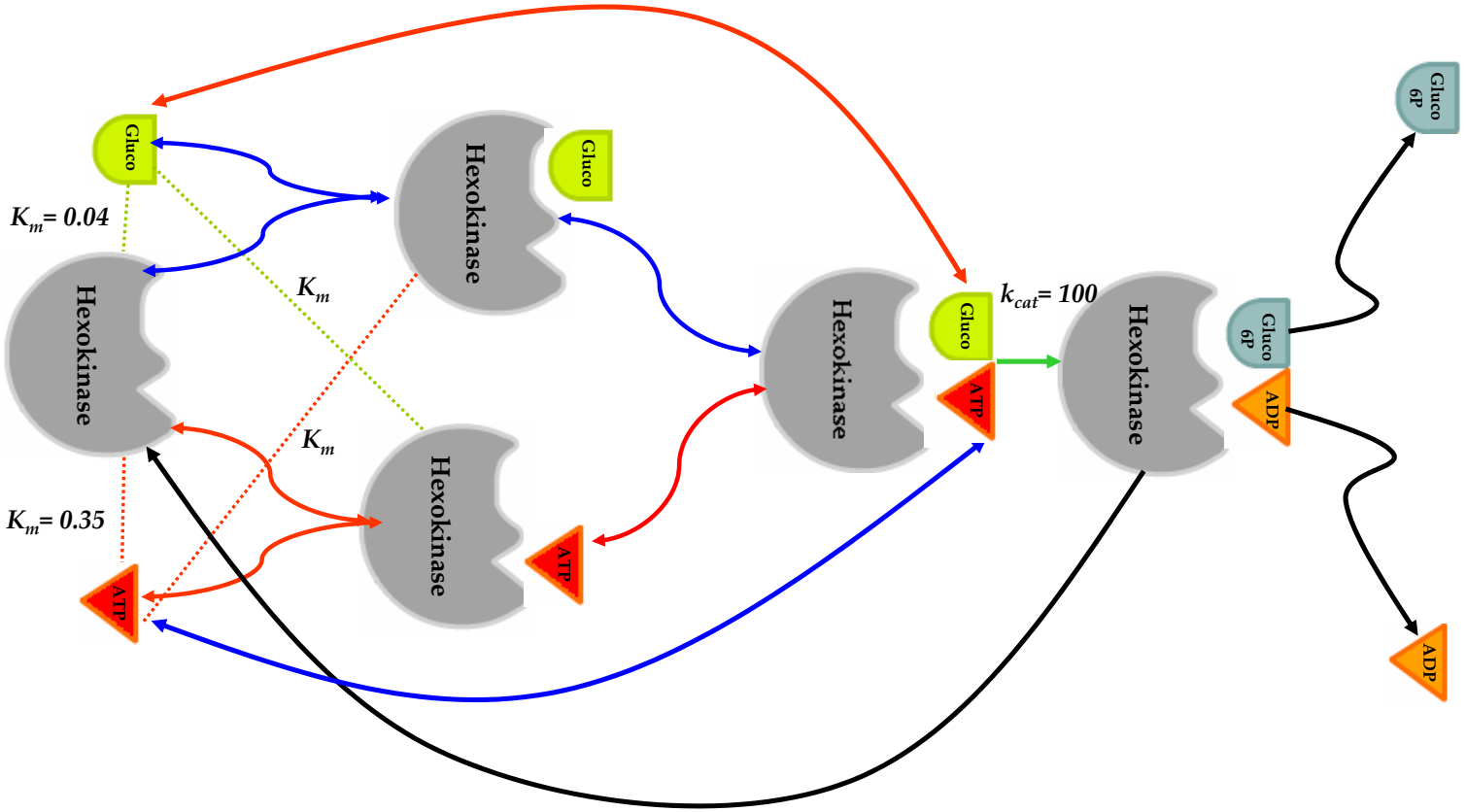


# ... and then interact by affinity



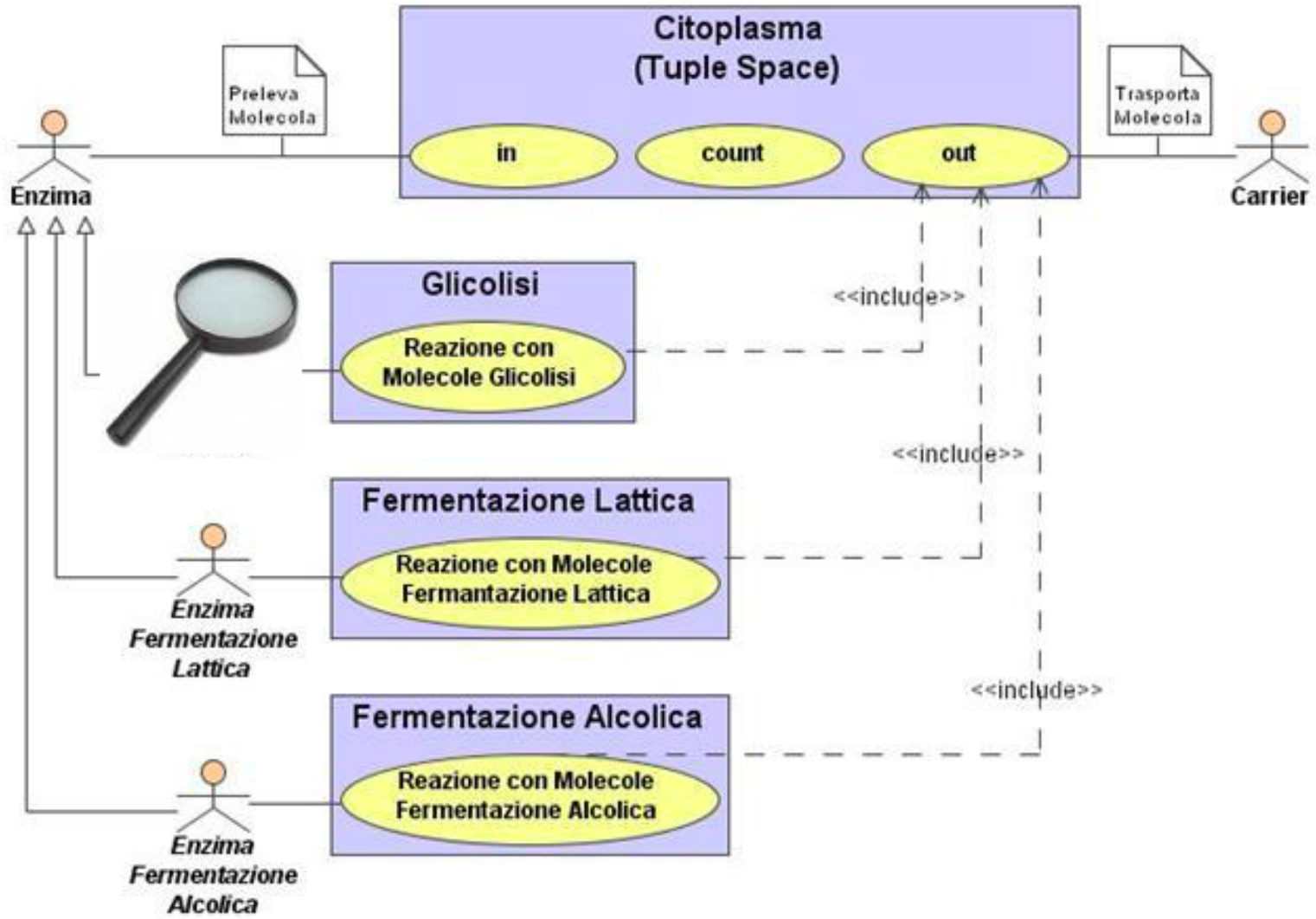

and . . . molecules have also their volume and many other parameters . . .

# ... and then interact by affinity




and . . . molecules have also their volume and many other parameters . . .

