



Intelligent Simulation of Coastal Ecosystems

António Manuel Correia Pereira

Department of Informatics Engineering

Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

July 2010



Intelligent Simulation of Coastal Ecosystems

António Manuel Correia Pereira

Master in Artificial Intelligence and Computation
- Faculty of Engineering, University of Porto -
Graduate in Electrical Engineer (Digital Systems and Computers)
- Faculty of Engineering, University of Porto -

Public dissertation submitted to Faculty of Engineering, University of Porto, as part of requisites to obtain the doctoral grade in Informatics Engineering.

Supervised by
Professor Luís Paulo Gonçalves dos Reis
Professor Pedro Manuel da Silva Duarte
Porto, July 2010

*To my parents, Lucinda and Joaquim, that have always
believed the greatest assets that parents can leave to
their children is education.*

Abstract

The simulation of physical, chemical and biological processes in coastal ecosystems is used as a way to understand the system internal dynamics and to predict its evolution over time, in order to promote behaviours environmentally friendly and to induce effective and efficient management of the ecosystem as a whole.

The complexity and diversity of those processes encompass wide areas of knowledge, involving many researchers and research teams for their mathematical modelling. Each research team adopts a given programming language to translate the model to a computer application, simulating the processes that they are specialists and are interested in, rarely thinking about the possibility of its integration with other models developed by other research teams, that simulate complementary processes. These applications are normally self-contained and, when used in a management context, require a lot of extra work to export and import results from one application to another, in order to share the knowledge acquired and complement the simulation. Furthermore, they do not integrate any kind of human rationality embedded to help a decision making activity.

The work presented in this document explores the capacity for the realistic simulation of complex systems, working consistently, integrating results generated by processes simulated in distinct applications and placing the human reasoning in the middle of the decision making system. It presents a framework for modelling and simulating coastal ecosystems and an associated methodology for creating an Environmental Decision Support Systems (EDSS). Both are based on object oriented programming (OOP) and, in the case of the EDSS, on Autonomous Intelligent Agents. The modelling software simulator (EcoDynamo) is written in C++ and includes several object dynamic link libraries (DLLs) for the simulation of the different physical and biogeochemical processes. These libraries were designed to be linked with different model shells, possibly written in different programming languages, for the sake of portability and reusability. A high level communication language (ECOLANG) was developed to allow the communications between EcoDynamo and the agents and applications that belong to the system. ECOLANG was designed to describe ecological systems in terms of regional characteristics, living agent's perceptions and actions and is independent from any hardware or software platform. The framework (Ecological Simulation Network – EcoSimNet) was developed to easily integrate and bring together the several pieces of the system – the simulator is the core of the framework and all the agents/applications communicate with it;

the agents have, also, the ability to communicate with each other and can define several scenarios for the simulation in order to optimize their own objectives. To improve the speed of the simulation process, the infrastructure provides mechanisms to integrate several simulators, enabling the parallel simulation of different configuration scenarios to increase the simulation speed. The EDSS uses the Analytic Hierarchy Process methodology (AHP) to integrate multiple qualitative and quantitative conflicting criteria. The user gives a structure with a pairwise comparison between criteria based on its importance, and a priority ranking of the pre-processed scenarios is achieved. To validate the portability of the DLLs, some objects were integrated in the COHERENS simulator (written in Fortran), and some processes simulated by COHERENS were invoked by the EcoDynamo simulator.

To validate the EcoSimNet framework, the simulation system was used both to simulate the Sungo Bay model (People's Republic of China) and Ria Formosa (Algarve, Portugal). The simulation system was used to simulate several distinct ecosystem configurations in order to demonstrate its flexibility. The framework was also used to optimize different scenarios with bivalve farming areas, and the EDSS was used with different management scenarios in the Ria Formosa lagoon (Portugal). The experiments performed indicate that these tools may be widely used by people involved in the management of coastal areas to integrate environmental, economic and social issues in the decision process, without an in-depth knowledge of modelling methodologies.

Resumo

A simulação de processos físicos, químicos e biológicos em ecossistemas costeiros é utilizada como uma forma de entender a dinâmica interna do sistema e prever a sua evolução ao longo do tempo, a fim de promover comportamentos ambientalmente amigáveis e para induzir a gestão eficiente e eficaz do ecossistema como um todo.

