ECOLANG – A COMMUNICATION LANGUAGE FOR SIMULATIONS OF COMPLEX ECOLOGICAL SYSTEMS

António Pereira
Pedro Duarte
CEMAS – UFP, University Fernando Pessoa
Praça 9 de Abril, 349, 4249-004 Porto, Portugal
E-mails: {apereira; pduarte}@ufp.pt

Luís Paulo Reis
NIAD&R - LIACC
Faculty of Engineering - University of Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
E-mail: lpreis@fe.up.pt

KEYWORDS

Ecological Modelling, Intelligent Agents, Simulation Models, Communication Language.

ABSTRACT

This document introduces ECOLANG, a communication language used for simulations of complex ecological systems. The language was developed with the main purpose of interchanging information between a simulation application of aquatic ecosystems (EcoDynamo) and several external agents. These agents have some goals about the simulation results or the simulated system. Examples of the former may be a calibration agent, with the goal of optimising the fit between observed and simulated results, whereas an example of the latter may be an aquaculture/farming agent, looking for production optimisation. This document focus on messages exchanged between EcoDynamo and the two mentioned agents providing examples of the communication protocol. This work is part of a larger project, where ECOLANG will be used as a tool for "goal-oriented intelligent simulations", towards sustainable management of coastal ecosystems.

INTRODUCTION

Coastal ecosystems are but a small part of the area covered by the seas. However, their importance is very large, considering that over 60% of the world population lives within 60 km from the sea. These ecosystems are used for multiple purposes such as fishing, tourism, aquaculture, harbour activities and as the final destination of many pollutants. This diversity of uses implies complex and, in most cases, conflicting management decisions. The huge number of possible combinations generated by the different management decisions and options, the opposite interests of stakeholders and some institutional authorities and the slowness of the decision process make it very difficult to implement efficient automatic management policies.

In this context, the use of intelligent agents (Weiss 1999; Wooldridge 2002) seems to be very promising. Each institutional authority and stakeholder may be modelled as an agent, interact with simulation tools - able to predict the outcome of different decisions - get results and configure new conditions for further simulation experiments.

The use of ecological modelling is a widespread practice in the management of coastal ecosystems. In particular, for coastal lagoons, located between land and open sea, where aquaculture plays an important role in local economy, the estimation of ecosystem carrying capacity can supply important indicators for ecosystem sustainability and returnable profit (Duarte et al. 2003).

Ecological models are simplified views of nature used to solve scientific or management problems. They include physical, chemical and biological processes to describe the main features of the ecosystem under analysis. One of the most important compromises is to find the optimal time and spaces scales for the model (Jørgensen and Bendoricchio 2001).

The simplest geometric representation is the zero-dimensional (0D) model, which simulates the system as a point and all changes are only time dependent. One-dimensional (1D) representation models assume that the system is characterized by a prevailing one-directional flow (horizontal or vertical) and the properties of the system vary along that direction and time. When the system is large enough to present sensible variation of the properties, vertical and horizontal division is required and two or three-dimensional (2D or 3D) representations are more common. Models of large lakes, coastal lagoons or river estuaries are examples of these representations.

The construction of mathematical models to make predictions about the evolution of an ecosystem, generally does not take into account the influence of the management decisions taken by the authorities. One possible way of doing that is to allow hypothetical authorities' decisions interacting with model simulation experiments.

This paper introduces ECOLANG, a communication language used to allow the interaction between ecological simulation experiments and several agents, representing either users of the system under simulation or applications designed to perform specific modelling tasks. The system conceptualisation, including a case study and the system architecture for the first application of ECOLANG is described in the next section.

Section 3 describes ECOLANG requirements, its messages format and types, and presents some examples. Finally, some conclusions are synthesised and directions of future research presented in the Conclusions section.

SYSTEM CONCEPTUALISATION

Case Study

The case study chosen for this work is Ria Formosa (south of Portugal); see Figure 1. This is a lagunary system with an area of c.a. 100 km^2, mostly included in a Natural Park. It is used for tourism, fishing, and aquaculture and as an harbour. Local population fluctuates dramatically between winter and summer due to tourism. Therefore, it is an example of an ecosystem with a lot of conflicting uses.

Ria Formosa is being modelled as a 2D vertically integrated, coupled hydrodynamic-biogeochemical model, based on a finite difference bathymetric staggered grid (Vreugdenhill 1989) with 282 lines by 470 columns (132540 grid cells) and a spatial resolution of 100m (Figure 1).

