

Oujda
21 October 2019

MAS Course 05

Yves Demazeau
Yves.Demazeau@imag.fr

CONTENTS

APPLICATION : ATRONS & PLAYWARE

DEVELOPMENT

AGENT BASED SIMULATION

ORIENTATION

DEPLOYMENT

APPLICATION : MULTIPLE DRONES

COMPLEMENTARY REFERENCES

ATRONS

ATRONS Project (with USD - Henrik Lund)

High level Programming language for reconfigurable modular robotics

VOWELS approach

- ATRONS as **A**, evolving in a 3D environment **E**
- **I** given IR sensors and physical grippers
- **O** as a physical organization, a problem to solve
- or a function to emerge

Applications

- Technological : to demonstrate modular robotics
- Scientific : to support **emergence engineering**

ATRONS Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 5

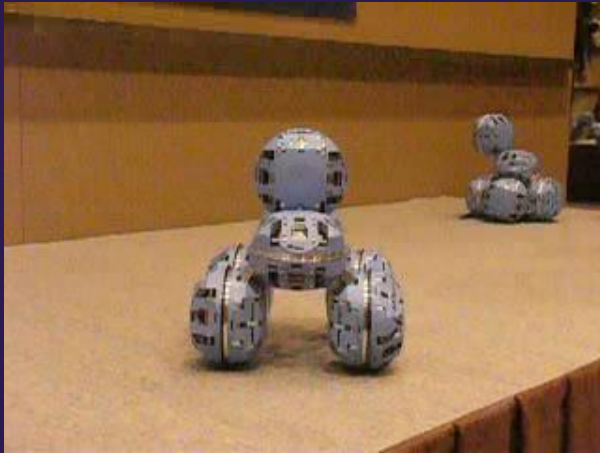
ATRONS Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 6

ATRONS Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 7

ATRONS Project (with USD – Ulrik Schultz)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 8

KILOBOTS

KILOBOTS

<https://www.youtube.com/watch?v=G1t4M2XnlhI>

PLAYWARE

PLAYWARE Project (with USD - Henrik Lund)

An interactive playground that recognizes and adapts to children

VOWELS approach

- Children as **U**, evolving in a 2D Environment **E**
- **A** as the agents situated in the playware tiles
- **I** given pressure sensor and visual actuators
- **O** as a dynamic structure arising during time

Applications

- Societal : to train children, to support healthcare
- Scientific : to introduce **U**ser-centered MAS

F. Hammer, A. Derakhshan, Y. Demazeau & H. Lund, "A Multi-Agent Approach to Social Human Behaviour in Children's Play", 6th Int. Conference on Agent Technology, IAT'06, IEEE/WIC/ACM, pp. 403-406, Hong-Kong, 2006.

PLAYWARE Project (with USD - Henrik Lund)



PLAYWARE Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 13

PLAYWARE Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 14

PLAYWARE Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAIZEAU - 15

PLAYWARE Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAIZEAU - 16

PLAYWARE Project (with USD - Henrik Lund)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAIZEAU - 17

PLAYWARE Project (with USD - Henrik Lund)

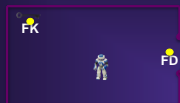


CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAIZEAU - 18

DEVELOPMENT

Reactive AI / Cognitive AI [Camps 10]



OD : door opened
CD : door closed
LD : door locked

FD position : in front of the door
FK position : in front of the key
K : Nao has the key
¬K : Nao does not have the key

Goal → Nao, a robot, has to exit the room

```

if FD  $\exists$  OD then exit
if FD  $\exists$  CD then open_door
if FD  $\exists$  LD  $\exists$  K then unlock_door
if FD  $\exists$  LD then random_walk
if FK  $\exists$  K then random_walk
if FK  $\exists$  ¬K then take_key
    
```

The transition function T is an unordered set of rules of type "if *condition* then action" that is executed in an infinite loop and in a nondeterministic way

Reactive Nao

```

goto FD
if OD then
  exit;
else
  if CD then
    open_door; exit;
  else
    if LD then
      unlock_door; open_door; exit;
    else
      goto FK;
      take_key;
      unlock_door; open_door; exit;
    endif
  endif
endif
    
```

The agent possesses a plan P to exit; it executes it sequentially and deterministically whatever its starting position and whatever the initial state of the world

Cognitive Nao

Evolution of Programming Paradigms

1950's

- Machine and assembly language

1960's

- Procedural programming

1970's

- Structured programming

1980's

- Object-Based programming, Declarative programming

1990's

- Frameworks, design patterns, scenarios, and protocols

2000's

- Agents... Multi-Agent Systems...

...

Features of Languages and Paradigms

Concept	Proc. L.	Object L.	Agent L.
abstraction	type	class	society
block	data	object	agent
model	procedure call	method message	perceive reason / act
paradigm	tree of procedures	interaction patterns	cooperative interaction
architecture	functional decomposition	inheritance polymorphism	managers assistants, peers
modes of	coding	designing and using	enabling and enacting
terminology	implement	engineer	activate

Evolution of Programming Paradigms

1950's

- ▣ Machine and assembly language

1960's

- ▣ Procedural programming

1970's

- ▣ Structured programming

1980's

- ▣ Object-Based programming, Declarative programming

1990's

- ▣ Frameworks, design patterns, scenarios, and protocols

2000's

- ▣ Agents... Multi-Agent Systems...

2010's

- ▣ Machine Learning Software / Tools...

