



Università di Camerino  
**UNICAM**

# BioAgent

An Intelligent Mobile Agents platform for BioData manipulation

Emanuela Merelli

Informatics and BioScience Group

Copenhagen July 2001



University of Camerino



# Outline

- Motivations
- Main choices in BioAgent
- System's architecture
- BioAgent model
- Preliminary results
- On-going work

# Motivations

## λ Electronic diagnostic tool

- Significant amount of data (ORFs, Experiments, ...) disseminated and duplicated in a myriad of different dbs and repositories
- A lot of dedicated tools with different access modes
- Several actors: researchers, physicians, lab-technicians ...
- ...

## λ Our 1<sup>st</sup> goals

- define a general Open-Source Intelligent Mobile Agent platform to support “biological data analysis”
- define a declarative language to specify BioAgents

# What is an agent?

An **intelligent mobile agent** is a program that can autonomously move between nodes by preserving the status and by managing information (i.e. knowledge) useful to perform its **mission**

Any agent is characterized by the following features

*Moving capability*

*Communication capability*

*Cloning capability*

*State-preserving capability*

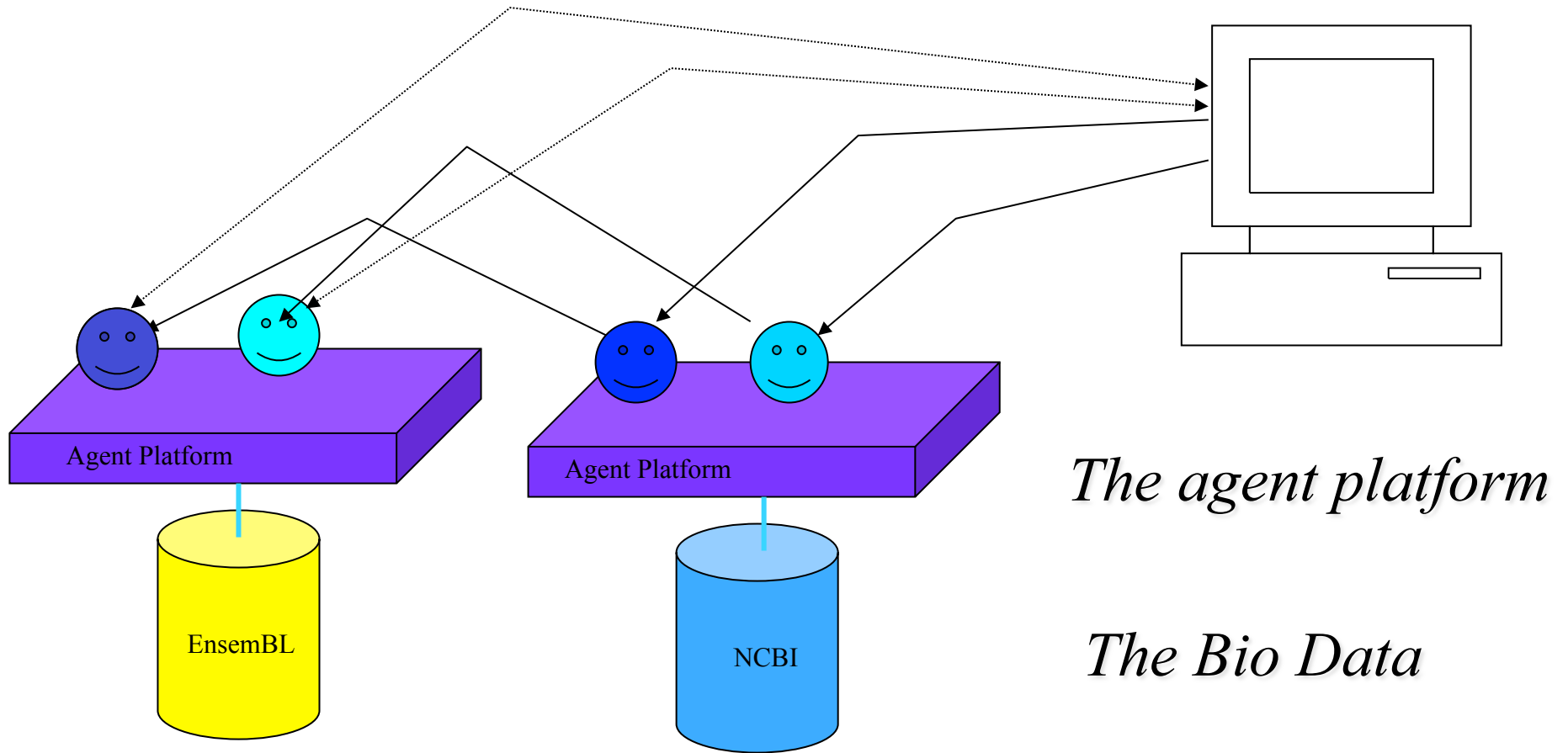
*Knowledge management capability*

a mobile agent can have a sort of “intelligence”. An agent can decide whether or not to undertake an action. It can represent and dynamically enrich its knowledge by using a descriptive language.