The model is forced by tidal height at the sea boundary, light intensity, air temperature, wind speed, cloud cover and boundary conditions for some of the state variables. The time step used is 3 seconds and the variables simulated are: water temperature, current speed and direction, dissolved substances, suspended matter and phytoplankton concentrations and bivalve species biomass.
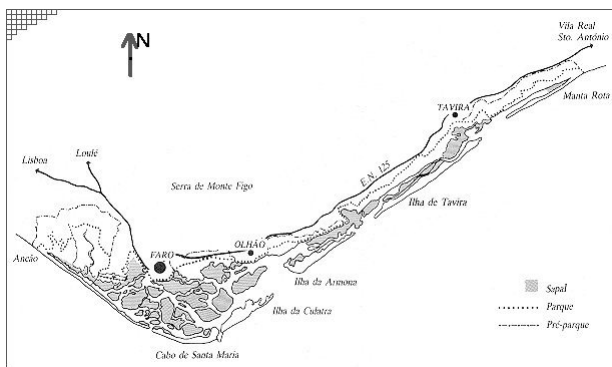

Figure 1 - Ria Formosa (South of Portugal). Part of the model grid is shown at the upper left corner (resolution of 100m)

System Architecture

The architecture proposed in this work will be exemplified using two agents - a Calibration agent and a Shellfish Farmer agent (Figure 2). These two agents are the first developed entities interacting with the simulation application EcoDynamo (Pereira & Duarte in prep).

EcoDynamo Application

EcoDynamo (**Eco**logical **Dyna**mics **Mo**del) is an application built to enable physical and biogeochemical simulation processes of aquatic ecosystems. It's an object oriented program application, built in C++ language, with a shell that manages simulation experiments runs, the graphical user interface, the communications between classes and the output devices where the simulation results are saved.

Different classes simulate different variables and processes, with proper parameter and process equations. Classes can be selected or deselected from shell dialogs determining its inclusion or exclusion in each simulation run of the model.
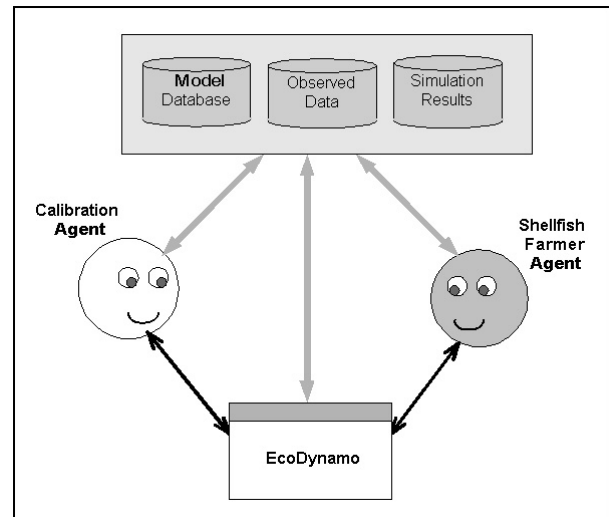

Figure 2 – Agent-based Simulation System Architecture

The simulated processes include:

- hydrodynamics of aquatic systems: current speeds and directions;

- thermodynamics: energy balances between water and atmosphere and water temperature;

- biogeochemical: nutrient and biological species dynamics;

- anthropogenic pressures, such as biomass harvesting.

The ecosystem characteristic properties are described in a model database: morphology (geometric representation of the model), dimensions (number of grid cells), classes, variables, parameter initial values and ranges.

In EcoDynamo there are different options available for the output of results – file output in text or hierarchical data format (*.hdf) – and graphical outputs. Both *.hdf and graphics output use MatLab® subroutines.

This application has an interface module (implementing the EcoDynamo Protocol based in ECOLANG) that enables communications with other programs for external control. For example, the simulation runs can be controlled by commands like start/stop/pause/restart/step simulation.

Simulation activity can be spied with the help of log files, activated previously before the simulation run.

Calibration Agent

The Calibration Agent (CA) is an Intelligent Agent (Wooldridge 2002) that communicates through a LAN with the simulation application with full control over the simulation (classes, time, parameters, etc.).

Details concerning the calibration procedure followed by the CA are described in (Pereira et al. 2004). Its main algorithm is depicted in Figure 3.
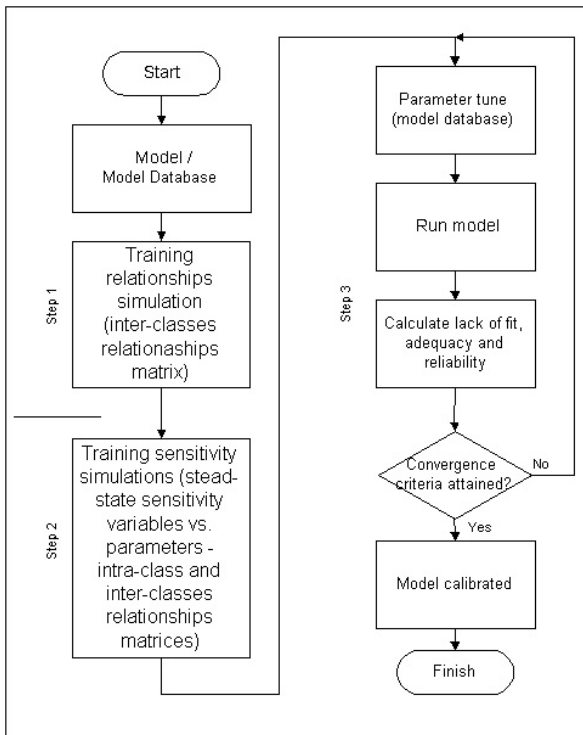