Evolution of Programming Paradigms

1950's

- ▣ Machine and assembly language

1960's

- ▣ Procedural programming

1970's

- ▣ Structured programming

1980's

- ▣ Object-Based programming, Declarative programming

1990's

- ▣ Frameworks, design patterns, scenarios, and protocols

2000's

- ▣ Agents... Multi-Agent Systems...

2010's

- ▣ Machine Learning Software / Tools...

Evolution of Programming Paradigms

1950's

- Machine and assembly language

1960's

- Procedural programming

1970's

- Structured programming

1980's

- Object-Based programming, Declarative programming

1990's

- Frameworks, design patterns, scenarios, and protocols

2000's

- Agents... Multi-Agent Systems...

2010's

- Machine Learning Software / Tools...

2020's

- ???

Agent Oriented Programming [Shoham 93]

A complete AOP system will include three primary components

- a **restricted formal language** with clear syntax and semantics for describing mental state: the mental state will be defined uniquely by several modalities, such as belief and commitment
- an **interpreted programming language** in which to define and program agents, with primitive commands such as REQUEST and INFORM: the semantics of the language will be required to be faithful to the semantics of the mental state
- an **"agentifier"**, converting neutral devices into programmable agents.

Shoham's notion of Agent

- An agent is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments
- These components are defined in a precise fashion, and stand in rough correspondence to their common sense counterparts
- It is up to the consumer of the theory to consider the application at hand, and judge whether the correspondence between theory and common sense is enough close that s/he will not be misled by allowing common sense intuition to guide reasoning with the formal construct.
- What entities can be viewed as having mental state ? Anything can be so described, although it is not always advantageous to do so.

Components of mental state

Components of mental state

- the future is determined by two factors : past history and current actions of agents.

The actions of an agent are determined by its decisions, or choices

Decisions are logically constrained, though not determined, by the agent's beliefs

- These beliefs refer to the state of the world, to the mental state of other agents, and to the capabilities of this and other agents

Basics : beliefs, capabilities, obligations (decision is simply an obligation to oneself)

A language for belief, obligation, and capabilities

time / action	holding(robot,cup) ^t
belief	$B_a^t \varphi$ φ a (recursive) sentence
obligation	$OBL_{a,b}^t \varphi$
decision	$\equiv_{\text{def}} OBL_{a,a}^t \varphi$
capability	$CAN_a^t \varphi$ $ABLE_a^t \varphi =_{\text{def}} CAN^{\text{time}(\varphi)}_a \varphi$
time(φ) the outermost time occurring on	

Properties of the various components

internal consistency	$\forall a, t \{ \varphi : B_a^t \varphi \}$ is consistent $\forall a, t \{ \varphi : OBL_{a,b}^t \varphi \}$ is consistent
good faith	$\forall t, a, b, \varphi \ OBL_{a,b}^t \varphi \supset B_a^t ((ABLE_a \varphi) \wedge \varphi)$
introspection	$\forall t, a, b, \varphi \ OBL_{a,b}^t \varphi \equiv B_a^t OBL_{a,b}^t \varphi$ $\forall t, a, b, \varphi \neg OBL_{a,b}^t \varphi \equiv B_a^t \neg OBL_{a,b}^t \varphi$
persistence	so are mental state, obligations
capability	capabilities do not fluctuate widely

Agent0 : basic loop

The role of agent programs is to control the evolution of an agent's mental state

- actions occur as a side-effect of the agent's being committed to an action whose time has come

Each agent iterates the following two steps at regular intervals

- read current messages, and update your mental state, including your beliefs and commitments (the agent program is crucial for this update)
- execute the commitments for the current time, possibly resulting in further belief change (this phase is independent of the agent's program)

Interaction Oriented Programming [Huhns 96 01]

Motivations

- errors will always be in complex systems;
- Error-free code can be a disadvantage;
- Where systems interact with the real world, there is a power that can be exploited

Example : children forming a circle

- conventional approach: create a C++ class for each type of object, write a control program that uses trigonometry to compute the location of each object
- interaction-oriented approach: children approach is robust due to local intelligence and autonomy, write the program based on objects having attitudes, goals, agent models

IOP : Active modules, declarative specification, modules that volunteer, modules hold belief about the world, especially about themselves and others

Organization Oriented Programming [Lemaitre 98]

The most well-known effort towards MAOP is AOP [Shoham 93] ... IOP [Huhns 97] is an alternative...

OOP is another one [Lemaitre 98] ...

EOP does not actually exist as a trend but looks like Artificial Life.

These approach respectively focus on Agents, on Interactions, on Organizations, on Environments, as being the respective basic bricks at the disposal of the designer / MAS / user...

Multi-Agent Oriented Programming

Not Object-Oriented Programming

- S = Objects + Message passing

Not Logic nor Expert Systems Programming

- S = Knowledge + Inference Mechanism

Not Ontology-Oriented Programming

- S = Knowledge + Problem Solving Methods

But Agent-Oriented Programming

- S = BDI Agents + KQML (Interactions)