# How it works ...

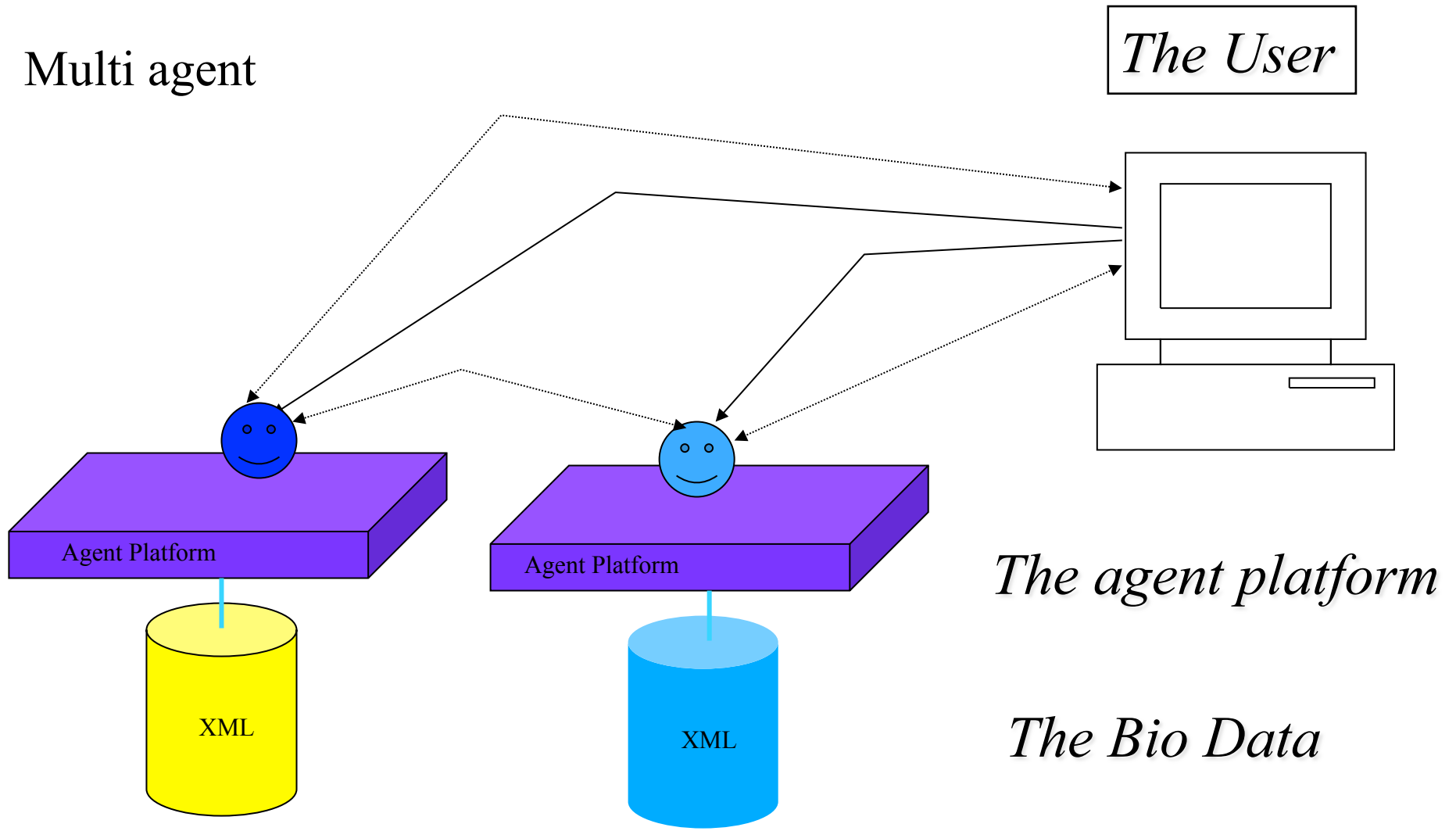
Single agent

*The user*



# How it works ...

Multi agent



# Benefits of mobile agents

- **Asynchronous** task execution
- Reduction of network traffic and client processing
- **Robustness**: reduction of dependence of network availability and client/server availability
- **Load-balance**
- Automation of distributed task processing
- Decentralized -local task processing
- **Flexibility**: on demand software distribution