Figure 3 - Calibration Agent procedure diagram (from Pereira et al. 2004)

Its purpose is to tune model equation parameters in order to fit the model to observed data, towards model calibration and validation. The CA acquires knowledge about the behaviour of the system processes in three phases (Pereira et al. 2004):

- Building matrices that synthesize the interclass and inter-variable relationships;

- Analysing the intra and interclass steady-state sensitivity of different variables to different parameters and among variables;

- Iterative model execution, measuring model lack of fit, adequacy ad reliability (Sholten and Van der Tol 1998; Scholten et al. 1998) until a convergence criteria is attained.

Shellfish Farmer Agent

The Shellfish Farmer Agent works in a later phase, when the model is calibrated and validated. This agent simulates human reasoning and interaction with the ecosystem using very simple rules. Its general objective is to maximise bivalve production without exceeding ecosystem carrying capacity (Duarte et al. 2003). The agent seed actions (introducing shellfish juveniles in the coastal region) will be restricted by environment rules that allow seeding in some areas but not in others.

Ecological Modelling Language

ECOLANG is a high-level language capable of describing ecological systems in terms of regional characteristics, living agent's perceptions and actions. This language, along with a specific communication protocol, will enable the agents in the multi-agent system to understand each other in the ecological domain. It enables communications at different levels:

- Configuration – intended to select classes and change variables or parameter initial values to run the model.

- Execution – commands over the simulation execution (run, stop, pause,…).

- Statistics – intended to collect results from simulation experiments, either online or offline operation, compare results with previous simulation experiments or observed data and advise the configuration module the expected actions to take.

- Definitions – used to define regions of the model domain by names and aggregate several cells (or boxes) or regions into one region, according to common properties.

- Events – spontaneous messages that agents generate to inform some important events or results.

ECOLANG MESSAGES

Requirements

The format of the ECOLANG messages enables easy readability, simplicity and expandability. It also reflects the independence from any computational platform used for simulation. The main requirements for the message format are:

- Easy to extend to new concepts and definitions;

- Easily readable by the agents but also by humans;

- Robust enough to enable simple syntax validation;

- Easy to implement unambiguous error messages.

Message definitions used by ECOLANG follow the BNF formalism. Backus-Naur Form (BNF) is the best-known meta-language (a language used for describing languages) in the field of computer science. It was invented by John Backus and Peter Naur (Naur 1960) to describe the syntax of Algol 60 in an unambiguous manner.

ECOLANG notation is an extension to the original BNF formalism adding the following meta-symbols:

- **{ }** used for repetitive items (one or more times);

- **[ ]** encloses types of values;

- Terminal symbols use bold face letters.

The basic message structure is:

```
<MESSAGE> ::= message (<ID> <SENDER> <RECEIVER>
    <MSG_CONTENT>)
<ID> ::= [integer]
<SENDER> ::= [string]
<RECEIVER> ::= [string]
<MSG_CONTENT>::= <DEFINITION_MSG> |
    <ACTION_MSG> | <PERCEPTION_MSG>
```

<ID> - The identifier of the message sent by sender - each sender manages its own numbering system for the messages.

<SENDER> - The name of the agent or application that sends the message. It must not have spaces.

<RECEIVER> - The name of the agent or application target of the message. It must not have spaces.

<MSG_CONTENT> - The message body properly.

Messages can be from three basic types: definitions, actions and perceptions. While definitions are generic messages (used by anyone of the communication partners), actions and perceptions are specific to each kind of involved agent.

ECOLANG syntax with some examples is demonstrated in the next paragraphs. The examples are for a Calibration Agent and a Shellfish Farmer Agent. The former includes Action and Perception messages that allow controlling model runs, to initialise model parameters and evaluate model results, respectively. The general approach is to compare these results with real data and change model parameters iteratively, until some convergence criteria are attained (Pereira et al. 2004). The latter example includes Action, Perception and Definition messages, to control model runs, analyse model results and define cultivation strategies, iteratively.