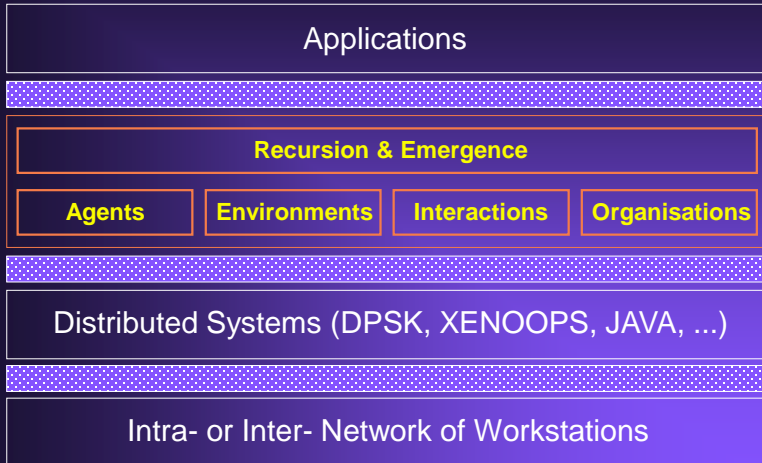
But (((A + I) + O) + E)-Oriented Programming

- S = ((A + I) + O) + E

But VOWELS Programming

- S = [A*; E*; I*; O*] + (Recursion & Emergence) Mechanism

Programming with the VOWELS approach



VOWELS Oriented Programming [Demazeau 97]

I defend an instance of Multi-Agent Oriented Programming, VOWELS, which consists :

- 1/ To express the problem independently of the domain
- 2/ To "vowellify" the problem in terms of A E I O, ...
- 3/ To choose understood frames of A, E, I, O, dynamics, and recursion
- 4/ To leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... To be deployed as self on a distributed settling...
- 6/ ... To be settled and used interactively

VOWELS Oriented Programming [Demazeau 97]

I defend an instance of Multi-Agent Oriented Programming, VOWELS, which consists :

- 1/ To express the problem independently of the domain
- 2/ To "vowellify" the problem in terms of A E I O, ...
- 3/ To choose understood frames of A, E, I, O, dynamics, and recursion
- 4/ To leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... To be deployed as self on a distributed settling...
- 6/ ... To be settled and used interactively

The **Programming** Principle

MAS = [A*; E*; I*; O*]
+ (Recursion & Emergence) Mechanism

VOWELS Oriented Programming [Demazeau 97]

I defend an instance of Multi-Agent Oriented Programming, VOWELS, which consists :

- 1/ To express the problem independently of the domain
- 2/ To "vowellify" the problem in terms of A E I O, ...
- 3/ To choose understood frames of A, E, I, O, dynamics, and recursion
- 4/ To leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... To be deployed as self on a distributed settling...
- 6/ ... To be settled and used interactively

The **Programming** Principle

MAS = [A*; E*; I*; O*]
+ (Recursion & Emergence) Mechanism

Multi-Agent System, Emergence, Recursion

The **Declarative** Principle

- $MAS = A + E + I + O$

The **Functional** Principle

- $Function(MAS) = \sum Function(entities) + Emergence Function$

The **Recursive** Principle

- $entity = basic\ entity \mid MAS$

VOWELS Oriented Programming [Demazeau 97]

The **Declarative** Principle

- $MAS = A + E + I + O$

The **Functional** Principle

- $Function(MAS) = \sum Function(entities) + Emergence Function$

The **Recursive** Principle

- $entity = basic\ entity \mid MAS$

The **Programming** Principle

$$MAS = [A^*; E^*; I^*; O^*] \\ + (Recursion \ \& \ Emergence) \ Mechanism$$

VOWELS Oriented Programming [Demazeau 97]

The **Declarative** Principle

□ $MAS = A + E + I + O$

The **Functional** Principle

□ $\text{Function}(MAS) = \sum \text{Function}(\text{entities}) + \text{Emergence Function}$

The **Recursive** Principle

□ $\text{entity} = \text{basic entity} \mid MAS$

The **Programming** Principle

$MAS = [A^*; E^*; I^*; O^*]$
+ (Recursion & Emergence) Mechanism

Y. Demazeau, "Steps towards Multi-Agent Oriented Programming" (slides Workshop), 1st International Workshop on Multi-Agent Systems, IWMAS '97, Boston, October 1997.

MAOP and Domain Orientation

The purpose of the domain drives the design

$((A + E) + I) + O$	Robotics Science
$((A + I) + O) + E$	Social Science
$((E + A) + (I + O))$	Life Science
$((I + O) + A) + E$	Military Science
$((O + I) + E) + A$	Economic Science

MAOP and Domain Orientation

The purpose of the domain drives the design

$((A + E) + I) + O$	Robotics Science
$((A + I) + O) + E$	Social Science
$((E + A) + (I + O))$	Life Science
$((I + O) + A) + E$	Military Science
$((O + I) + E) + A$	Economic Science

MAS are not always A-centered ! See praxis !

The purpose of the Domain

In all this, where is the User ? Is it at her right place ?

$((A + E) + I) + O$	Robotics Science
$((A + I) + O) + E$	Social Science
$((E + A) + (I + O))$	Life Science
$((I + O) + A) + E$	Military Science
$((O + I) + E) + A$	Economic Science

The purpose of the User

In all this, where is the User ? Is it at her right place ?

$((((A + E) + I) + O) + U)$	Robotics Science
$((((A + I) + O) + E) + U)$	Social Science
$((((E + A) + (I + O)) + U)$	Life Science
$((((I + O) + A) + E) + U)$	Military Science
$((((O + I) + E) + A) + U)$	Economic Science

The traditional place of the « User » is at the « End- »

The purpose of the User

But, in every domain, is the User only a Consumer ?

$((((A + E) + I) + O) + U)$	Robotics Science
$((((A + I) + O) + E) + U)$	Social Science
$((((E + A) + (I + O)) + U)$	Life Science
$((((I + O) + A) + E) + U)$	Military Science
$((((O + I) + E) + A) + U)$	Economic Science

The purpose of the User

She should be at its right place ! At least with the A !