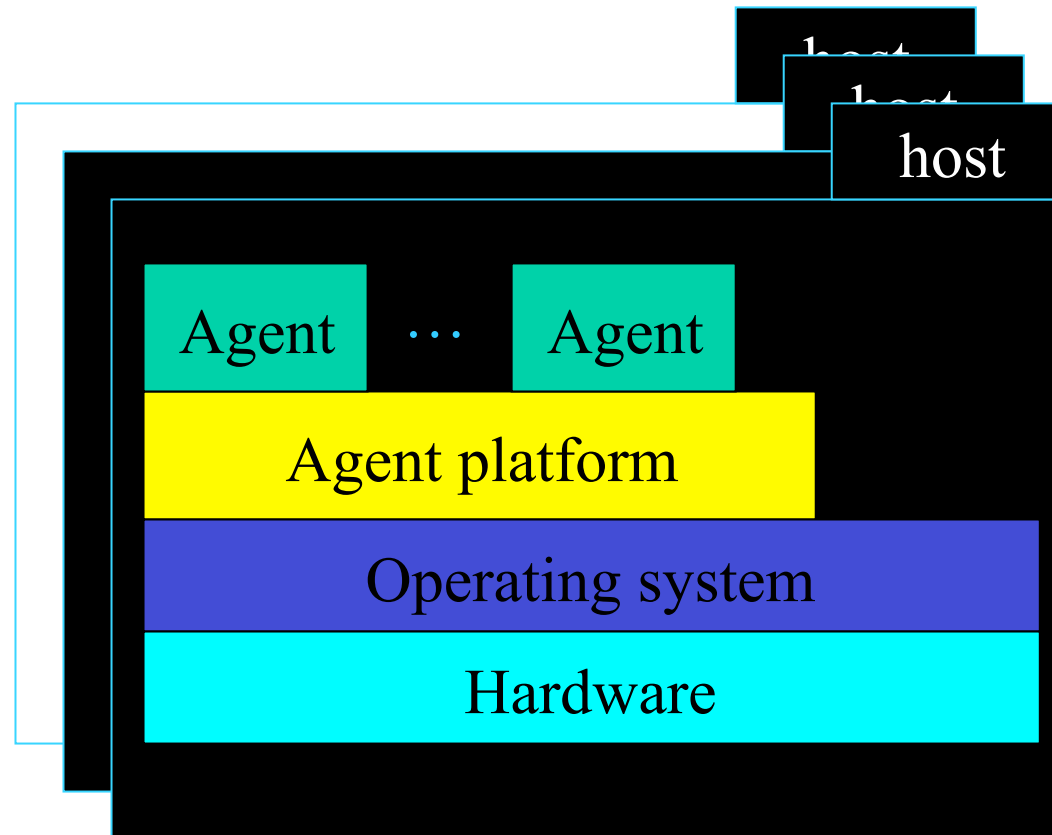
# Problems

- Security
  - λ Malicious host, malicious agent
- Task execution control
  - λ Location and status of agent may be unknown
- Fault management
  - λ Orphan detection, data inconsistency
- Performance
  - λ Interpreted code is slow
- Heterogeneity of hosts
- Code transfer overhead
- Access to existing services (e.g. CORBA)

# Requirements

- **Agent execution**
  - λ Agent creation, execution, deletion, execution control, task control
- **Agent mobility**
  - λ Remote execution, migration, naming, addressing, location, domains
- **Agent communication**
  - λ Inter-agent, user/agent, naming, addressing, location of peers
- **Agent system security**
  - λ System protection (agent = virus?), agent communication