Definitions

Messages used for definitions are limited, for the time being, to define regions. Each region will be referenced by a name, a specific type and a specific area. It can be defined, also, as a union of other regions:

```
<DEFINITION_MSG> ::= define (<REG_NAME>
    <REGION>)
<REG_NAME> ::= [string]
<REGION> ::= <REGION_TYPE> <REGION_AREA> |
    {<REG_NAME>}
```

The type of region defines land or water and, in this case, it will be characterized by its quality and the type and quality of its sediments:

```
<REGION_TYPE> ::= <LAND_REGION> |
    <WATER_REGION>
<LAND_REGION> ::= land
```

```
<WATER_REGION> ::= <WATER_CARACT>
    <SEDIMENT_CARACT>
<WATER_CARACT> ::= <SUB_INTERTIDAL>
    <WATER_QUALITY>
<SUB_INTERTIDAL> ::= subtidal | intertidal
<WATER_QUALITY> ::= <QUAL_SCALE>
<QUAL_SCALE> ::= excellent | good | poor
<SEDIMENT_CARACT> ::= (<SEDIMENT_TYPE>
    <SEDIMENT_QUALITY>)
<SEDIMENT_TYPE> ::= sandy | sand_muddy | muddy
<SEDIMENT_QUALITY> ::= <QUAL_SCALE>
```

The region area is the union of one or more simple regions, each one of these defined by a point or a basic polygon (rectangle, square, circle or arc):

```
<REGION_AREA> ::= {<SIMPLE_REGION>}
<SIMPLE_REGION> ::= <POINT> | (rect <POINT>
    <POINT>) | (square <POINT> <POINT> <POINT>
    <POINT>) | (circle <POINT> [real]) | (arc
    <POINT> [real] [real] [real] [real])
<POINT> ::= (point [real] [real])
```
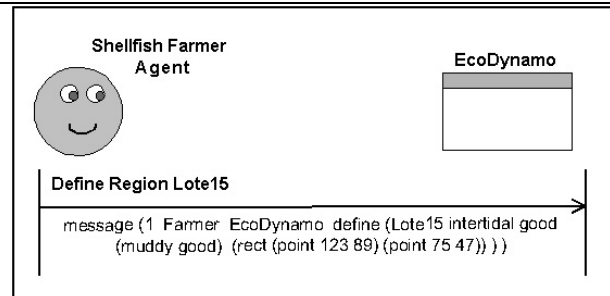

Figure 4 – Definition message

Shellfish Farmer Agent

Actions

The shellfish farmer agent cultivates bivalves and its actions are seed, inspect and harvest bivalves:

```
<ACTION_MSG> ::= <SEED_ACTION> |
    <INSPECT_ACTION> | <HARVEST_ACTION>
```

When the shellfish farmer agent wants to seed bivalves, it indicates the characteristics of the bivalves and where they will be seeded. To inspect, it indicates the region(s). To harvest, it indicates the region(s), the characteristic of the bivalves to harvest and when harvest should occur:

```
<SEED_ACTION> ::= seed (<REG_NAME> <TIME>
    <BIVALVE> <WEIGHT>)
<INSPECT_ACTION> ::= inspect (<REG_NAME>
    <TIME>)
<HARVEST_ACTION> ::= harvest (<REG_NAME> <TIME>
    <BIVALVE>)
<BIVALVE> ::= <BTYPE> <BCARACT>
<BTYPE> ::= mussel | oyster
<BCARACT> ::= <SHELL_LENGTH> | <DENSITY>
<SHELL_LENGTH> ::= (length [real])
<DENSITY> ::= (density [real])
```

The reference time can be immediately (now) or a value that is the number of seconds from January 1, 1970 00:00:

```
<TIME> ::= now | [integer]
```

Perceptions

The perceptions of this agent are the answers to the actions that, previously, it has done:

```
<PERCEPTION_MSG> ::= <SEED_RESULT> |
    <INSPECT_RESULT> | <HARVEST_RESULT>
```