$((((U + A) + E) + I) + O)$	Robotics Science
$((((U + A) + I) + O) + E)$	Social Science
$((E + (U + A)) + (I + O))$	Life Science
$((I + O) + (U + A)) + E$	Military Science
$((O + I) + E) + (U + A)$	Economic Science

The purpose of the User

She should be at its right place ! At least with the A !

$((((U + A) + E) + I) + O)$	Robotics Science
$((((U + A) + I) + O) + E)$	Social Science
$((E + (U + A)) + (I + O))$	Life Science
$((I + O) + (U + A)) + E$	Military Science
$((O + I) + E) + (U + A)$	Economic Science

MAS are not always ((U + A)-centered ! See games !

Interactive Games

General

- ---> A to be replaced by $(U + A)$

Handling

- The goal is to master emergence, to optimize a cost
- The understanding of the whole system has to be easy
- The (E)nvironment has to be as realistic as possible
- ---> E to be replaced by $(U + E)$

Strategy

- The goal is to use and to plan the use of resources
- The visualization of the interactions is highly desirable
- The key parameters of the game are (I)nteractions
- ---> I to be replaced by $(U + I)$

Role

- The goal is to increase the competences of the User
- Competences of the characters have to be visualized
- The (O)rganization constitutes the entry of the game
- ----> O to be replaced by $(U + O)$

The purpose of the User and of the Domain

From U as consumer...

$((((A + E) + I) + O) + U)$	Robotics Science
$((((O + I) + E) + (U + A))$	Economic Science

To U as a partner...

$((((I + O) + (U + A)) + E)$	Military Science
$((((E + (U + A)) + (I + O))$	Life Science
$(((((U + A) + I) + O) + E)$	Social Science

Towards U as a creator...

$(((((U + A) + E) + I) + O)$	Robotics Science
$(((((U + (O + I)) + E) + A)$	Economic Science

VOWELS A E I O Decomposition

Agents

- internal architectures of the system processing entities

Environment

- domain-dependent elements for structuring external interactions between entities

Interactions

- elements for structuring internal interactions between entities

Organisations

- elements for structuring sets of entities within the MAS

VOWELS A E I O U Decomposition [Demazeau 03]

Agents

- internal architectures of the system processing entities

Environment

- domain-dependent elements for structuring external interactions between entities

Interactions

- elements for structuring internal interactions between entities

Organisations

- elements for structuring sets of entities within the MAS

Users

- internal architectures of the end-user processing entities

VOWELS A E I O U Decomposition [Demazeau 03]

Agents

- internal architectures of the system processing entities

Environment

- domain-dependent elements for structuring external interactions between entities

Interactions

- elements for structuring internal interactions between entities

Organisations

- elements for structuring sets of entities within the MAS

Users

- internal architectures of the end-user processing entities

VOWELS Oriented Programming

The **Declarative** Principle

- $MAS = A + E + I + O$

The **Functional** Principle

- $Function(MAS) = \sum Function(entities) + Emergence Function$

The **Recursive** Principle

- $entity = basic\ entity \mid MAS$

The **Programming** Principle

$$MAS = [A^*; E^*; I^*; O^*] + (Recursion \ \& \ Emergence)$$

VOWELS Oriented Programming [Demazeau 03]

The Declarative Principle

□ $MAS = A + E + I + O + U$

The Functional Principle

□ $Function(MAS) = \sum Function(entities) + Emergence\ Function$

The Recursive Principle

□ $entity = basic\ entity \mid MAS$

The Programming Principle

$MAS = [A^*; E^*; I^*; O^*; U^*] + (Recursion \ \& \ Emergence)$

VOWELS Oriented Programming [Demazeau 03]

The Declarative Principle

□ $MAS = A + E + I + O + U$

The Functional Principle

□ $Function(MAS) = \sum Function(entities) + Emergence\ Function$

The Recursive Principle

□ $entity = basic\ entity \mid MAS$

The Programming Principle

$MAS = [A^*; E^*; I^*; O^*; U^*] + (Recursion \ \& \ Emergence)$

VOWELS Oriented Programming

MAOP subsumes AOP, IOP, OOP-like OP...

We defend an instance of MAOP, the VOWELS framework in which :

- 1/ to express the problem to solve independently of the domain
- 2/ to "vowellify" the problem in terms of A E I O, ...
- 3/ to choose understood frames of A, E, I, O, dynamics, and recursion
- 4/ to leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... to be deployed as self on a distributed settling...
- 6/ ... to be settled and used interactively

VOWELS Oriented Programming [Demazeau 03]

MAOP subsumes AOP, IOP, OOP-like OP...

We defend an instance of MAOP, the VOWELS framework in which :

- 1/ to express the problem to solve independently of the domain
- 2/ to "vowellify" the problem in terms of A E I O **U**, ...
- 3/ to choose understood frames of A, E, I, O, **U**, dynamics, and recursion
- 4/ to leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... to be deployed as self on a distributed settling...
- 6/ ... to be settled and used interactively

VOWELS Oriented Programming [Demazeau 03]

MAOP subsumes AOP, IOP, OOP-like OP...

We defend an instance of MAOP, the VOWELS framework in which :

- 1/ to express the problem to solve independently of the domain
- 2/ to "vowellify" the problem in terms of A E I O U, ...
- 3/ to choose understood frames of A, E, I, O, U, dynamics, and recursion
- 4/ to leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... to be deployed as self on a distributed settling...
- 6/ ... to be settled and used interactively

Y. Demazeau, "Créativité Emergente Centrée Utilisateur" (keynote), 11èmes Journées Francophones sur les Systèmes Multi-Agents, pp. 31-36, Hermès, Hammamet, Novembre 2003.