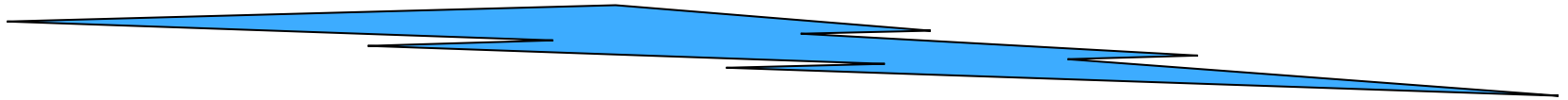
# System's architecture



# Mobile agent platforms

- [Aglets](http://www.tri.ibm.co.jp/aglets): [www.tri.ibm.co.jp/aglets](http://www.tri.ibm.co.jp/aglets)
  - [D'agents](http://agents.cs.dartmouth.edu): [agents.cs.dartmouth.edu](http://agents.cs.dartmouth.edu)
  - [Grasshoper](http://www.ikv.de): [www.ikv.de](http://www.ikv.de)
  - [Voyager](http://www.objectspace.com): [www.objectspace.com](http://www.objectspace.com)
  - [Macondo](http://www.cs.unibo.it/~ciancarini): [www.cs.unibo.it/~ciancarini](http://www.cs.unibo.it/~ciancarini)
  - [Others](#): Amates, Mole
- 
- Mobile agent platforms are implemented in Java

# Main choices for BioAgent



# Choice 1: all XML

- It's a W3C standard for data transfer
- Many tools are or will be available
- It's easy to use
- ...
  
- Bio data DTDs are under definition

# Choice 2: all declarative

- Fast deployment
- Easy administration
  - $\lambda$  e.g., dynamic change of rules, experiment categories ...
- Automatic verification
  - $\lambda$  e.g., correctness of protocols
  - $\lambda$  e.g., fairness of an extraction system
- etc.

# Choice 3: all autonomous

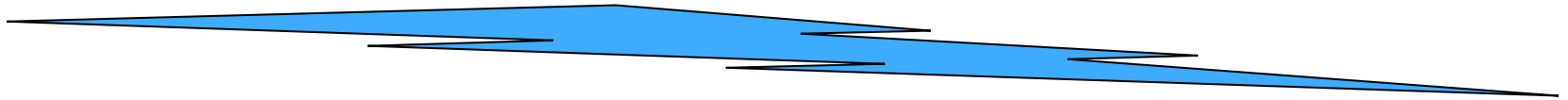
- Each node is an autonomous elaborative unit
- Each node can have its operating system
- Each node can have a DBMS and/or XML repository



# Choice 4: all Java

- Each node must have a JVM
  - λ Hardware independent (runs on Windows, Unix, MacOS, PDA, ...)
  - λ Object oriented
  - λ Ubiquitous (Java does everything, everyone does Java)
  - λ Network centric
    - Code mobility (applets, servlets, class loading, serialisability)
    - Network communication (TCP/UDP sockets)
    - Security (secure sockets, secure manager)

# BioAgent model



# BioAgent model

- **Basic features**

- $\lambda$  communication, ability to exchange information
- $\lambda$  memory, ability to freeze its status before to move with human customer or communicate
- $\lambda$  cloning, ability to instantiate a copy with others agents
- $\lambda$  moving ability to move the code from one site to another
- $\lambda$  knowledge mng ability to believe, decide and enrich

- **Application features**

- $\lambda$  Mission
- $\lambda$  Working Tools
- $\lambda$  Knowledge (domain)

# Who will use and specify an agent?

- bioinformatics,
- biochemical researchers,
- physicians,
- lab-technicians,
- biologists
- ...

Are they able to program in Java?  
Must they learn it?

# BioAgent

Intelligent Mobile Agent platform for Biology

**BioAgent** is an abstract machine able to interpreter the application agent instances specified and activated by using a declarative language

## Features

- High accessibility
- High modularity
- No propriety protocols
- Secure
- Simplicity
- Portability

## Modules

- Interface
- Space
- Core
- Communication
- Security

# BioAgent\_L

Biological Agent - Language

BioAgent\_L is a declarative language, “easy to use”, for specifying an intelligent mobile agent

DDL + DML

Example: how to specify a new environment

**let** <envname>

**be** *VirtualEnv*

**with** *Instruments, Events, Objects, Languages ...*

• • •

Example: how to specify a new agent

**create** <agent> **like** <tipo> **in** <environment>

**with**

**missions** (m1,m2 ...mn)

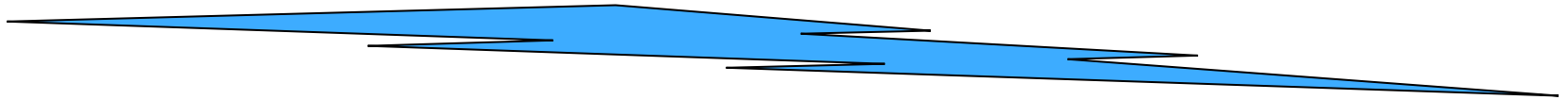
**reactions** (r1,r2, ... rn)