The answer to the seed action can be positive or negative. The answer to the inspect action will be a message with bivalve characteristics. The answer to the harvest action can be negative or positive and, in this case, the kind and amount of bivalves harvested:

```
<SEED_RESULT> ::= seed_result (<ACTION_ID>
    <ACTION_RESULT>)
<INSPECT_RESULT> ::= inspect_result
    (<ACTION_ID> {<BIVALVE>})
<HARVEST_RESULT> ::= harvest_result
    (<ACTION_ID> <ACTION_RESULT> <WEIGHT>)
<ACTION_RESULT> ::= ok | failed
<ACTION_ID> ::= <ID>
<WEIGHT> ::= [real]
```
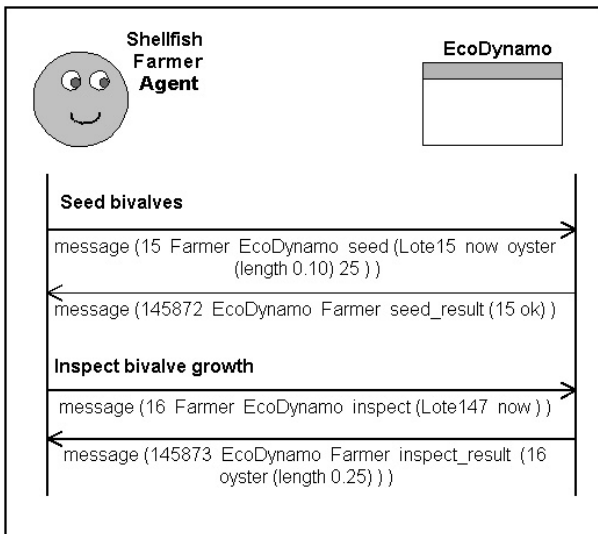
The <ACTION_ID> field identifies the <ID> of the action message.

Figure 5 – Farmer Agent seeding and inspecting regions

Calibration Agent

Actions

The calibration agent is designed for model calibration, in order to approximate simulation results to observed data. The actions allowed to this agent are divided in four different types:

```
<ACTION_MSG> ::= <MODEL_ACTION> | <EXEC_ACTION>
    | <SPECS_ACTION> | <REG_ACTION>
```

- Actions to select the simulation model: open or close model, ask the model in simulation:

```
<MODEL_ACTION> ::= <OPEN_MODEL> | <CLOSE_MODEL>
    | <GET_MODEL>
<OPEN_MODEL> ::= open_model <MODEL_NAME>
<MODEL_NAME> ::= [string]
<CLOSE_MODEL> ::= close_model
<GET_MODEL> ::= model_name
```

- Actions over the simulation execution: initialise, run, pause, step, stop:

```
<EXEC_ACTION> ::= initialise | run | stop |
    step | pause
```

- Actions over the simulation model: select/enquiry classes, initial values for variables and parameters, time step and time interval and simulation subdomain (part of the model grid):

```
<SPECS_ACTION> ::= <SP_CLASSES> | <SP_VARS> |
    <SP_PARMS> | <SP_TIME> | <SUB_DOMAIN>
<SP_CLASSES> ::= <GET_CLASSES> |
    <SELECT_CLASSES>
<GET_CLASSES> ::= get_available_classes |
    get_selected_classes
<SELECT_CLASSES> ::= select_classes
    {<CLASS_NAME>}
<CLASS_NAME> ::= ([string])
<SP_VARS> ::= <GET_CLASS_VARS> |
    <GET_VAR_VALUE> | <SET_VAR_VALUE>
<GET_CLASS_VARS> ::= get_variables <CLASS_NAME>
<GET_VAR_VALUES> ::= get_variable_value
    <CLASS_NAME> <VAR_NAME> <CELL>
<VAR_NAME> ::= ([string])
<CELL> ::= [integer]
<SET_VAR_VALUE> ::= set_variable_value
    <CLASS_NAME> {(<VAR_NAME> <BOXES> [real])}
<BOXES> ::= (<SUB_DOMAIN>) | ({<CELL>})
<SP_PARMS> ::= <GET_PARMS> | <SET_PARMS>
<GET_PARMS> ::= get_parameters <CLASS_NAME>
<SET_PARMS> ::= set_parameters <CLASS_NAME>
    {(<PARM_NAME> [real])}
<PARM_NAME> ::= ([string])
<SP_TIME> ::= <GET_TIME> | <SET_TIME>
<GET_TIME> ::= get_time_spec
<SET_TIME> ::= set_time_spec <STEP>
    <START_TIME> <FINISH_TIME>
<STEP> ::= [integer]
<START_TIME> ::= [integer]
<FINISH_TIME> ::= [integer]
<SUB_DOMAIN> ::= subdomain <DOMAIN>
<DOMAIN> ::= all | ({<REG_NAME>})
```

- Actions over the output results: select registry variables, type, frequency, time interval and subdomain of register and activate survey mode:

```
<REG_ACTION> ::= <REG_FILE> | <REG_VARS> |
    <REG_LOG> | <REG_TIME> | <REG_TRACE>
<REG_FILE> ::= output_file <FILE_NAME>
<FILE_NAME> ::= ([string])
<REG_VARS> ::= <GET_VARS> | <SELECT_VARS>
<GET_VARS> ::= get_available_variables
<SELECT_VARS> ::= select_variables
    <OUTPUT_TYPE> ({<VAR_NAME>})
    (<SUB_DOMAIN>)
<OUTPUT_TYPE> ::= file | graph | table
<REG_LOG> ::= log <LOG_TYPE> ({<LOG_STEP>})
<LOG_TYPE> ::= xml | xls | txt
<LOG_STEP> ::= [integer]
```

```
<REG_TIME> ::= <GET_REG_TIME> | <SET_REG_TIME>
<GET_REG_TIME> ::= get_output_time
<SET_REG_TIME> ::= set_output_time <STEP>
    <START_TIME> <FINISH_TIME>
<REG_TRACE> ::= trace
```