# But ... Zooming-in

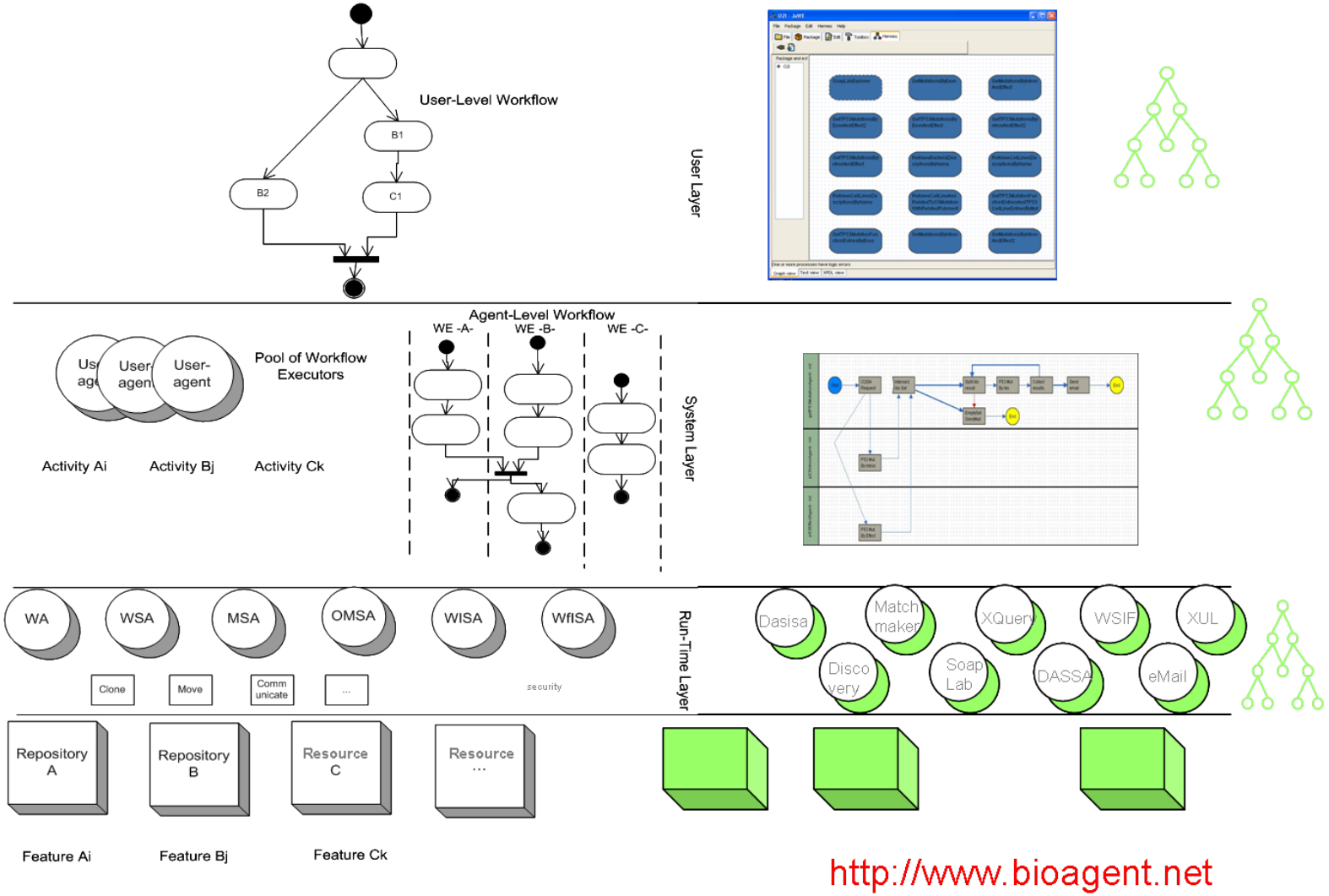


## Third simulator: Orion

- Orion is a MAS simulator over Hermes (an agent-based coordination middleware)
- Orion agents define a virtual **space** and **time** where **biological** entities can **move** and **interact**