**instruments** (s1,s2, ... sn)

**language** (l1,l2, ... ln)

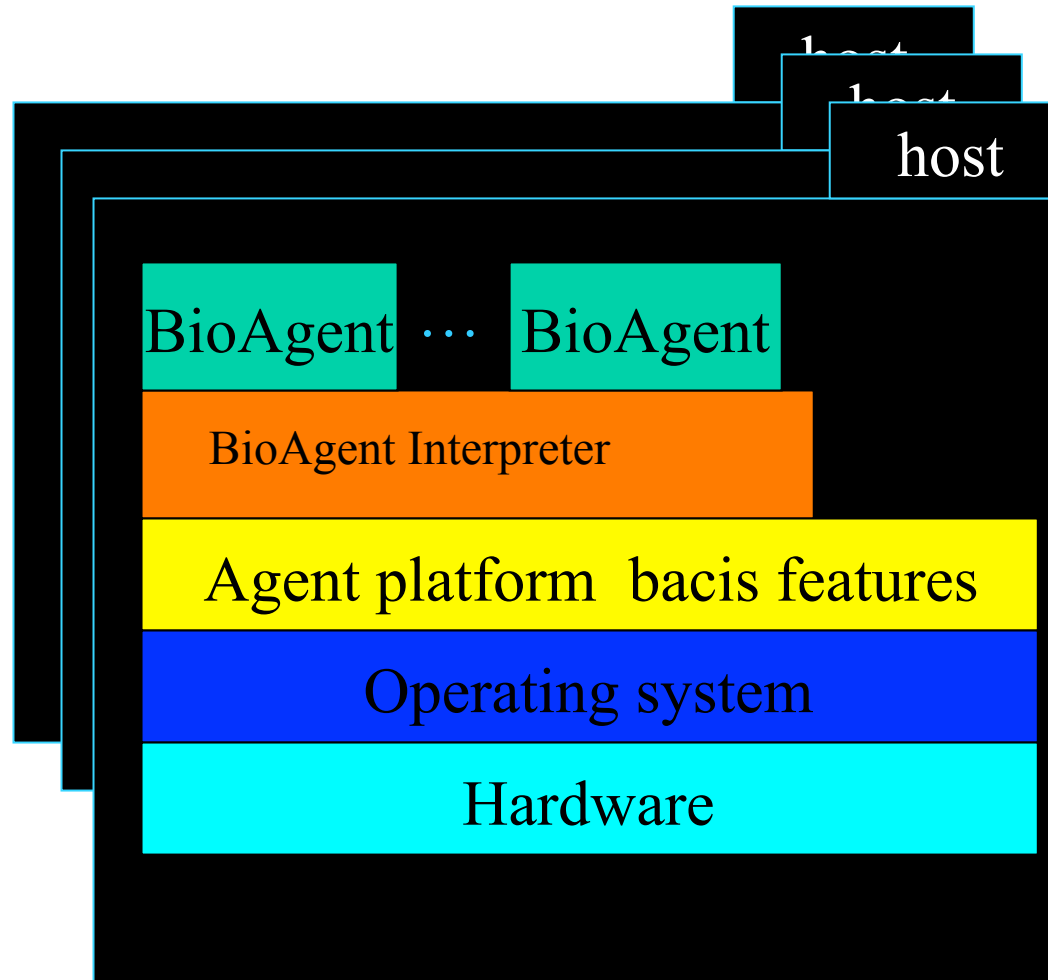
**agent-status** (o1,o2, ... on)

# System's architecture

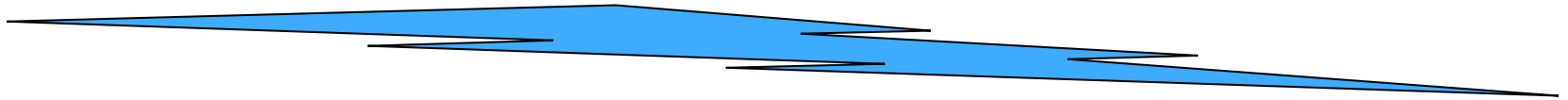




# BioAgent's Architecture



# Computational Analysis of Biological Data




# Agent's Missions

- Bio data search
- Bio data integration
- Bio data clustering
- Bio data extraction
- Bio pattern recognition
- Bio knowledge discovering
- Bio data prediction
- ...