Playing with VOWELS and Domain Orientation

PhD Boissier	$(A + I) + O$	ASIC
PhD Sichman	$A + O$	DEPNET
PhD Ferrand	$((A + I) + O) + E$	SANPA
PhD Baeijs	$((A + E) + I) + O$	SIGMA
PhD Van Aeken	$O + A$	SMAMS
PhD Ribeiro	$I + A$	DIM
PhD Ricordel	Development	VOLCANO
PhD Tavares	Planning	
PhD Deguet	Emergence	
PhD Piolle	$((U + A) + I) + O) + E$	PAW
PhD Joumaa	m Observation	MASPAJE
PhD Crepin	$((U + O) + I) + A) + E$	HIPPO
PhD Lacomme	Dynamics	
PhD Lamarche	M Observation	

AGENT BASED SIMULATION

Decentralized System Simulation

Developing a computational model for **simulating** the actions and interactions of a group of autonomous **individuals**

The individuals (**agents**) might represent Plants and animals in an ecosystem, Vehicles in traffic, People in crowds, Politicians, Autonomous characters in animation and games, etc.

Usually known as **Agent Based Simulation**

When the individuals represent **people** or **societies** then the term **Agent Based Social Simulation (ABSS)** is more often used

Agent Based Social Simulation

High number of complex entities (some agents may represent people) interacting in a non-trivial way

Influence of agents by agents is a major concern
Main feature is social interaction between agents

Recognises the cognitive limitations to rationality (bounded rationality)

Dynamically changing and heterogeneous environment

Aims to create emergent social behaviour / phenomena from the bottom-up

Distributed Cognition [Hutchins 95]

Incorporates aspects from sociology, cognitive science and psychology

Proposes that human knowledge and cognition are not confined to the individual, it is distributed by placing memories, facts, or knowledge on the objects, individuals, and tools in our environment

The system is a set of representations, model the interchange of information between them

The development of knowledge is due to human agents interacting dynamically with artefacts.

(Re)designing work practices. CSCW (Computer Supported Collaborative Work)

Activity Theory [Vygotsky 78]

The basic unit of analysis is human (work) activity. It is distributed among the members of the community

The activity is usually mediated by one or more tools or resources

Activities can be considered as having three hierarchical levels: **activity**, **action** and **operation**, which can be individual or cooperative [Kutti 96]

An activity may be achieved through various actions, an action may contribute to different activities, operations may contribute to several actions

Map activities, actions and operations, are easily mapped to agents behaviors

Situated Cognition [Brown 89]

Situated cognition argues that knowing is inseparable from doing

Stresses the importance of context (social, cultural and physical contexts). Idea: Cognition cannot be separated from the context

Argues against the storage and retrieval of conceptual knowledge but says that we only know if we take into account context

Once an intention (goal) is adopted, the agent's perception (attention) is attuned to the properties of the environment [Gibson 79]

Implies constant situation awareness

Social Intelligence [Thorndike 20]

"The ability to understand and for people to act wisely in human relations"

When dealing with knowledge of social situations the term **Social Cognition** is more often used

More concerned with how people process social information, especially its encoding, storage, retrieval, and application to social situations

Uses the idea of social schemas

To represent believable social interactions between agents we may need to model concepts such as: empathy, influence, and concern

DEPLOYMENT

Object-Based (Concurrent) Languages

Actors-like Languages

- ▣ Massive Parallelism
- ▣ Every interaction is made by passing of asynchronous buffered messages
- ▣ Use of the local continuity
- ▣ Object-oriented design
- ▣ Full distributed control

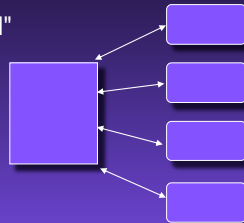
Actors [Hewitt]

- ▣ An actor is composed of acquaintances, scripts, ...
- ▣ A message is an actor containing method, continuation, and complaint actors
- ▣ A method is a set of actions : create a new actor, send a message, modify actor's behavior

(Distributed) Blackboard Systems

BB-Like frameworks

- ▣ Interactions through the "blackboard"
- ▣ Opportunistic activation of the knowledge sources
- ▣ Initially Centralized Control



DBB

- ▣ Blackboard
 - ▣ A global database representing state of the problem
 - ▣ Hierarchically organized into levels of granularity
- ▣ Knowledge Sources
 - ▣ Produce changes (hypotheses) onto given levels of BB
 - ▣ Contain particular subset of domain knowledge
- ▣ Centralized Control
 - ▣ Evaluates the status of the problem
 - ▣ Controls the knowledge sources

Characteristics of the Integrative Environments

Language for constructing agents

- Different agent architectures
- Language for knowledge representation
- Mechanisms for reasoning, deciding, controlling

Representing and dealing with the environment

- Representation of the environment and its evolution
- Implementing the perception of the environment
- Implementing the actions of the agents in the environment

Representing and dealing with other agents

- Primitives, protocols, message processing
- Implementing communication between agents

Development Interface

- Visualization, trace, inputs-outputs

Application Interface

Interface with Target (Distributed or not) System

Integrative Environments in the 90's

Cognitive MAS

- ABE [Erman]
- AOP [Shoham]
- MASK [Demazeau et al.]
- MECCA [Steiner]
- MICE [Durfee]
- PACT [Cutkosky]
- PHOENIX [Cohen]
- ...