A complexidade e a diversidade dos processos abrangem grandes áreas do conhecimento, envolvendo muitos investigadores e equipas de investigação para a sua modelação matemática. Cada equipa de investigação adopta uma linguagem de programação traduzindo o modelo para uma aplicação de computador, simulando os processos em que são especialistas e estão interessados, raramente pensando na possibilidade da sua integração com outros modelos desenvolvidos por outras equipas de investigação, que simulam processos complementares. Estas aplicações são normalmente auto-suficientes e, quando usadas num contexto de gestão, exigem muito trabalho extra para exportar e importar os resultados de uma aplicação para outra, a fim de partilhar o conhecimento adquirido e completar a simulação. Além disso, não integram qualquer tipo de racionalidade humana para auxiliar uma actividade de tomada de decisão.

O trabalho apresentado neste documento explora a capacidade de simulação de sistemas complexos, trabalhando de forma consistente, integrando os resultados gerados por diferentes processos simulados e colocando o factor humano no meio do processo de tomada de decisão. Apresenta uma plataforma para modelação e simulação de ecossistemas costeiros e uma metodologia associada à criação de um Sistema de Apoio à Decisão Ambiental (Environmental Decision Support System - EDSS). Ambos são baseados em programação orientada a objectos (OOP) e, no caso do EDSS, em Agentes Inteligentes. O software do simulador (EcoDynamo) é escrito em C++ e inclui várias bibliotecas de objectos de ligação dinâmica (Dynamic Link Libraries - DLLs) para a simulação dos diferentes processos físicos e biogeoquímicos. As bibliotecas foram concebidas para serem utilizadas por outros programas de simulação, possivelmente escritos em diferentes linguagens de programação, permitindo portabilidade e reutilização. Uma linguagem de comunicação de alto nível (ECOLANG) foi desenvolvida para permitir a comunicação entre o EcoDynamo, os agentes e as aplicações que pertencem ao sistema. A ECOLANG foi projectada com o objectivo de descrever os sistemas ecológicos, considerando características regionais, as percepções e as acções dos agentes e é independente de qualquer plataforma de hardware ou software. A plataforma EcoSimNet

(Ecological Simulation Network) foi desenvolvida para integrar e reunir as diversas partes do sistema - o simulador é a aplicação central da plataforma e todos os agentes/aplicações comunicam com ele; os agentes têm, também, a capacidade para comunicar uns com os outros e podem definir vários cenários para a simulação, a fim de otimizar os seus próprios objectivos. Para melhorar a velocidade do processo de simulação, a infra-estrutura dispõe de mecanismos para integrar vários simuladores, permitindo a simulação paralela de diferentes cenários de configuração, aumentando a frequência de geração de resultados. O EDSS usa o método AHP (Analytic Hierarchy Process) para integrar múltiplos critérios antagónicos, descritos qualitativa ou quantitativamente. Neste método o utilizador insere a importância dos critérios numa matriz, comparando-os aos pares, e o AHP apresenta uma ordem de prioridade para os cenários pré-processados.

Para validar a portabilidade das bibliotecas dinâmicas, alguns objectos foram integrados no simulador COHERENS (escrito em Fortran) e alguns processos simulados pelo COHERENS foram invocados a partir do simulador EcoDynamo.

Para validar a plataforma EcoSimNet, o sistema foi utilizado com o modelo de Sungo Bay (República Popular da China) para otimizar diferentes cenários de cultura de bivalves, e o EDSS foi utilizado com diferentes cenários de gestão da Ria Formosa (Portugal). As experiências realizadas indicam que estas ferramentas podem ser utilizadas na gestão das zonas costeiras, integrando as questões ambientais, económicas e sociais no processo de decisão, por utilizadores sem um conhecimento profundo das metodologias de modelação.

Résumé

La simulation des propriétés physiques, chimiques et biologiques dans les écosystèmes côtiers est utilisée comme un moyen de comprendre la dynamique interne et de prédire leur évolution au fil du temps, afin de promouvoir des comportements respectueux de l'environnement et à induire une gestion efficace de l'écosystème dans son ensemble.

La complexité et la diversité des processus englobent de vastes domaines de connaissances, impliquant de nombreux chercheurs et équipes de recherche pour leur modélisation mathématique. Chaque équipe de recherche adopte un langage de programmation pour traduire le modèle d'une application informatique pour simuler les processus qui sont des spécialistes et sont intéressés par, rarement penser à la possibilité de son intégration avec d'autres modèles élaborés par d'autres équipes de recherche, qui simulent des processus complémentaires. Ces applications sont généralement autonomes et, lorsqu'il est utilisé dans un contexte de gestion, exigent beaucoup de travail supplémentaire pour exporter et importer les résultats d'une application à une autre, afin de partager les connaissances acquises et de compléter la simulation. En outre, ils ne s'intègrent pas n'importe quel type de rationalité humaine intégrés pour aider à une décision faisant de l'activité.