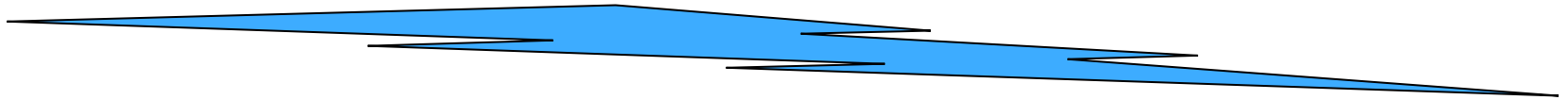
# Agent's Working tools

- Optimization
  - λ Combinatorial algorithms
  - λ Heuristics algorithms
  - λ CLP: Constraints Logic Programming
  - λ ...
- Classification
  - λ Neural nets
  - λ Kohonen self-organizing map
  - λ ...
- Knowledge Discovering
  - λ Data mining
  - λ Multidimensional analysis
  - λ ...

# Agent's Knowledge

- Basic knowledge (well-established)  
λ Dtd, DAML+OIL
- Extended knowledge  
λ *new knowledge*  

- Local knowledge  
λ any, not in XML format

# Application: analysis of gene expression

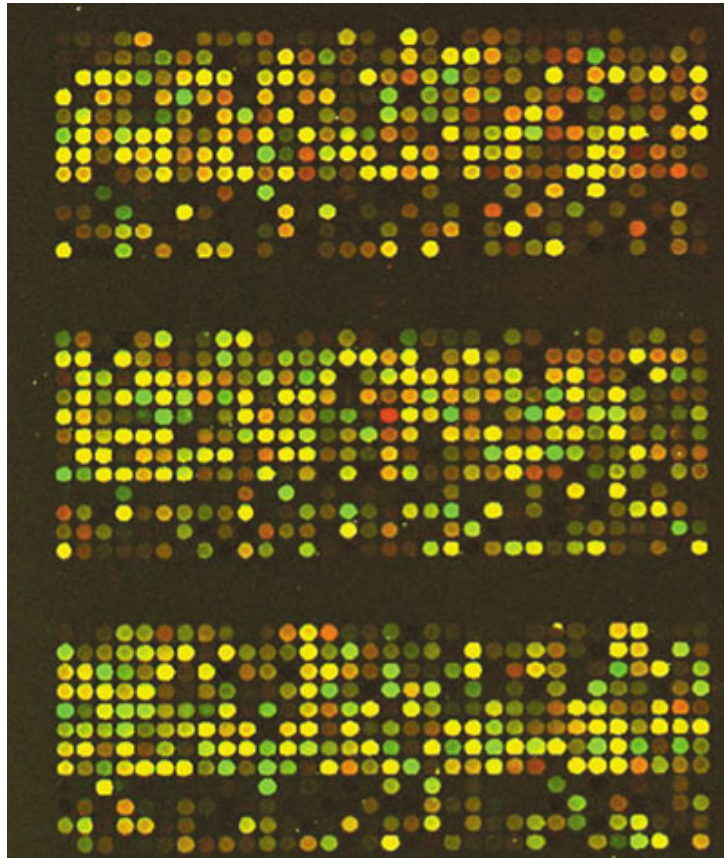


# Microarray experiments

“...normally an experiment should include a set of hybridisations which are inter-related and performed in a limited period of time.”

*MIAME (Minimal Information About Microarray Experiments) document by  
MGED  
(Microarray Gene Expression Database group, UK)*

# Hybridisation



- Each hybridisation is constituted by a collection of experimental data (spots) usually one spot for each ORF (Open Reading Frame).
- The intensity of each spot quantifies the expression of the related ORF under the chosen experimental conditions



# MicroArray Markup Languages

- MAML proposed by the European Molecular Biology Lab (EMBL) and the European Bioinformatic Institute (EBI) and recently submitted to OMG (Object Management Group)
- GEML (proposed by a public-private community, the GEML community [21]).
- BSML (Bioinformatic Sequence Markup Language) proposed by Visualgenomic, Inc. USA [20].

# MAML DTD structure

1. Experimental design: the set of the hybridization experiments as a whole;
2. Array design: each used array and each element (spot) on the array;
3. Samples: used samples, the extract preparation and labeling;
4. Hybridizations: procedures and parameters;
5. Measurements: **images**, quantitation, specifications;
6. Controls: types, values, specifications.