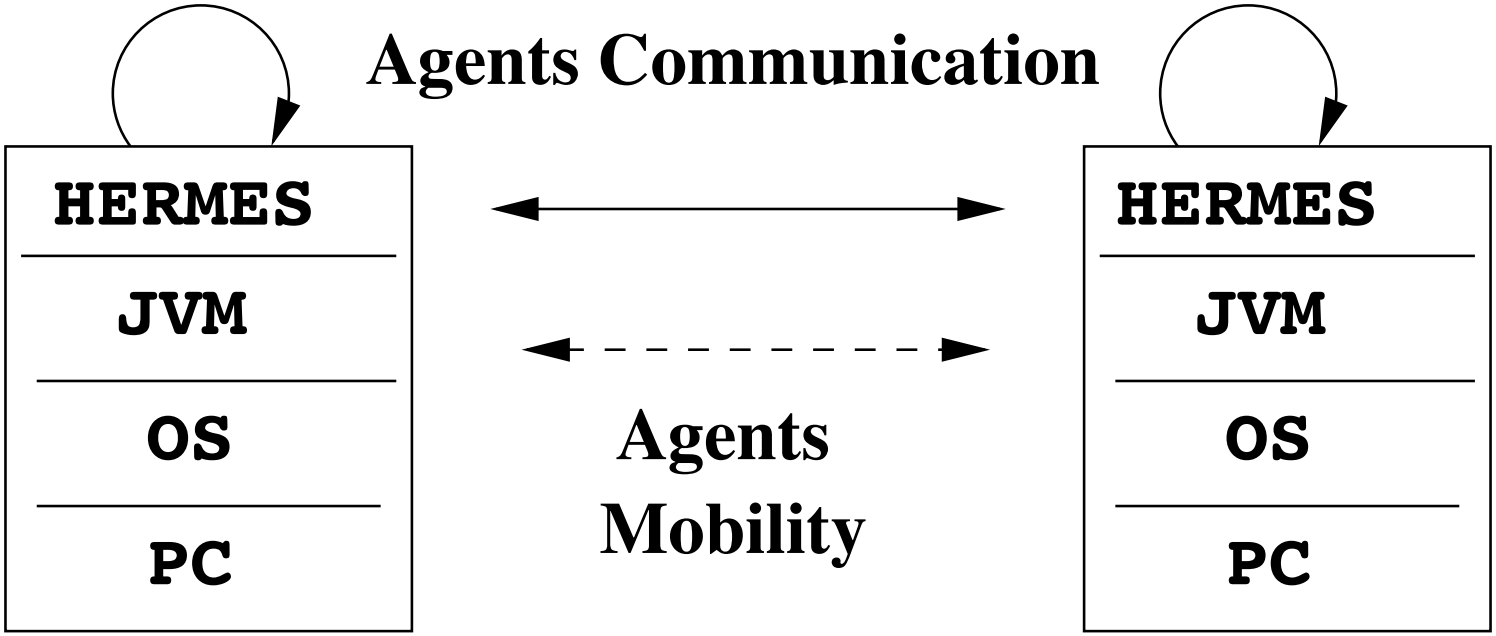
## What is Hermes?

- Hermes is a middleware, 80 Kb of **Java** code, developed at COSY Lab, University of Camerino
- Hermes supports the execution of **Activity-based** and **Distributed** software applications
- You can specialize a class **Agent**. Each Agent executes its own activities in its dedicated **thread**
- Agents live in the Hermes platform and **communicate** with ServiceAgents (discovery, services) or each-other
- Agents can **move** from an Hermes platform to another simply invoking a primitive




http://www.bioagent.net

## Hermes Main Features




Get it at `hermes.cs.unicam.it`

## System behavior emerging from components behavior

- Orion adopts a bottom-up strategy: model small components as autonomous Agents and make them live in a **concurrent** environment with **random** interactions
- Orion would support experimental model validation: correctness of the model (through agents behavior)
- and . . . theoretical validation: accuracy of the approach (concurrent stochastic interactions and movements)

## Glycolysis case study: complexes and reactions

- An enzyme binds its substrate with a given affinity, becoming a **Complex**
- When all of the substrate is bound, the reaction happens at the given velocity: products are released and the enzyme is ready for another binding
- Enzymes have **affinity constants** ($K_m$) for each substrate they can bind to
- The quantity of products in the time unit is a function of the concentrations and an **enzyme-constant** ($K_{cat}$)

## Glycolysis Pathway

- An ubiquitous pathway in life: produce ATP from Glucose
- A pathway is composed by a series of reactions in wich **metabolites** and **enzymes** are involved

Step 1: Hexokinase + Glucose + ATP $\longrightarrow$ Glucose 6-Phos. + ADP

- Hexokinase is an enzyme which promotes the reaction between Glucose and ATP

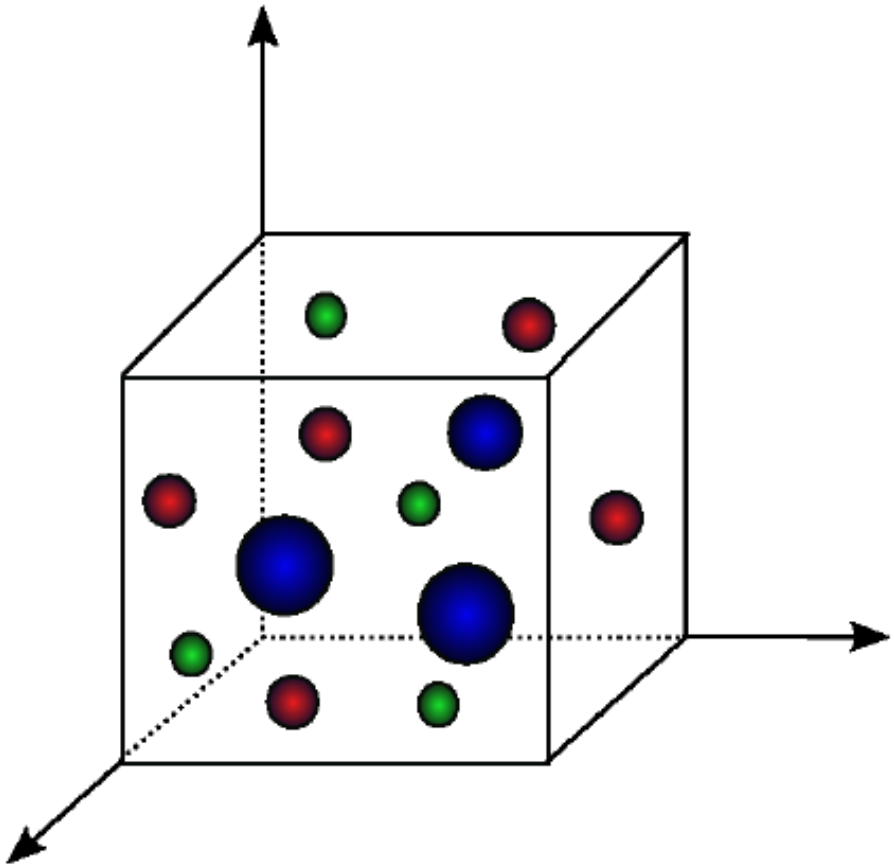
# Motion of the molecules

- We model the cell at mesoscale ($10^{-8}$) level
- At this scale the predominant force is Brownian Motion
- The molecules move in random directions with a calculated offset (Einstein relation)