Le travail présenté dans ce document explore la capacité de la simulation de systèmes complexes, en travaillant de manière cohérente, intégrant les résultats générés par les processus simulés dans des applications distinctes et à la mise au raisonnement de l'homme dans le milieu du processus décisionnel. Il présente un cadre pour la modélisation et la simulation des écosystèmes côtiers et une méthodologie associée pour créer un Système d'Aide à la Décision Environnementale (Environmental Decision Support Systems - EDSS). Les deux sont basés sur la programmation orientée objet (POO) et, dans le cas de l'EDSS, sur les agents autonomes intelligents. Le simulateur logiciel de modélisation (EcoDynamo) est écrit en C++ et comprend plusieurs bibliothèques d'objets de liens dynamiques (Dynamic Link Library - DLL) pour la simulation de les différents processus physique et biogéochimiques. Ces bibliothèques ont été désigné pour être lié avec des différent simulateurs, peut-être écrites dans des langages de programmation différents, permettant la réutilisation et la portabilité. Un langage de haut niveau (ECOLANG) a été développé pour permettre les communications entre EcoDynamo, les agents et les applications qui appartiennent au système. ECOLANG a été désigné pour décrire des systèmes écologiques en termes de caractéristiques régionales, les perceptions et les actions des agents et est indépendante de tout matériel ou plate-forme

logicielle. La plate-forme EcoSimNet (Ecological Simulation Network) proposée a été conçue pour intégrer facilement et de réunir les pièces du système à plusieurs - le simulateur est le noyau du plate-forme et tous les agents/applications communiquer avec lui; les agents ont, en outre, la capacité de communiquer les uns avec les autres et permet de définir plusieurs scénarios pour la simulation afin d'optimiser leurs propres objectifs. Pour améliorer la rapidité du processus de simulation, l'infrastructure fournit des mécanismes pour intégrer plusieurs simulateurs, permettant la simulation parallèle des scénarios de configuration différentes pour augmenter la fréquence de génération de résultats. L'EDSS utilise la méthode AHP (Analytic Hierarchy Process) pour intégrer de multiples critères contradictoires, décrit qualitativement ou quantitativement. L'utilisateur donne une structure avec une comparaison par paires entre les critères en fonction de son importance, et l'AHP réalise un ordre de priorité des scénarios prétraitement.

Pour valider la portabilité des DLLs, certains objets ont été intégrées dans le simulateur COHERENS (écrit en Fortran), et certains processus simulés par COHERENS ont été invoqués par le simulateur EcoDynamo.

Pour valider la plate-forme EcoSimNet, le système a été utilisé avec le modèle Sungo Bay (République Populaire de Chine) afin d'optimiser les différents scénarios avec répartition des zones de conchyliculture, et le score EDSS a été utilisé avec différents scénarios de gestion dans la lagune de Ria Formosa (Portugal). Les expériences réalisées montrent que ces outils peuvent être largement utilisés dans la gestion des zones côtières à intégrer les enjeux environnementaux, économiques et sociaux dans la prise de décision, pour les utilisateurs sans une connaissance approfondie des méthodes de modélisation.

Acknowledgements

Writing a thesis is an opportunity to thank all the people involved in our lives for their support, encouragement, friendship, dedication, tolerance, tenure and love.

Personally, I have to thank a million of souls who allowed me to share incredible moments, unforgettable experiences, encouraged and pushed me for this amazing research task started about twenty years after the graduation, and gave me support to achieve this stage in my professional career.

First of all I have to thank my advisors, professor Pedro Duarte and professor Luís Paulo Reis, for their friendship, their omnipresent criticism to my work, their constant availability to clarify my doubts and my inexperience, and their guidance during this difficult process of writing a dissertation. I will never have words to thank your persistence with me. I know I won two great friends for the rest of my life.

To you, Pedro, thanks for your professionalism, your joy, your advices, your constant inspirational tips and confidence in my work - all I know about biology, ecology, modelling and natural sciences were learned from you. Thank you.

To you, Luís, thanks for your believe in my work. Since the beginning you never doubted about my capacity, even when I passed for less pleasant moments. Your imagination allied to an enormous rhythm of work and will to win in everything you put your hands, surpassed always my expectations and brought me again to the battle. Thank you.

I have to thank University Fernando Pessoa the reception they gave me at the beginning of this adventure and the support provided to perform my job. I really enjoyed working there with great people, namely Nelson Barros, Ana Fonseca, Maria João, José Torres, Álvaro Rocha and Lemos de Sousa. I will never forget their kindly advices.