Reactive MAS

- ECO [Drogoul]
- LYDIA [Connah]
- PACO [Demazeau et al.]
- RDL [Steels]
- ...

Palo Alto Collaborative Testbed

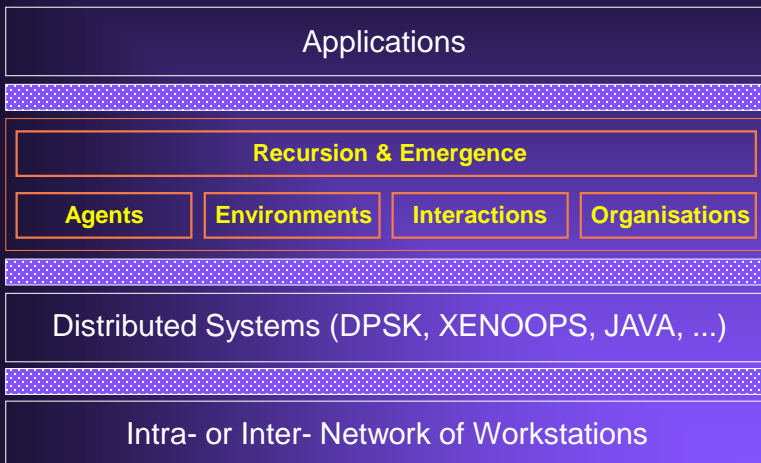
Computer-Aided Concurrent Engineering

- Stanford, Lockheed, Hewlett-Packard, Enterprise Integration Technologies
- Have integrated four pre-existing CE systems into common distributed framework supporting aspects of small robot

Uses quasi-standards

- Knowledge Interchange Format (KIF) [DARPA-92]
- Knowledge Query & Manipulation Language (KQML) [Finin-92]
- Product Data Exchange Specification (PDES)

Programming with the VOWELS approach



MAS Deployment in the 2000's

Academics

- Firefly (MIT then Microsoft)
- MadKit (LIRMM Montpellier - Ferber's group) (-> 2017)
- Simula (II Porto Alegre - Alvares's group)
- PRS (MIT) then dMARS (AAIL) then Jack (see below)
- AgentSpeak then Jack (see below)
- ...

Industrials

- Aglets (IBM) - freeware (Java) (-> 2004)
- Javabeans (Sun then Oracle) - freeware (Java)
- Jade (Telecom Italia) – freeware (Java, ACL) (-> 2011)
- Jack (Agent Oriented Software) (Java)
- Agentbuilder (Reticular then Acronymics) (Java, KQML)
- ZEUS (BT) - freeware product (Java, ACL)
- ...

Qualification criteria

Four qualitative properties for each stages:

- Completeness: quantity & quality
- Applicability: scope, restrictions
- Complexity: competence required, workload
- Reusability: reuse of previous work

16 criteria + availability & support

	Analysis	Design	Development	Deployment
Completeness				
Applicability				
Complexity				
Reusability				

Selected platforms

Platforms requirements :

- based on a strong academic model
- high quality software, well maintained
- cover as many aspects as possible of MAS
- cover the four methodological stages

Madkit, Jack, Zeus, AgentBuilder

- Evaluation as of first semester 2000
- Detailed presentation for MadKit (2006)
- Detailed presentation for Jack (2007)

P.-M. Ricordel & Y. Demazeau, "From Analysis to Deployment: A Multi-Agent Platform Survey", Engineering Societies in the Agents' World, ESAW'00, pp. 93-105, Berlin, 2000.

MadKit[®]

Developed by O. Gutknecht & J. Ferber, LIRMM

Based on the AALAADIN organizational model

Graphical multi-agent runtime engine

Good versatility

Light methodology, no BDI

	Analysis	Design	Development	Deployment
Completeness	none	Aalaadin	Pure Java	G-Box
Applicability	n / a	broad range	simple A	small large MAS
Complexity	n / a	intuitive	few code base	GUI
Reusability	n / a	design patterns	classes	dynamic reconf.

Jack™

Developed by Agent Oriented Software Pty.

Including the dMARS BDI model

Great versatility

Focus on the development stage

	Analysis	Design	Development	Deployment
Completeness	none	ident. of classes	extended Java	manual
Applicability	n / a	Jack BDI A	Any MAS	n / a
Complexity	n / a	Jack BDI A	Java & Logic P.	n / a
Reusability	n / a	difficult	classes	n / a

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 79

Zeus

Developed by British Telecom

All stages covered, from analysis to deployment

Methodological and Software tools

Limited to a single agent model

	Analysis	Design	Development	Deployment
Completeness	role modelling	finding solutions	5 activities	tools docs
Applicability	role o. MAS	task o. A	Zeus A model	debug visualis.
Complexity	UML	design skills	GUI tools	GUI
Reusability	role models	reusable formal.	partial	A reconf.

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 80

AgentBuilder®

Developed by Reticular Systems Inc.

Grounded on Agent0/Placa BDI architecture

Almost all stages covered

Complete graphical tools

Limited to a single agent model

	Analysis	Design	Development	Deployment
Completeness	ontology	A definition	behavioural rules	RT A engine
Applicability	universal	cognitive A	AgentBuild. BDI	small societies
Complexity	OO GUI	MAS design GUI	Logic P. GUI	GUI
Reusability	ontology	protocols	A	none

VOLCANO [Ricordel 01]

A multi-agent platform

- Based on the VOWELS method
- Full analysis-to-deployment chain
 - └ Problem/domain decomposition
 - └ AEIO modeling
 - └ Open library of models (simplicity, versatility, reusability)
 - └ Intelligent deployment tools

But

- Has never been further reused...
- Has still to be fully evaluated...
- Is looking for software engineers...
- Is maybe not really necessary...