Perceptions

The perceptions of the calibration agent could be answers to previous actions initiated by the agent itself or spontaneous messages sent by the simulation application when significant events occur. There are five types of perceptions:

```
<PERCEPTION_MSG> ::= <MODEL_RESULT> |
    <EXEC_RESULT> | <SPECS_RESULT> |
    <REG_RESULT> | <EVENT_MSG>
```

- Answers to actions over the model:

```
<MODEL_RESULT> ::= <OPEN_RESULT> |
    <CLOSE_RESULT> | <GET_RESULT>
<OPEN_RESULT> ::= open_result (<ACTION_ID>
    <ACTION_RESULT>)
<CLOSE_RESULT> ::= close_result (<ACTION_ID>
    <ACTION_RESULT>)
<GET_RESULT> ::= model (<ACTION_ID>
    <MODEL_NAME>)
```


Figure 6 – Calibration Agent opens model
"RiaF2D100m"

- Answers to actions over the simulation model:

```
<SPECS_RESULT> ::= <CLASSES_RESULT> |
    <VARS_RESULT> | <PARMS_RESULT> |
    <TIME_RESULT> | <SUB_DOMAIN_RESULT>
<CLASSES_RESULT> ::= <CLASSES_AVAILABLE> |
    <CLASSES_SELECTED>
<CLASSES_AVAILABLE> ::= classes_available
    (<ACTION_ID> {<CLASS_NAME>})
<CLASSES_SELECTED> ::= classes_selected
    (<ACTION_ID> {<CLASS_NAME>})
<VARS_RESULT> ::= <CLASS_VARS> | <VAR_VALUE> |
    <VAR_SET>
<CLASS_VARS> ::= variables (<ACTION_ID>
    {<VAR_NAME>})
```
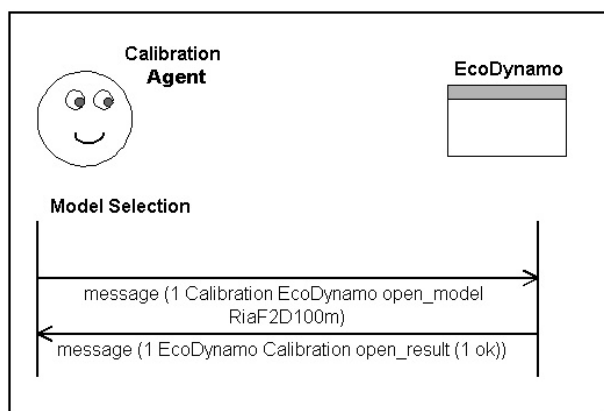
```
<VAR_VALUE> ::= variable_value (<ACTION_ID>
    <VAR_NAME> <CELL> [real])
<VARS_SET> ::= variable_set_result (<ACTION_ID>
    <ACTION_RESULT>)
<PARMS_RESULT> ::= <CLASS_PARMS> | <PARMS_SET>
<CLASS_PARMS> ::= parameters_value (<ACTION_ID>
    {(<PARM_NAME> [real])})
<PARMS_SET> ::= parameters_set_result
    (<ACTION_ID> <ACTION_RESULT>)
<TIME_RESULT> ::= time_spec (<ACTION_ID> <STEP>
    <START_TIME> <FINISH_TIME>)
<SUB_DOMAIN_RESULT> ::= subdomain_result
    (<ACTION_ID> <ACTION_RESULT>)
```
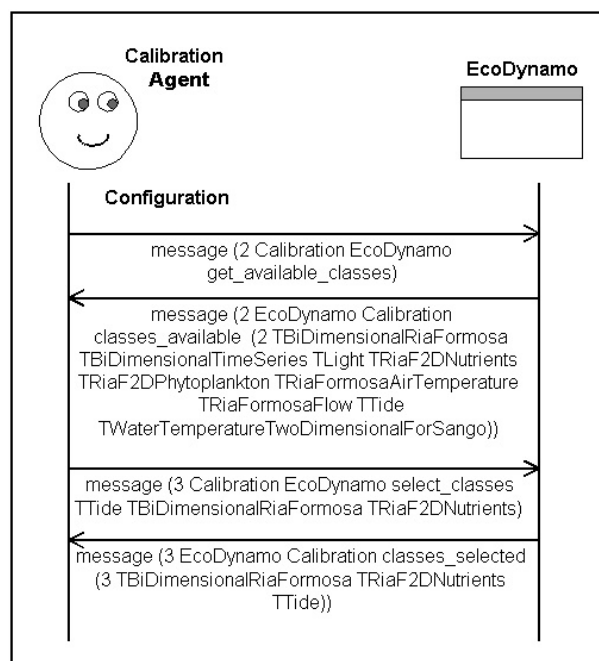