# Major goals pursued during the analysis of hybridization data

- data mining
- model-based/model-free
- functional classification
- clustering

# Clustering

- Clustering using experiments euc. distance  
 $\lambda$  (ORF “guilty-by-association”)  
 $\lambda$  Present use: discover gene function
- Clustering using ORFs euc. distance  
 $\lambda$  (transcriptional fingerprint)  
 $\lambda$  Future use: diagnostic tool

# BioAgent

**Data:** Biological data related to an “organism”

$\lambda$  set of inter-related hybridisations

**Mission:** analysis of hybridisation data

$\lambda$  Clustering by experiments

$\lambda$  Clustering by ORF

**Tool:**

$\lambda$  Kohonen self-organizing map

Basic Knowledge: MAML DTD

*Extended Knowledge:* Kohonen map

21-Jun-01

E. Merelli

# Kohonen Algorithm

1. We define with  $w_{ij}(t)$  the weights between the input neuron  $i$ th and the  $j$ th neuron in the map at time  $t$ . The weights initial values are random assigned in the  $[0,1]$  range.

2. Given an input  $x_0(t), x_1(t), ..x_n(t)$ , where  $x_i(t)$  is the  $i$ th input

3. Calculate the distance  $d_j$  between input  $i$  and each output neuron  $j$

$$d_j^2 = \sum (x_i(t) - w_{ij}(t))^2$$

4. Select neuron with minimum distance,  $j^*$

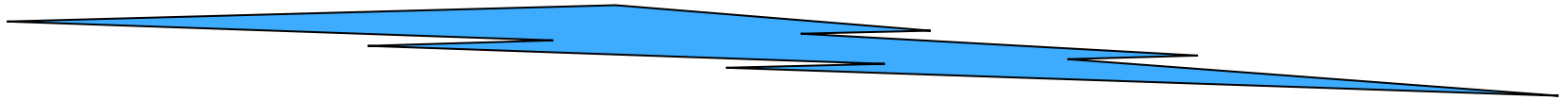
5. Modify the weights of the input neuron  $i$  and  $j^*$  and its neighbours  $N_i(j^*)$

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t)) \text{ for each } j \text{ in } N_i(j^*) \text{ and } 0 \leq i \leq n$$

$\eta(t)$  is a gain function  $0 \leq \eta(t) \leq 1$

6. Cycle from step 2

# Preliminary results



## Database source (ExpressDB Aach et al. 2000)

- 147 distinct hybridisation experiments
- 6053 ORFs

## Agent's Platform

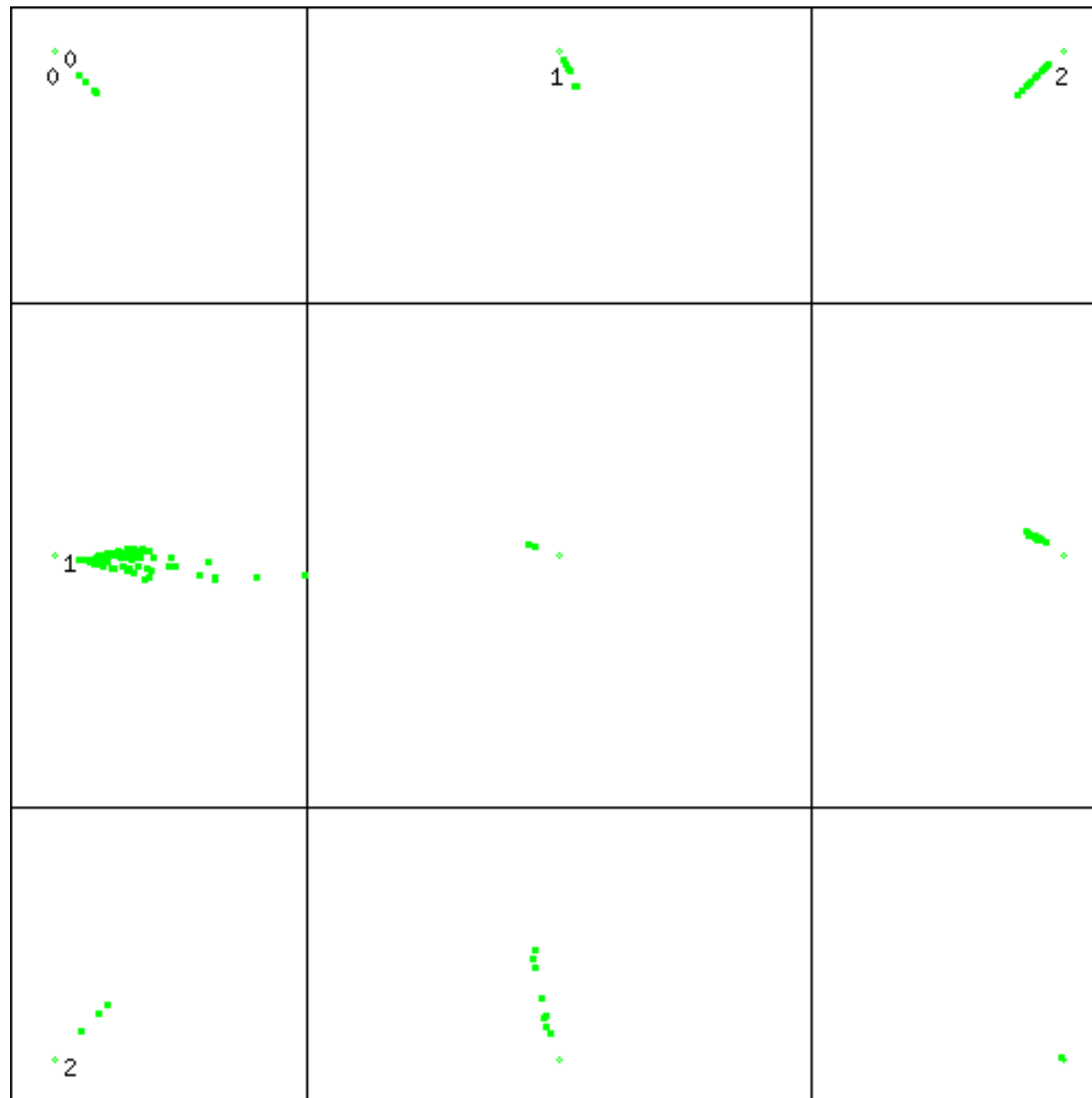
- *Macondo* [Ciancarini97]
- *MJada* for agents coordination and synchronization