## Volume of simulation

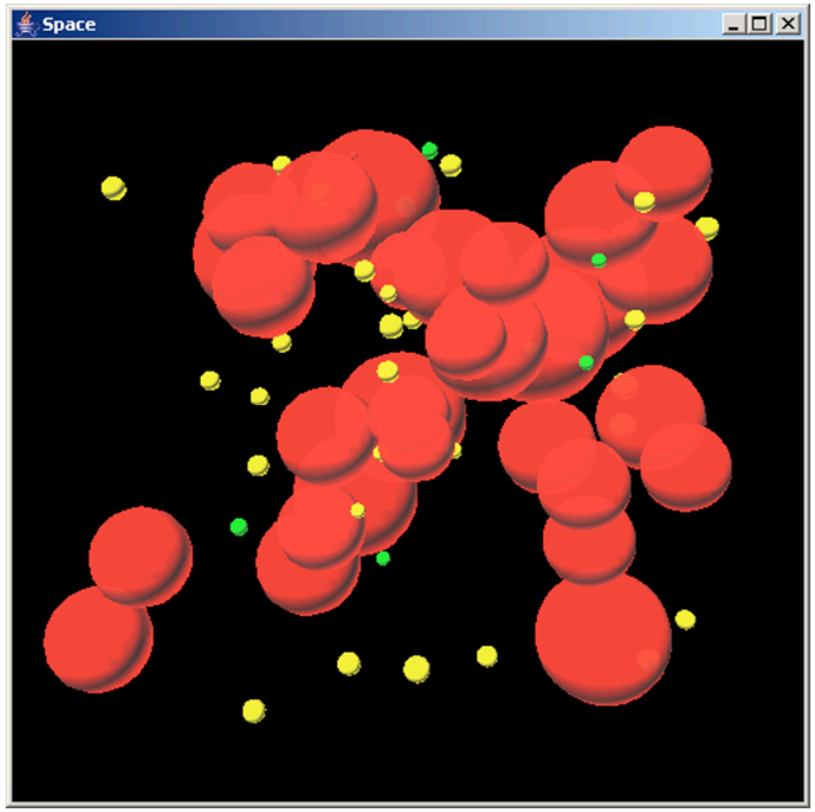
- We simulate the glycolysis pathway in a cube of cytoplasm of a yeast cell
- Temperature is considered constant at 25°C
- Cytoplasm viscosity is considered
- Uniform distribution of the molecules in the cytoplasm is assumed
- Molecules are represented as 3D spheres with a radius proportional to their weight
- A reasonable choice of volume is the femtolitre: $10^{-15}\ell = 10^{-18}m^3$ (the cube side is $10^{-6}m$)

# Cube of simulation



# Orion Project Design

- Model Enzymes and Complexes as Hermes Agents
- Model Metabolites as Java Objects
- Define ServiceAgents for managing space and time
- All move according to Brownian Motion
- Enzymes and Complexes perceive metabolites in their action radius
- Probabilistically (using the affinities $K_m$) Enzymes and Complexes bound metabolites
- Complexes react producing new metabolites in a given time (inverse of $K_{cat}$) and become Enzymes again




A snapshot of a simulation in Orion, showing the molecule agents moving and acting in a portion of cytoplasm

## Getting biological data

- Orion uses an XML database (eXist) to store the biological data
- Biological data about the pathway and the molecules are stored in SBML: Systems Biology Markup Language - `www.sbml.org`
- Data for different pathways and molecules are easy to import

# A sample of the SBML description

```
<pathway organism=Saccharomyces_cerevisiae
id=glycolysis>
 <list_molecules> .... </list_molecules>
 <list_reactions>
  <reaction kcat=50.0>
   <list_interactions>
    <interaction>
          <reactants>
           <reactant id=GLC/>
           <reactant id=HEX/>
          </reactants>
          <products>
        <product id=HEX+GLC/>
          </products>
     <Km_constant>0.028</Km_constant>
    </interaction>
    <interaction>
          <reactants>
           <reactant id=ATP/>
           <reactant id=HEX/>
          </reactants>
          <products>
        <product id=HEX+ATP/>
          </products>
     <Km_constant>0.05</Km_constant>
```

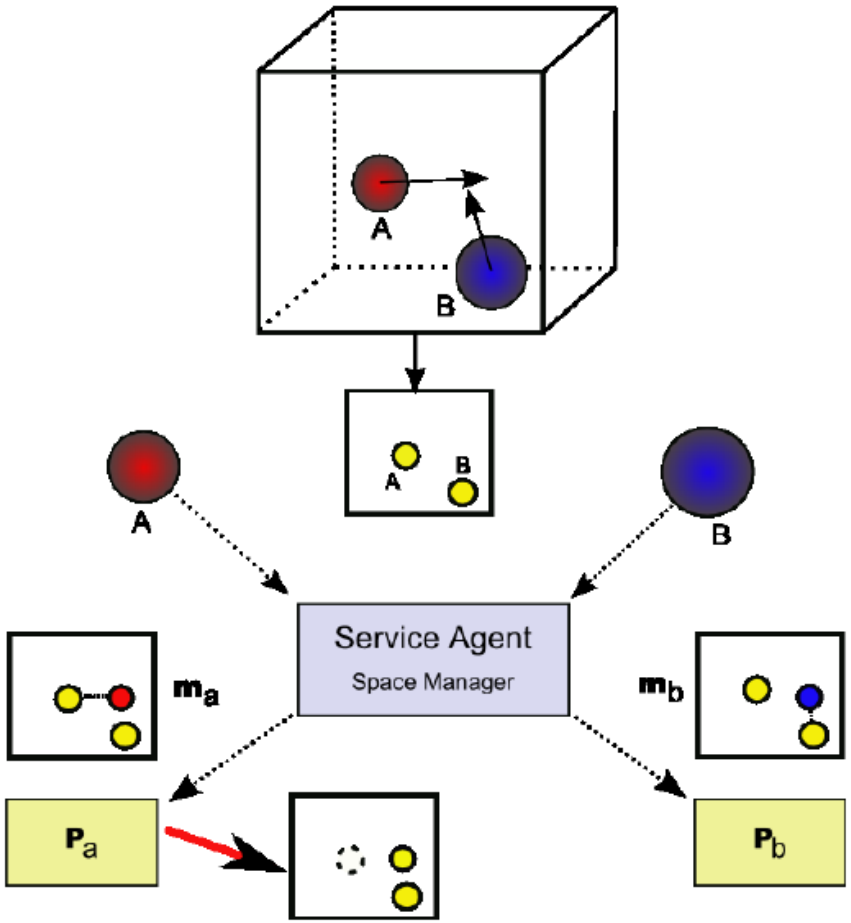
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <sbml xmlns="http://www.sbml.org/sbml/level2"
xmlns:jd="http://www.sys-bio.org/sbml"
xmlns:sl2="http://projects.eml.org/bcb/sbml/level2" metaid="metaid_0000001"
level="2" version="1"> - <model metaid="metaid_0000002" id="Glycolysis_Nielsen"
name="Nielsen1998_Glycolysis"> - <notes> - <body
xmlns="http://www.w3.org/1999/xhtml">
  <p>Reference: Nielsen et al; Biophys. Chem. (1998) 72:49-62</p>
- <p>
  The reaction looks like this:
  <br />
  reaction_1: GLC + ATP -> F6P + ADP;
  <br />
  reaction_2: F6P + ATP -> FBP + ADP;
  <br />
  reaction_3: FBP => 2 * GAP;
  <br />
  reaction_4: GAP + NAD -> DPG + NADH;
  <br />
  reaction_5: DPG + ADP => PEP + ATP;
  <br />
  reaction_6: PEP + ADP -> PYR + ATP;
  <br />
  reaction_7: PYR -> ACA;
  <br />
  reaction_8: ACA + NADH => EtOH + NAD;
  <br />
  reaction_9: AMP + ATP => 2 * ADP;
  <br />
  reaction_10: F6P -> P;
  <br />
  flow reactor:
  <br />
```