Figure 7 – Calibration Agent configuring simulation

- Answers to actions over the simulation execution:

```
<EXEC_RESULT> ::= exec_result (<ACTION_ID>
    <ACTION_RESULT>)
```


Figure 8 – Calibration Agent runs model
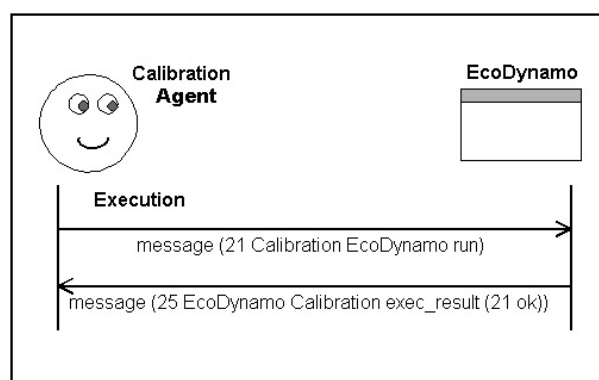
- Answers to actions over the output register:

```
<REG_RESULT> ::= <FILE_RESULT> |
    <REG_VARS_RESULT> | <LOG_RESULT> |
    <REG_TIME_RESULT> | <TRACE_RESULT>
<FILE_RESULT> ::= output_file_result
    (<ACTION_ID> <ACTION_RESULT>)
<REG_VARS_RESULT> ::= <GET_VARS_RESULT> |
    <SELECT_VARS_RESULT>
<GET_VARS_RESULT> ::= variables_available
    (<ACTION_ID> {<VAR_NAME>})
<SELECT_VARS_RESULT> ::= se-
    lect_variables_result (<ACTION_ID>
    <ACTION_RESULT>)
<LOG_RESULT> ::= log_result (<ACTION_ID>
    <ACTION_RESULT>)
<REG_TIME_RESULT> ::= output_time(<ACTION_ID>
    <STEP> <START_TIME> <FINISH_TIME>)
<TRACE_RESULT> ::= trace_result <TRACE_STATUS>
<TRACE_STATUS> ::= on | off
```

- Spontaneous messages from simulation application:

```
<EVENT_MSG> ::= <REG_MSG> | <LOG_MSG>
<REG_MSG> ::= register (<REG_INDEX> <REG_TIME>
    <CELL> <VAR_NAME> [real])
<REG_INDEX> ::= [integer]
<GET_TIME> ::= [integer]
<LOG_MSG> ::= logger (<STEP_NR> <CLASS_NAME>
    <FUNC_TYPE> <DATA_CLASS> <VAR_NAME> <CELL>
    [real])
<STEP_NR> ::= [integer]
<FUNC_TYPE> ::= Inquiry | Update
<DATA_CLASS> ::= <CLASS_NAME>
```
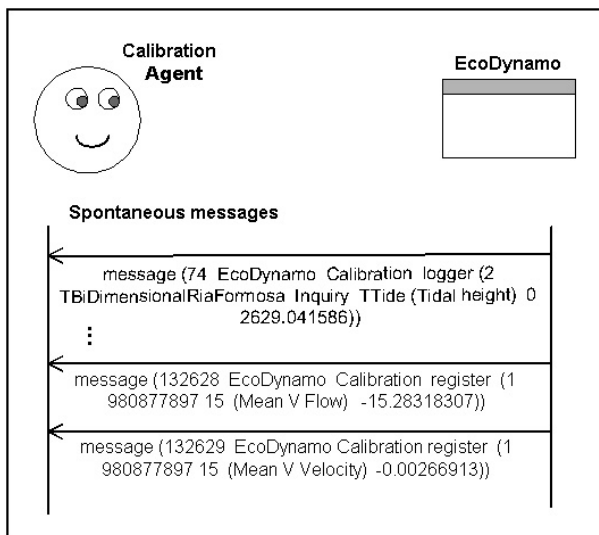

Figure 9 – Calibration Agent receiving spontaneous messages

CONCLUSIONS AND FUTURE WORK

The definition of a communication language for simulation of complex ecological systems is the first step to an open road of possibilities to make the simulation experiments of ecological models more interactive and comprehensible to end users. With high-level messages,

it is possible to build intelligent agents that can intermediate the simulation experiments and the users.