I extend my thanks to the Faculty of Engineering - University of Porto (FEUP), in particular to the Artificial Intelligence and Computer Science Laboratory (LIACC), namely the Distributed Artificial Intelligence & Robotics Group (NIAD&R), the reception, support and encouragement they have given me, and the belief that this task would end. I am fortunate to work in the LIACC/NIAD&R group and collaborate with accomplished researchers, like professor Eugénio Oliveira, the heart of the laboratory, from whom I have always a lot to learn. His criticism,

sometimes provocative, and incisive advices, always forced me to reflect about my work, and seek continuously the novelties it can improve.

A special thank to my colleagues and friends that always pulled me up in University Fernando Pessoa, in Faculty of Engineering, University of Porto, in Viana do Castelo Polytechnic Institute and in ISPGaya. I know that I will miss to mention many, but I will never forget them. António (my homonym), Henrique, Ana Paula, Ademar, Tânia, Pedro Abreu, Daniel, Vasco, Pedro Mendes, Célia, Joana, Luís, Pedro Faria, Pedro Moreira, José Moreira, João Neta and so many more. Thank you all.

I don't forget, in this extended acknowledgement, the students that I co-supervised and that helped me with work done and ideas for the future, namely Paulo Granja, Filipe Cruz and Pedro Valente.

I have to thank also my friends that don't belong to the academy. They are so special that even a million of words were not enough to express my gratitude. Their support to my living is incommensurable. I will never forget my friends from EFACEC and Siemens – without knowing them my life would never sound as it sounds now.

And, finally, there are no words to express the gratitude to my family (in this moment my tears felt). My mother and my father (I'm missing you) for their intelligence, simplicity, idealism and dedication, my brothers, my sisters in law, my nephews, my uncles, my cousins, my grandparents (I miss you too), my mother in law, my brother in law and, of course, Graça (my wife), Patrícia and Filipa (our two little pretty daughters) for the affection, love, support and understanding of my absences in games and on family outings. This thesis is also yours. You are my reason for living. I love you.

The research presented here was partially funded by the DITTY European project (EVK3-2002-00084) and by the FCT scholarship grant SFRH/BD/16337/2004, and I give my thanks to the National Foundation for Science and Technology (FCT) and to the taxpayers of this country, for keeping me fed for four years. I hope I start to return something back with these pages.

Agradecimentos

A escrita de uma tese proporciona uma oportunidade para agradecer a todas as pessoas que tocam a nossa vida pelo seu apoio, encorajamento, amizade, dedicação, tolerância e amor.

No meu caso, tenho de agradecer a muitas pessoas com quem partilhei momentos incríveis, experiências inesquecíveis, que me encorajaram e incentivaram para esta fantástica aventura de investigação, iniciada cerca de vinte anos após a licenciatura, e que me apoiaram para atingir esta meta na carreira profissional.

Antes de mais quero agradecer aos meus orientadores, professores Pedro Duarte e Luís Paulo Reis, pela sua amizade, pela sua crítica construtiva constante ao meu trabalho, pela sua total disponibilidade para clarificar as minhas dúvidas e a minha inexperiência, e pelo seu constante acompanhamento durante o difícil processo de escrita da tese. Nunca conseguirei exprimir por palavras o meu agradecimento à sua persistência comigo e sei que ganhei dois excelentes amigos para o futuro.

A ti, Pedro, obrigado pelo profissionalismo, alegria, conselhos, inspiração e permanente confiança no meu trabalho – tudo o que agora sei sobre biologia, ecologia, modelação e ciências naturais a ti o devo. Obrigado.

A ti, Luís, agradeço o teu acreditar no meu trabalho. Desde o início que nunca duvidaste da minha capacidade para a sua concretização, mesmo quando passei por momentos menos agradáveis. A tua imaginação, associada a um enorme ritmo de trabalho e vontade de vencer em todos os objectivos que sonhas, ultrapassaram sempre as minhas expectativas e colocaram-me, de novo, no campo de batalha. Obrigado.

Tenho de agradecer à Universidade Fernando Pessoa o acolhimento no início desta aventura e o apoio concedido para fazer o meu trabalho. Gostei mesmo de trabalhar com excelentes pessoas, nomeadamente Nelson Barros, Ana Fonseca, Maria João, José Torres, Álvaro Rocha e Lemos de Sousa. Nunca esquecerei os vossos gentis conselhos.