```
 - <reaction metaid="metaid_0000059" id="reaction_1"
reversible="false"> - <annotation> - <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"> - <rdf:Description
rdf:about="#metaid_0000059"> - <dc:relation> - <rdf:Bag>
  <rdf:li rdf:resource="http://www.ebi.ac.uk/IntEnz/#EC 2.7.1.2" />
  <rdf:li rdf:resource="http://www.ebi.ac.uk/IntEnz/#EC 5.3.1.9" />
  <rdf:li rdf:resource="http://www.genome.jp/kegg/reaction/#R00771" />
  <rdf:li rdf:resource="http://www.genome.jp/kegg/reaction/#R00299" />
  <rdf:li rdf:resource="http://www.reactome.org/#70112" />
  </rdf:Bag>
  </dc:relation>
  </rdf:Description>
  </rdf:RDF>
  </annotation>
- {\bf <listOfReactants>
  <speciesReference species="GLC" />
  <speciesReference species="ATP" />
  </listOfReactants>
- <listOfProducts>
  <speciesReference species="F6P" />
  <speciesReference species="ADP" />
  </listOfProducts> }
- <kineticLaw> - <math xmlns="http://www.w3.org/1998/Math/MathML"> - <apply>
  <times />
 ...
  </kineticLaw>
  </reaction>
```

## Agents

- Enzymes and Complexes are Agents
- Metabolites are Objects whose movement is managed by a ServiceAgent
- **Space** occupation and movements are managed by a ServiceAgent (bumps currently not permitted) which resolves conflicts randomly