With this language, it is possible to change some model characteristics and system configuration during simulation experiments. It may be very useful for the calibration of large and complex models – a generally hard and tedious process – if one may define an agent, capable of simulating the trial and error learning process that any modeller has to go through, when calibrating a model.

Generally, the management of coastal ecosystems may be done in many different ways and there is hardly one optimal solution, but most likely a "family" of "good" management options. Giving the large complexity of these systems and the numerous synergies between environmental conditions and management options, finding "good" choices cannot be reduced to simple optimisation algorithms, assuming linear or some defined form of non-linear relationship between a set of parameters, variables and goal seeking functions. Mathematical models may be very useful in finding "good" management solutions. However, finding these may require many trial and error simulation experiments and this is why using agents that may look automatically for the mentioned solutions may be advantageous. This will require the *a priori* definition of "good" solutions and constraints. For example, one may wish to increase aquaculture production but keeping water quality within certain limits for other uses. In any case, a high level language as ECOLANG is necessary to link simulation software with specialized agents.

REFERENCES

Duarte, P.; R. Meneses; A.J.S. Hawkins; M. Zhu; J. Fang and J. Grant. 2003. "Mathematical modelling to assess the carrying capacity for multi-species culture within coastal waters", *Ecological Modelling* 168 (2003): 109-143.

European Commission. 2003. "Development of an Information Technology Tool for the Management of European Southern Lagoons under the influence of river-basin runoff – The DITTY Project". S.P.I.03.135, Directorate-General of Joint Research Centre, Institute for Environment and Sustainability.

Jørgensen, S.E. and G. Bendoricchio. 2001. *Fundamentals of Ecological Modelling*, Elsevier Science Ltd, 3rd edition.

Naur, P. (ed.). 1960. "Revised Report on the Algorithmic Language ALGOL 60.", *Communications of the ACM*, Vol. 3 No.5 (May), 299-314.

Pereira, A. and P. Duarte. in prep. EcoDynamo – Ecological Dynamics Model Application, Technical Report, University Fernando Pessoa.

Pereira, A.; P. Duarte and L. P. Reis. 2004. "Agent-Based Ecological Model Calibration – On the Edge of a New Approach". In: C. Ramos and Z. Vale (eds), *Proceedings of the International Conference on Knowledge Engineering and Decision Support*, pp. 107-113, ISEP, Porto, Portugal, July. ISBN: 972-8688-24-5.

Scholten H. and M.W.M. Van der Tol. 1998. Quantitative validation of deterministic models: when is a model acceptable? In: *The proceedings of the Summer Computer Simulation Conference, SCS*, San Diego, CA, USA: 404-409, ISBN: 1-56555-149-4.

Scholten H.; M.W.M. Van der Tol and A.C. Smaal. 1998. *Models or measurements? Quantitative validation of an ecophysiological model of mussel growth and reproduction*. Paper presented at the ICES Annual Science Conference, Cascais, Portugal.

Vreugdenhil, C.B. 1989. *Computatinal hydraulics, An introduction*, Springer-Verlag.

Weiss, G. (ed.). 1999. *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press.

Wooldridge, M. 2002. *An Introduction to Multi-Agent Systems*, John Wiley & Sons, Ltd.

AUTHOR BIOGRAPHIES

ANTÓNIO PEREIRA was born in Porto, Portugal and obtained a degree in Electrotechnical Engineering and a MSc in Artificial Intelligence and Computation in the University of Porto. After graduation, he worked 15 years in EFACEC Electronic Systems (a portuguese industrial company) and 2 years in Siemens, Information and Communications group. Since 2003 he is dedicated to research in the field of Agent-Based Simulation systems and is a PhD student in Electrotechnical Engineering. His e-mail address is apereira@ufp.pt.

PEDRO DUARTE was born in Lisbon, Portugal, obtained his degree in Biology at the University of Lisbon and his PhD in Environmental Sciences at the New University of Lisbon, in 1995. He has been involved in several European projects doing research in the field of Ecological Modelling applied to environmental problems such as carrying capacity estimation for aquaculture. His e-mail address is pduarte@ufp.pt.

LUIS PAULO REIS was born in Porto, Portugal and have a PhD in Electrotechnical Engineering (Coordination in Multi-Agent Systems) in the University of Porto. He has been researching in the area of (Multi-Agent) intelligent simulation for several years in different projects including FC Portugal simulated robotic soccer team – World and European champion in 2000. His e-mail address is lpreis@fe.up.pt and his web page can be found at http://www.fe.up.pt/~lpreis.