Os meus agradecimentos estendem-se, naturalmente, à Faculdade de Engenharia da Universidade do Porto (FEUP), em particular ao Laboratório de Inteligência Artificial e de Ciência de Computadores (LIACC), nomeadamente ao Núcleo de Inteligência Artificial Distribuída e Robótica (NIAD&R), pelo acolhimento, apoio e encorajamento que me proporcionaram, e pelo acreditar que esta tarefa teria fim. Tenho sorte em trabalhar no grupo

LIACC/NIAD&R e em colaborar com os seus investigadores, como o professor Eugénio Oliveira, o fundador e alma do laboratório, de quem tenho sempre muito a aprender. A sua crítica constante, às vezes provocadora, e conselhos incisivos obrigaram-me a reflectir continuamente no meu trabalho e a procurar constantemente identificar os seus aspectos inovadores.

Um agradecimento especial aos meus colegas e amigos que sempre me incentivaram na Universidade Fernando Pessoa, na Faculdade de Engenharia, no Instituto Politécnico de Viana do Castelo e no ISPGaya. Sei que me esquecerei de mencionar todos mas nunca os esquecerei. António (meu homónimo), Henrique, Ana Paula, Ademar, Tânia, Pedro Abreu, Daniel, Vasco, Pedro Mendes, Célia, Joana, Luís, Pedro Faria, Pedro Moreira, José Moreira, João Neta e tantos mais. Obrigado a todos.

Também recordo, neste extenso agradecimento, os estudantes que supervisionei e que me ajudaram com o seu trabalho e as suas ideias para o futuro, nomeadamente o Paulo Granja, o Filipe Cruz e o Pedro Valente.

Tenho de agradecer, também, aos meus amigos que não pertencem à academia. São tão especiais que nem um milhão de palavras conseguiriam exprimir a minha gratidão. O seu apoio no meu dia-a-dia é incomensurável. Também não esqueço os amigos que deixei na EFACEC e na Siemens – sem eles a minha vida não teria a mesma sonoridade.

E, finalmente, não há palavras para agradecer à minha família (neste momento as folhas ficam manchadas com as lágrimas). À minha mãe e ao meu pai (que saudades!) pela sua inteligência, simplicidade, idealismo e dedicação, aos meus irmãos, cunhadas, sobrinhos, tios, primos, avós (também sinto a vossa falta), sogra, cunhado e, claro, Graça (minha esposa), Patrícia e Filipa (as nossas filhotas) pelo afecto, amor, apoio e compreensão pela minha ausência nos jogos e passeios de família. Esta tese também é vossa. São a minha razão de viver. Amo-vos.

A investigação apresentada foi parcialmente suportado pelo projecto europeu DITTY (EVK3-2002-00084) e pela bolsa de doutoramento SFRH/BD/16337/2004 da Fundação para a Ciência e a Tecnologia (FCT) portuguesa, à qual agradeço, e aos contribuintes portugueses, o manter-me financiado durante quatro anos. Espero que este trabalho seja o início do retorno ao seu contributo.

Preface

"We don't have free will, but we do have free won't."

Gregory, 1990 (Blackmore, 2004: 129)

What is the motivation for pursuing a PhD degree? This is an interesting question that I recursively ask to myself and for which I don't have a single answer. And, more interesting than that, for someone who is an electrical engineer, with a large background in the supervisory, control, automation and informatics fields, and enjoys the study of life sciences and artificial intelligence, the answers are several times unrelated and opposite. I think that this is boosted by my personal experience made by seventeen years in industry, between the graduation and the beginning of the research that originates this document.

Sincerely I never thought that someday I would do a work like this. It was very far from my imagination and I have to confess that it was more laborious and inspirational than I thought. As an engineer, I force myself to obtain results in each task I put my hands (and arms and shoulders...), and to apply the results on the day life, and this document is a summary of the work done over the last six years, work made always with the intention of generating useful tools for the authorities and stakeholders to manage our coastal zone more effectively.