- **Time** impulses (1 tick is 1 millisecond, for glycolysis) are given by a ServiceAgent when **all** the agents and metabolites have finished their moves and reactions in the current time slice

# Space Manager



## Dimensions of the simulation

- In the simulation volume the higher and dominant concentration is those of ATP
- Using bio-data we calculate approximatively 2 700 000 ATP molecules in the space
- They have to be maintained in memory and moved according to Brownian Motion in each time slice
- Moreover other metabolites have to be managed
- Enzymes and Complexes also compete for space
- A standalone workstation cannot perform the simulation in a reasonale space and time

## Cube splitting

- Our solution (work in progress) is to decompose the cube of simulation in smaller cubes
- Each cube manages a slice of the virtual space
- Cube splitting must be transparent to Agents
- Cubes boundaries are guarded by ServiceAgents who implement the transparent splitting
- Coordination models and languages are needed to achieve the goal

## Deployment

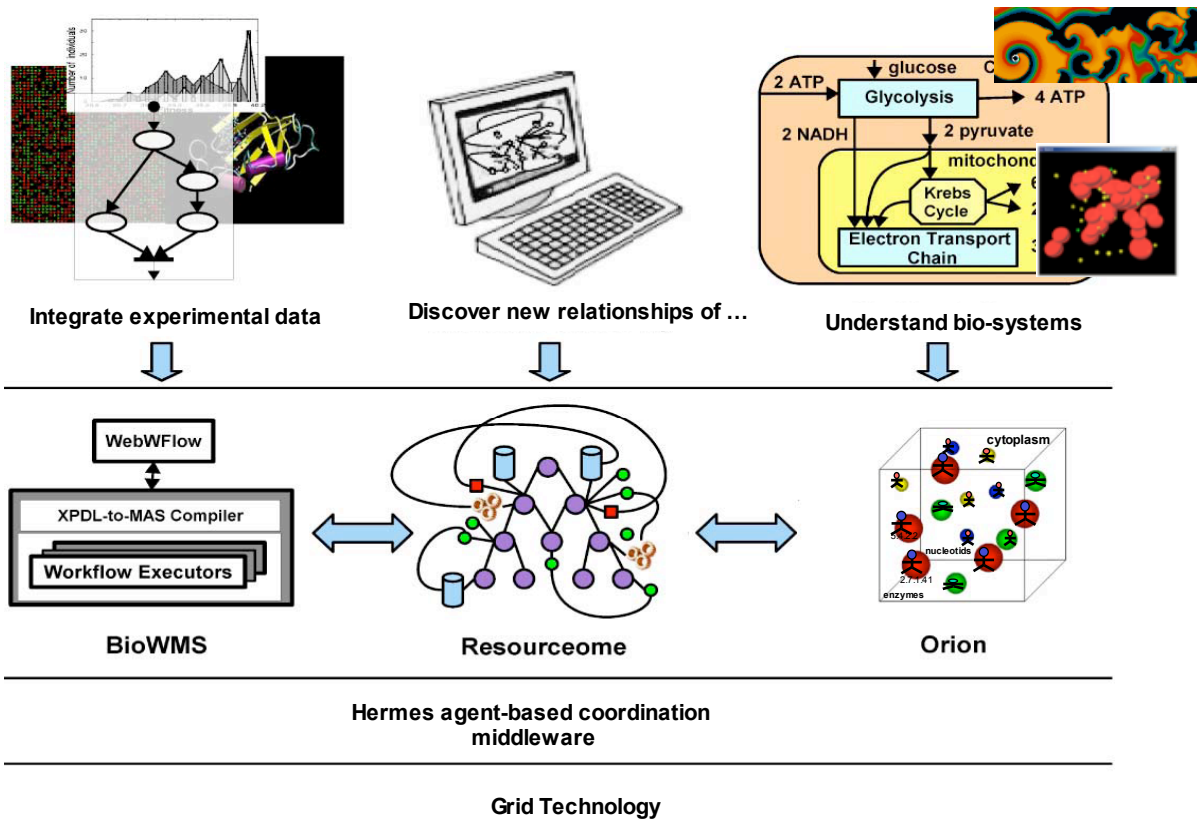
- LitBio: Laboratory of Interdisciplinary Technologies in Bioinformatics - `litbio.cs.unicam.it`
- Collaborating with Cilea (`www.cilea.it`)