# Kohonen algorithm

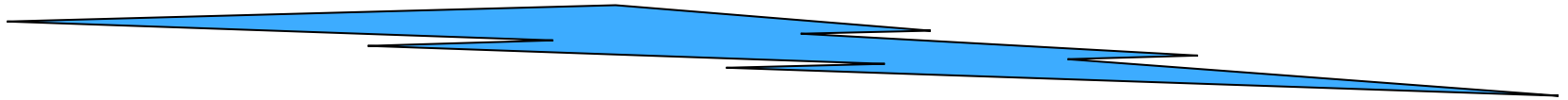
- Completely implemented in C, next version will be in Java
- 9 clusters
- The algorithm converges in 10000 cycles

# The output



10000

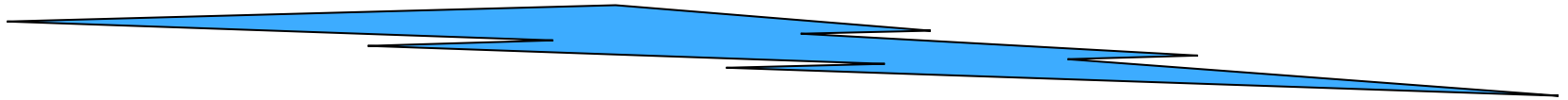
# Simple Demo



# Demo for single agent platform

- We simulate four different sites by four DOS windows. Three will be the place with the data repository and the fourth will be the starting and ending point for an agent
- Open 4 DOS windows
- For each window, in `C:\macondo` run `setenv`
- I window: in `C:\macondo\esempi\place` run  
`java exResourcePlace1`
- II window: in `C:\macondo\esempi\place` run  
`java exResourcePlace2`
- III window: in `C:\macondo\esempi\place` run  
`java exResourcePlace3`
- IV window: in `C:\macondo\esempi\place` run  
`java RunAgent localhost`
- The result is in `C:\macondo\esempio\risultato.res`
- PROTOTYPE

On-going work



# On going work

- 1st Phase
  - λ Analysis and design of the BioAgent system
  - λ Mobile agent basic implementation
  - λ 1st prototype fo Microarray clustering
  - λ BioAgent to interface EnsEMBL
- 2nd Phase
  - λ Analysis and design of the knowledge system
  - λ Definition of BioAgent declarative language
  - λ Implementation of BioAgent\_L interpreter
- 3<sup>rd</sup> Phase
  - λ Validate the system
  - λ Genaralize the platform so to allow the definition of personal “views” by BioAgent

# Informatics and BioScience group at Camerino University

Intelligent

Mobile

Agent platform

for

Biodata analysis

Emanuela Merelli

Rosario Culmone

Leonardo Mariani

Computer Scientists

Lorenzo Moglie,

Diego Bonura,

Gloria Rossi

Students in Computer Science

Mauro Angeletti

Biochemistrist

Cristina Marchini

Biologist