MAS Deployment in the 2000's

Academics

- Firefly (MIT then Microsoft)
- MadKit (LIRMM Montpellier - Ferber's group) (-> 2017)
- Simula (II Porto Alegre - Alvares's group)
- PRS (MIT) then dMARS (AAIL) then Jack (see below)
- AgentSpeak then Jack (see below)
- Volcano (LEIBNIZ Grenoble - MAGMA group) [Ricordel 01]
- ...

Industrials

- Aglets (IBM) - freeware (Java) (-> 2004)
- Javabeans (Sun then Oracle) - freeware (Java)
- Jade (Telecom Italia) – freeware (Java, ACL) (-> 2011)
- Jack (Agent Oriented Software) (Java)
- Agentbuilder (Reticular then Acronymics) (Java, KQML)
- ZEUS (BT) - freeware product (Java, ACL)
- ...

MAS Deployment in the 10's

Academics

- MadKit (LIRMM Montpellier - Ferber's group) (-> 2017)
- AgentSpeak then JASON (PUCRS - Bordini's group) (Java)
- GOAL (VUA Amsterdam - Hindricks's group) (Prolog)
- SARL (UTBM Belfort - Galland's group) (Java)
- JaCaMo (ENSM Saint-Etienne - Boissier's group)
- ...

JaCaMo [Boissier 13]

- Jason Agent Oriented Programming
- Cartago Environment Oriented Programming
- Moïse Organisation Oriented Programming

MAS Deployment in the 10's

Academics

- MadKit (LIRMM Montpellier - Ferber's group) (-> 2017)
- AgentSpeak then JASON (PUCRS - Bordini's group) (Java)
- GOAL (VUA Amsterdam - Hindricks's group) (Prolog)
- SARL (UTBM Belfort - Galland's group) (Java)
- JaCaMo (ENSM Saint-Etienne - Boissier's group)
- ...

Industrials

- Aglets (IBM) - freeware (Java) (-> 2004)
- Javabeans (Sun then Oracle) - freeware (Java)
- Jade (Telecom Italia) - freeware (Java, ACL) (-> 2011)
- Jack (Agent Oriented Software) (Java)
- Anylogic (Anylogic) (Java)
- ...

ABS Tools in the 10's

NetLogo

- <https://ccl.northwestern.edu/netlogo/>
- Developed by Uri Wilensky, Northwestern Univ.

Swarm

- http://www.swarm.org/wiki/Swarm_main_page
- Developed by the Santa Fe Institute

RePAST

- <http://repast.sourceforge.net>
- Developed by David Sallach, University of Chicago

MASON

- <http://cs.gmu.edu/~eclab/projects/mason/>
- Developed by George Mason University

RANA

- <https://github.com/sojoe02/RANA>
- Developed by Søren Vissing Jørgensen [XXX 15]

S. Jørgensen, Y. Demazeau, & J. Hallam, "RANA, a Real-Time Multi-Agent System Simulator" 15th Int. Conference on Intelligent Agent Technology, IAT'15, pp. 92-95, Singapore, 2015.

Ease of use versus modelling power



NETLOGO

A multi-agent programming language and integrated modelling environment

**VERY easy to use (novices), but can be restrictive..
Huge library of running models (downloadable or executable from their website)**

- Economics (e.g. stock exchange), Biology (e.g. AIDS), Physics, Chemistry, Earth Science (e.g. climate change), Mathematics (e.g. fractals), psychology, etc.

Based on the Logo language

Free but not open source

Large user base

SWARM

“Father of all ABM tools”

The fundamental component that organises the agents in the 'swarm'

A swarm is a collection of agents with a schedule of events over those agents

An agent can be composed of swarms of other agents in nested structures

Major impact in spreading the ABS methodology

SWARM Package

Simulation package.

- ▣ a library of object-oriented classes
- ▣ provides many tools for implementing, observing, and conducting experiments on ABMs
- ▣ Users write their own software but
 - ▣ follow Swarm's conceptual framework and conventions
 - ▣ use the Swarm libraries to do much of the work (library is written in Objective-C).
 - ▣ Originally, use Swarm using Objective-C. Now, can write in Java.

Platforms: Linux, Windows, or Mac, etc

Free

Strong user community (into recline..)

RePAST

Recursive Porous Agent Simulation Toolkit"

Re-designed and re-worked version of Swarm.

- but easier to use and better documented

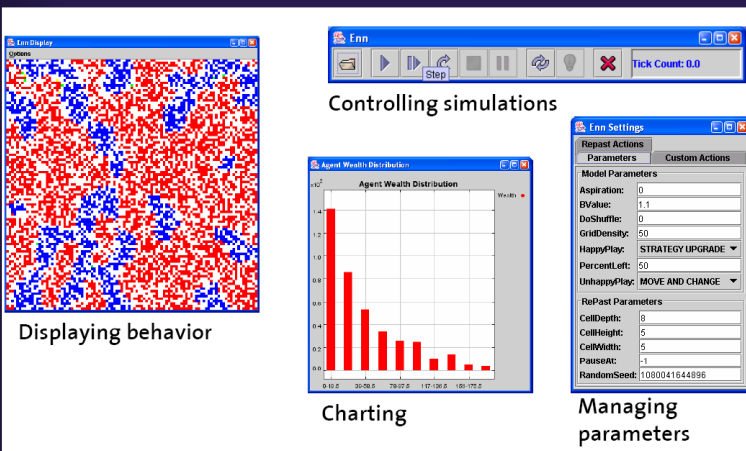
Repast focuses on social simulation, but can be used for any domain

Open-source, cross-platform, agent-based modelling and simulation toolkit.