Contents

| | | |
|-------|--|----|
| 1 | Introduction and Objectives..... | 1 |
| 1.1 | Motivation..... | 1 |
| 1.2 | Objectives..... | 4 |
| 1.3 | Document Organization | 5 |
| 2 | Concepts and State-of-the-Art: Modelling and Simulation..... | 7 |
| 2.1 | Introduction | 7 |
| 2.2 | Models and Simulation | 8 |
| 2.2.1 | The Simulation Process | 11 |
| 2.2.2 | Continuous Simulation | 14 |
| 2.2.3 | Discrete Simulation | 14 |
| 2.2.4 | Statistical Simulation | 15 |
| 2.2.5 | Stochastic Systems | 15 |
| 2.3 | Ecological Modelling | 16 |
| 2.3.1 | Space and Time Resolution | 17 |
| 2.3.2 | Physical Processes | 18 |
| 2.3.3 | Chemical Processes | 20 |
| 2.3.4 | Biological Processes..... | 21 |
| 2.3.5 | Modelling Process in Ecosystems..... | 24 |
| 2.3.6 | Programming Languages in Ecological Simulation..... | 28 |
| 2.4 | Coastal Models and Simulators..... | 30 |
| 2.4.1 | Watershed Modelling..... | 32 |
| 2.4.2 | Hydrodynamic Modelling..... | 33 |
| 2.4.3 | Biogeochemical Modelling | 39 |
| 2.5 | Summary | 42 |
| 3 | Concepts and State-of-the-Art: Artificial Intelligence and Optimization..... | 43 |
| 3.1 | Introduction | 43 |
| 3.2 | Artificial Intelligence and “Fifth Generation” Models..... | 44 |

| | | |
|-------|--|-----|
| 3.2.1 | Agent Definition | 46 |
| 3.2.2 | Multi-Agent Systems Definition | 49 |
| 3.2.3 | Individual-Based Modelling | 55 |
| 3.2.4 | Agent-Based Modelling and Simulation | 56 |
| 3.3 | Intelligent Optimization Techniques | 60 |
| 3.3.1 | Metaheuristics..... | 62 |
| 3.3.2 | Individual Search Metaheuristics | 66 |
| 3.3.3 | Population Search Metaheuristics | 71 |
| 3.3.4 | Optimization Metaheuristics Discussion | 77 |
| 3.4 | Machine Learning Techniques | 80 |
| 3.4.1 | Ecological Modelling and Machine Learning Integration..... | 84 |
| 3.5 | Decision Support and Management Systems | 85 |
| 3.5.1 | DSS Origins | 86 |
| 3.5.2 | Architecture..... | 89 |
| 3.5.3 | Environmental Management | 90 |
| 3.5.4 | Multi-Criteria Analysis | 91 |
| 3.6 | Summary | 94 |
| 4 | Methodology and Implementation | 97 |
| 4.1 | Introduction | 97 |
| 4.2 | Generic Network Infrastructure | 98 |
| 4.3 | Simulation System..... | 100 |
| 4.3.1 | Model description | 101 |
| 4.3.2 | EcoDynamo..... | 104 |
| 4.3.3 | Library of Dynamic Linkable Objects | 110 |
| 4.4 | Multi-Agent Simulation System | 116 |
| 4.4.1 | ECOLANG | 118 |
| 4.4.2 | EcoSimNet Framework..... | 121 |
| 4.4.3 | Development Agent | 126 |
| 4.4.4 | Optimization and Machine Learning | 132 |
| 4.4.5 | Farmer Agent..... | 133 |
| 4.5 | Decision Support System..... | 146 |

| | | |
|-------|---|-----|
| 4.5.1 | Construction and Evaluation | 148 |
| 4.5.2 | EDSS Manipulation – AHP Methodology..... | 149 |
| 4.6 | Simulations..... | 152 |
| 4.6.1 | Sites Description..... | 152 |
| 4.6.2 | Scenarios Generation | 157 |
| 4.7 | Summary | 158 |
| 5 | Results and Discussion | 161 |
| 5.1 | Introduction | 161 |
| 5.2 | Realistic Simulation of Ecosystems | 161 |
| 5.3 | Flexible Simulation | 167 |
| 5.3.1 | Integrating Objects in Simulations | 167 |
| 5.3.2 | Integrating DLLs with legated platforms | 168 |
| 5.4 | Culture Optimization..... | 169 |
| 5.4.1 | Spatial Optimization..... | 170 |
| 5.4.2 | Carrying Capacity..... | 191 |
| 5.5 | Environmental DSS with AHP | 193 |
| 5.6 | Conclusions | 197 |
| 6 | Conclusions and Future Work | 199 |
| 6.1 | Summary and Conclusions | 199 |
| 6.2 | Future Work | 202 |
| | References..... | 203 |
| | Annex 1 – EcoDynamo User’s Manual | 223 |
| | Annex 2 – EcoDynamo Classes Diagram | 269 |
| | Annex 3 – ECOLANG Specification | 277 |
| | Annex 4 – Farmer Agent Configuration..... | 291 |