- Provides (work in progress) a **GRID** solution for Orion
- Each small cube is managed by an instance of Orion in a GRID node
- Coordination models guarantee the correctness and consistency of the simulation

**Grid** is emerging technology for sharing geographically distributed computing, network, storage, data and resources

The **Grid** and **agent communities** both develop concepts and mechanisms for open distributed systems, albeit from different perspectives.

The **Grid community** has historically focused on **"brawn"**: infrastructure, tools, and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations. In contrast, the **agents community** has focused on **"brain"**: autonomous problem solvers that can act flexibly in uncertain and dynamic environments.

# LITBIO Project: Unicam Framework Architecture




[litbio.cs.unicam.it]

## We Another simulator . . . **Kalliope**

Kalliope has been developed in X-Klaim and run over Klava framework

- a Klaim Tuple Space (TS) represents the Cytoplasm
- the Cytoplasm molecules are tuples in the TS
- a set of Klaim agents manipulated the TS to simulate the Cytoplasm reaction during the Glycolysis

# LINDA model

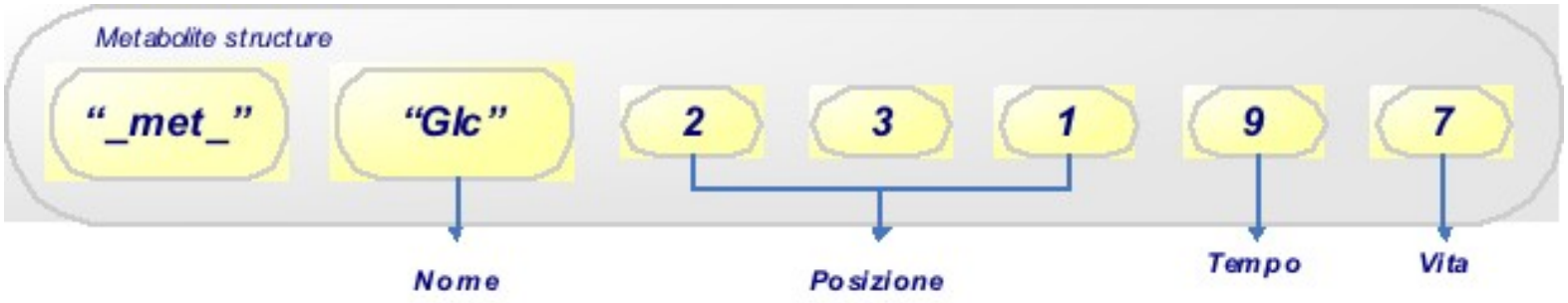