Growing user community

- Major impact in showing the 'maturity' of Decentralized Simulation technology

RePAST Frameworks



Displaying behavior

Controlling simulations

Charting

Managing parameters

RePAST Implementations

Repast has multiple implementations in several languages

- Repast Symphony (Repast S)
- Repast for Java (Repast J)
- Repast for Microsoft .NET (Repast .NET)
- Repast for Python Scripting (Repast Py)

RepastJ, Repast.Net and RepastPy

- have now reached maturity
- are no longer being developed but still maintained
- have been superseded by Repast Symphony (RepastS)

Free

Relatively easy to use, widely used

MASON

Designed to be a smaller and faster alternative to Repast... It's fast indeed

Clear focus on computationally demanding models with many agents executed over many iterations.

Tends to include general rather than domain specific tools.

MASON carefully delineates between model and visualisation, allowing models to be dynamically detached from or attached to visualizers

Open source and free, JAVA based

Some common practical problems of frameworks

Differences in terminology

Concept/ Term	MASON	NetLogo	Repast	Swarm
Object that builds and controls simulation objects	model	observer	model	modelswarm
Object that builds and controls screen graphics	model WithUI	interface	(none)	observer swarm
Object that represents space and agent locations	field	world	space	space
Graphical display of spatial information	portrayal	view	display	display
User-opened display of an agent's state	inspector	monitor	probe	probe display
An agent behaviour or event to be executed	stappable	procedure	action	action
Queue of events executed repeatedly	schedule	forever procedure	schedule	schedule

Some common practical problems of frameworks

Documentation for many packages is largely incomplete

Installation of packages is not always obvious

Many test cases models do not run

Models developed in one version may not work in a later version (backwards incompatibility)

They may use obscure object oriented programming constructs

MULTIPLE DRONES

ALAVs (Autonomous Light Air Vessels) [Berk 07]

ALAVs [Berk 07]

<https://www.youtube.com/watch?v=8CKrjCUMhdA>

eMotionSpheres [FESTO 14]

ALAVs [Berk 07]

<https://www.youtube.com/watch?v=8CKrjCUMhdA>

eMotionSpheres [FESTO 14]

<https://www.youtube.com/watch?v=5iqP1oPZ3Qw>

Ehang 184 [Ayanian 17]

ALAVs [Berk 07]

<https://www.youtube.com/watch?v=8CKrjCUMhdA>

eMotionSpheres [FESTO 14]

<https://www.youtube.com/watch?v=5iqP1oPZ3Qw>

Ehang 184 [EHANG 16]

https://www.youtube.com/watch?v=_vGd1Oy7Cw0

Crazyswarm [Ayanian 17]

ALAVs [Berk 07]

<https://www.youtube.com/watch?v=8CKrjCUMhdA>

eMotionSpheres [FESTO 14]

<https://www.youtube.com/watch?v=5iqP1oPZ3Qw>

Ehang 184 [EHANG 16]

https://www.youtube.com/watch?v=_vGd1Oy7Cw0

Crazyswarm [Ayanian 17]

<https://www.youtube.com/watch?v=36jCUTWGBAo>

OFFSET [DARPA 19]

ALAVs [Berk 07]

<https://www.youtube.com/watch?v=8CKrjCUMhdA>

eMotionSpheres [FESTO 14]

<https://www.youtube.com/watch?v=5iqP1oPZ3Qw>

Ehang 184 [EHANG 16]

https://www.youtube.com/watch?v=_vGd1Oy7Cw0

Crazyswarm [Ayanian 17]

<https://www.youtube.com/watch?v=36jCUTWGBAo>

OFFSET [DARPA 19]

<https://www.youtube.com/watch?v=ruWC10AW87E>

MULTIPLE DRONES

ALAVs [Berk 07]

<https://www.youtube.com/watch?v=8CKrjCUMhdA>

eMotionSpheres [FESTO 14]

<https://www.youtube.com/watch?v=5iqP1oPZ3Qw>

Ehang 184 [EHANG 16]

https://www.youtube.com/watch?v=_vGd1Oy7Cw0

Crazyswarm [Ayanian 17]

<https://www.youtube.com/watch?v=36jCUTWGBAo>

OFFSET [DARPA 19]

<https://www.youtube.com/watch?v=ruWC10AW87E>

COMPLEMENTARY REFERENCES

Complementary references

Almost a VOWELS environment, even I is still missing

O. Boissier, R. Bordini, J. Hübner, A. Ricci & A. Santi, "Multi-Agent Oriented Programming with JaCaMo". *Science of Computer Programming*, Vol. 78, pp. 747–761, 2013.

The first reference to Interaction Oriented Programming

M. Huhns, "Interaction-Oriented Software Development", *Int. Journal of Software Engineering and Knowledge Engineering*, Vol. 11, pp. 259-279, 2001.

Introduction to Agent Oriented Programming introduction

Y. Shoham, *Agent-Oriented Programming*. *Artificial Intelligence*, 60(1):51–92, 1993.

A recent development of Interaction Oriented Programming

M. Singh, "Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language", *AAMAS 2011*, pp. 491-498, 2011.