List of Tables

| | |
|--|-----|
| Table 2-1: Examples of aquatic ecosystem models published between 1988 and 2004 (Pereira et al., 2006)..... | 41 |
| Table 3-1: Table of relative scores for PCM matrix..... | 93 |
| Table 4-1: Generic DLLs included in EcoDynamo | 111 |
| Table 4-2: Summary of EcoDynamo forcing classes and simulated variables | 113 |
| Table 4-3: Summary of EcoDynamo physical and biogeochemical classes and simulated variables | 113 |
| Table 5-1: Relevant results (oysters: 5 cells – 1000 steps)..... | 173 |
| Table 5-2: Relevant results (oysters: 5 cells – 86 400 steps)..... | 174 |
| Table 5-3: Relevant results (oysters: 30 cells – 86 400 steps)..... | 176 |
| Table 5-4: Summary of bivalves monoculture experiments | 178 |
| Table 5-5: Summary of polyculture experiments..... | 179 |
| Table 5-6: Initial values for production indicators (P.I.) of oysters and scallops | 179 |
| Table 5-7: Results from monoculture experiments – oysters (60 cells, 86 400 steps – 1 month)..... | 180 |
| Table 5-8: Results from monoculture experiments – scallops (60 cells, 86400 steps – 1 month)..... | 181 |
| Table 5-9: Results from polyculture experiments – oysters (30 cells) and scallops (30 cells) [86400 steps, 1 month] | 183 |
| Table 5-10: Experiments extended to one year and a half | 187 |
| Table 5-11: Results from monoculture experiments – oysters (60 cells, 1 555 200 steps – 1.5 year) | 187 |
| Table 5-12: Results from monoculture experiments – scallops (60 cells, 1 555 200 steps – 1.5 year) | 188 |
| Table 5-13: Results from polyculture experiments - oysters (30 cells) and scallops (30 cells) [1555200 steps, 1 year and half]..... | 189 |
| Table 5-14: Carrying capacity experiments..... | 192 |
| Table 5-15: Simulation results for different clams' density | 195 |
| Table 5-16: Scenarios with indicators values | 195 |

| | |
|--|-----|
| Table 5-17: AHP scores under fixed and elastic prices with PCM A_1 , A_2 and A_3 | 196 |
|--|-----|

List of Figures

| | |
|--|-----|
| Figure 1-1: Population density - Portugal 2007 (INE, 2008: 44)..... | 2 |
| Figure 2-1: Simulation, Verification and Validation – adapted from (Sargent, 1991)..... | 9 |
| Figure 2-2: The Simulation Development Process – adapted from (Smith, 1998a)..... | 12 |
| Figure 2-3: The Simulation Development Process – adapted from (Law, 2007) | 13 |
| Figure 2-4: Biogeochemical cycle of nutrients in an aquatic environment (Jørgensen and Bendoricchio, 2001) | 22 |
| Figure 2-5: Modelling procedure for ecological systems - adapted from (Jørgensen and Bendoricchio, 2001) | 26 |
| Figure 3-1: Typical structure of a multi-agent system (Jennings, 2000) | 50 |
| Figure 3-2: General model for learning agent (Russel and Norvig, 2002)..... | 82 |
| Figure 4-1: Generic network integration infrastructure | 100 |
| Figure 4-2: EcoDynamo internal structure | 105 |
| Figure 4-3: EcoDynamo main window | 106 |
| Figure 4-4: Inheritance diagram for EcoDynClass | 112 |
| Figure 4-5: Generic agent model used in this work (Pereira et al., 2004b) | 118 |
| Figure 4-6: ECOLANG message structure | 120 |
| Figure 4-7: Example of ECOLANG messages (EcoDynamo side) | 121 |
| Figure 4-8: EcoSimNet architecture | 124 |
| Figure 4-9: EcoSimNet as a logic HLA architecture | 126 |
| Figure 4-10: Ria Formosa lagoon with customized colours adapted to B/W documents..... | 127 |
| Figure 4-11: 2D representation of the Alqueva dam (Alentejo-Portugal) 3D model..... | 128 |
| Figure 4-12: Sungo Bay 2D model with benthic species exploration areas | 128 |
| Figure 4-13: Zoomed area of the simulated model | 129 |
| Figure 4-14: A deeper zoom of the simulated model area | 130 |
| Figure 4-15: ECOLANG messages exchanged after connection to the simulator | 131 |
| Figure 4-16: System architecture for experimental Farmer Agent | 134 |