Based on generative communication: processes (agents) communicate by inserting or retrieving data objects (called Tuples) from shared Data Space ( called Tuple space)

## Operations

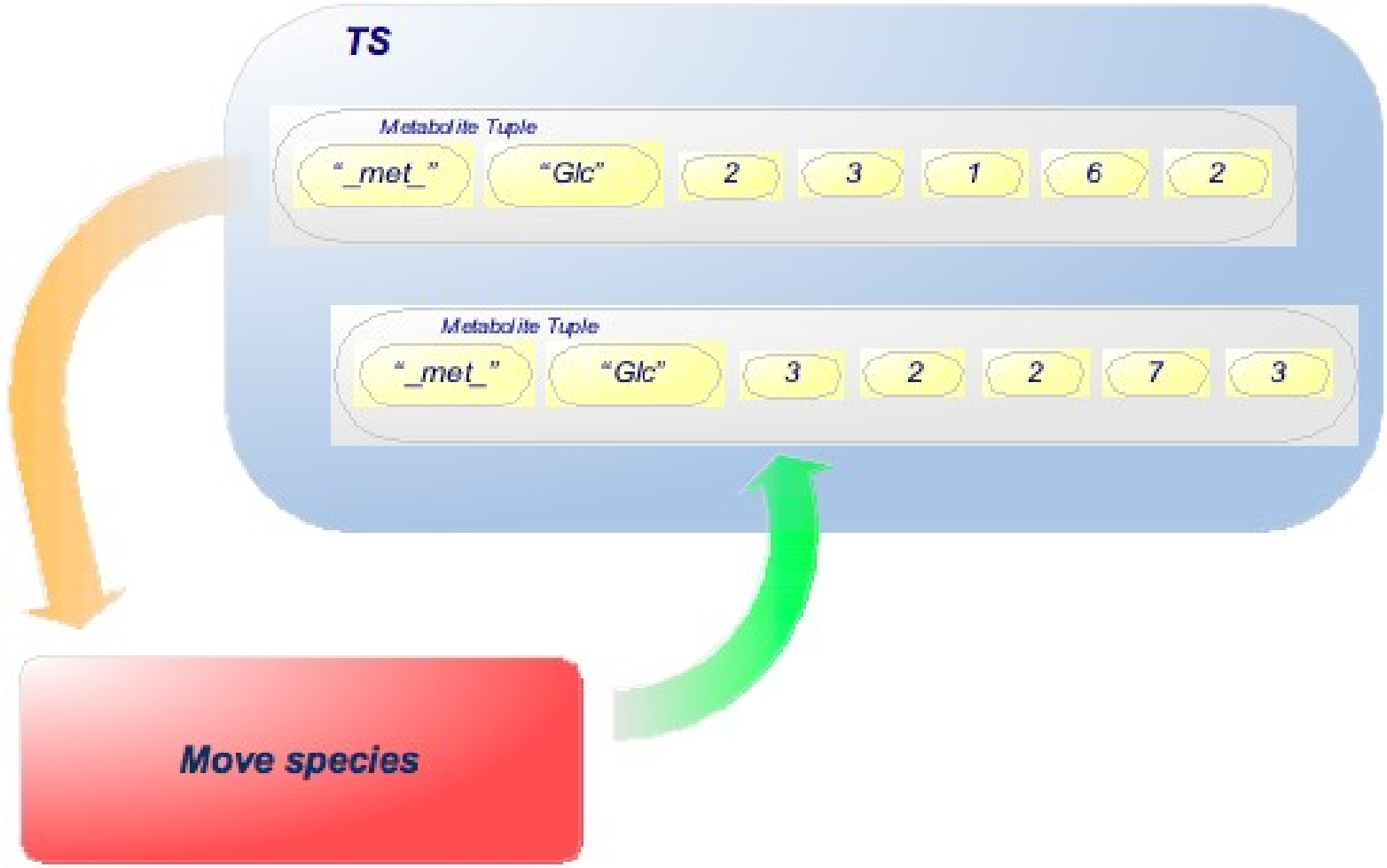
- out(t) insert a tuple t into the Tuple space (non-blocking)
- in(t) find and remove a *matching* tuple from the tuple space; block until a matching tuple is found
- rd(t) like in(t) except that the tuple is not removed
- eval(t) add the active tuple t to the tuple space

The model is structurally recursive: a tuple space can be one field of a tuple.

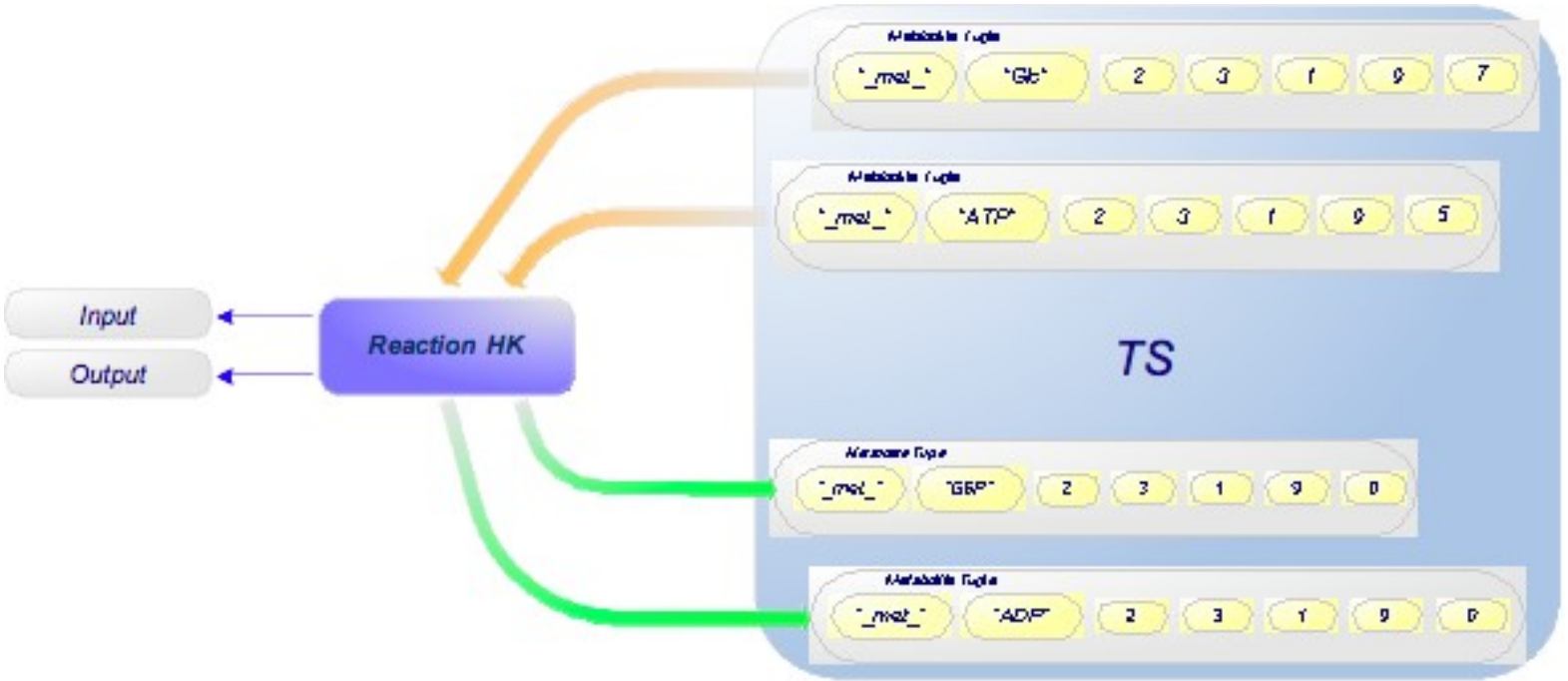
# Kalliope-Klaim tuples



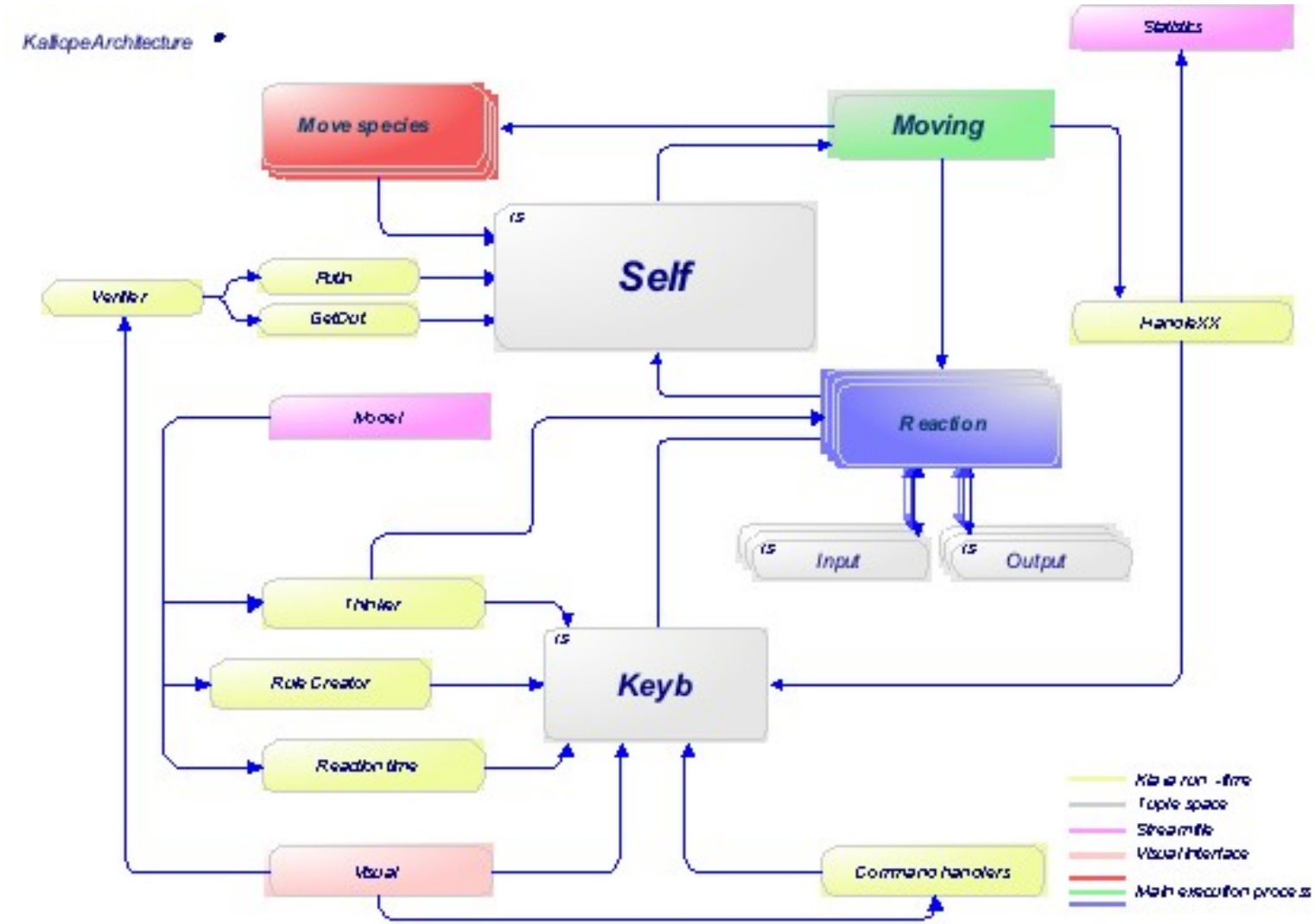
# Kalliope movement



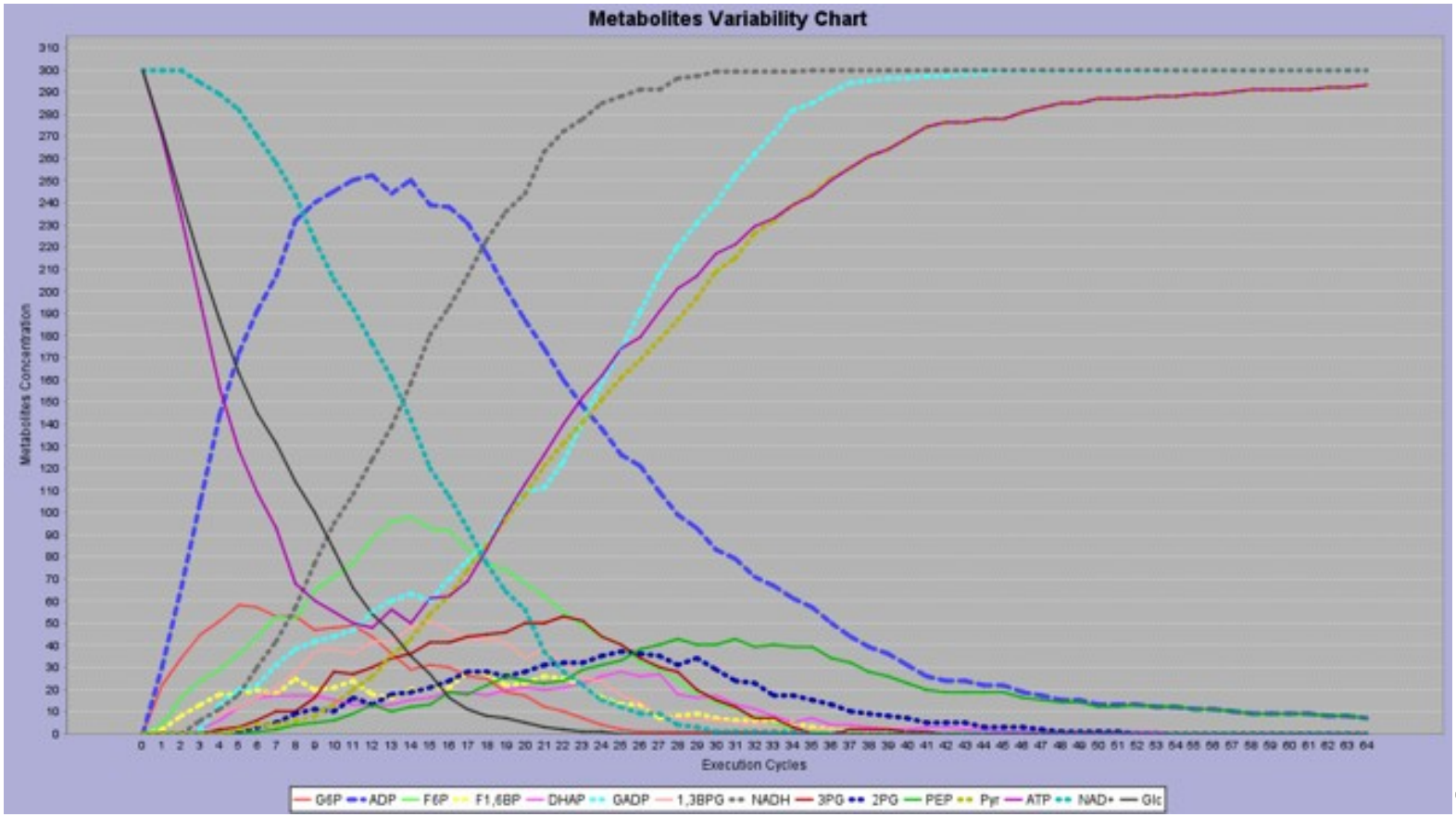
# Kalliope reactions



# Kalliope simualtor architecture



# Kalliope simulation results




Contact us for the Kalliope sw: emanuela.merelli, nicola.cannata, luca.tesei@unicam.it

**Coordinator**

Flavio Corradini

**Researchers**

Diletta Cacciagrano
Nicola Cannata
Rosario Culmone
Maria Rita Di Berardini
Emanuela Merelli
Andrea Polini
Alberto Polzonetti
Luca Tesei

**Phd Students**

Ezio Bartocci
Francesco De Angelis
Francesca Piersigilli
Barbara Re
Oliviero Riganelli
Leonardo Vito
*Roberta Alfieri*
*Federica Chiappori*

# Cosy Reserach Group