| | |
|---|-----|
| Figure 4-17: Farmer Agent optimization process - sequence diagram | 134 |
| Figure 4-18: FA optimization process – general sequence diagram | 135 |
| Figure 4-19: ECOLANG messages exchanged between FA and EcoDynamo during one iteration of the optimization cycle | 136 |
| Figure 4-20: EcoSimNet with "simulation islands" | 137 |
| Figure 4-21: Initial configuration of FarmerSA..... | 139 |
| Figure 4-22: Configuration of FarmerTabu | 140 |
| Figure 4-23: Configuration of FarmerGA..... | 141 |
| Figure 4-24: Configuration of FarmerRL algorithm with next solution options visible..... | 142 |
| Figure 4-25: Parallel SA algorithm (coordinator mode) | 145 |
| Figure 4-26: Parallel SA algorithm - worker node..... | 146 |
| Figure 4-27: DSS options available in Development Agent..... | 149 |
| Figure 4-28: Window to choose the type of value considered for each variable analysis | 150 |
| Figure 4-29: Pair-wise Comparison Matrix definition | 151 |
| Figure 4-30: Saving the results with one scenario name | 151 |
| Figure 4-31: Select files for DSS analysis | 152 |
| Figure 4-32: Geographic location of Ria Formosa and its inlets (Falcão et al., 2003)..... | 153 |
| Figure 4-33: Ria Formosa watershed (Falcão et al., 2003)..... | 154 |
| Figure 4-34: General circulation patterns during the flood and the ebb cycles. The vertical line separates the Western Ria from the Eastern Ria and the rectangles in red represent the two sub-domains considered (Duarte et al., 2005a) | 156 |
| Figure 4-35: Sungo Bay location, with model domain, bathymetry and part of the model grid - adapted from (Duarte et al., 2003)..... | 157 |
| Figure 5-1: Ria Formosa lagoon with locations of tide-gauge stations (triangles) used for hydrodynamic calibration | 162 |
| Figure 5-2: Predicted and measured velocities at Faro - Harbour | 163 |
| Figure 5-3: Predicted and measured velocities at Fuzeta - Canal..... | 163 |
| Figure 5-4: Ria Formosa lagoon with locations of water quality stations (circles) - WQA, WQB, WQC - used for biogeochemical calibration | 164 |

| | |
|--|-----|
| Figure 5-5: Simulated and observed ammonia at station WQB..... | 164 |
| Figure 5-6: Simulated and observed nitrate at station WQB..... | 165 |
| Figure 5-7: Simulated and observed phosphate at station WQB..... | 165 |
| Figure 5-8: Simulated and observed water temperature at station WQB..... | 166 |
| Figure 5-9: Predicted (line) and mean observed (squares) clams weight at two different points..... | 166 |
| Figure 5-10: Oysters growth in one location of Sungo Bay with (red line) and without (blue line) phytoplankton object included..... | 167 |
| Figure 5-11: Ria Formosa - value of light variables at 10 a.m. of July 1..... | 168 |
| Figure 5-12: Ria Formosa - value of light variables at noon of July 1..... | 169 |
| Figure 5-13: Ria Formosa - value of light variables at 6 p.m. of July 1..... | 169 |
| Figure 5-14: Experimental area and one admissible solution..... | 170 |
| Figure 5-15: Optimization results evolution during an experimental round, with visible gains when FarmerGa actuates – experiment DM::IO.60_sca (cf. 5.4.1 – Fourth experimental set))..... | 172 |
| Figure 5-16: Graphical representation of the 2 best (top) and the worst (bottom) solutions (oysters: 5 cells, 1000 steps – 8h20m)..... | 173 |
| Figure 5-17: Graphical representation of the best solutions (oysters: 5 cells, 86 400 steps – 1 month)..... | 175 |
| Figure 5-18: Graphical representation of the worst solutions (oysters: 5 cells, 86 400 steps – 1 month)..... | 175 |
| Figure 5-19: Graphical representation of the best solutions (oysters: 30 cells, 86 400 steps – 1 month)..... | 177 |
| Figure 5-20: Graphical representation of the worst solutions (oysters: 30 cells, 86 400 steps – 1 month)..... | 177 |
| Figure 5-21: Exploitation areas for the new experiments..... | 178 |
| Figure 5-22: Graphical representation of the best solution for experiment CM::A.60_oys (1 month)..... | 180 |
| Figure 5-23: Graphical representation of the best solutions for experiments DM::IO.60_oys and DM::BC.60_oys (1 month)..... | 181 |
| Figure 5-24: Graphical representation of the best solution for experiment DM::IO.60_sca (1 month)..... | 182 |

| | |
|---|-----|
| Figure 5-25: Graphical representation of the best solutions for experiments CM::A.60_sca and DM::BC.60_sca (1 month) | 182 |
| Figure 5-26: Graphical representation of the best solution for experiment DP::I.30_oys+O.30_sca (1 month)..... | 184 |
| Figure 5-27: Graphical representation of the best solution for experiment DP::IO.30_oys+IO.30_sca (1 month) – 12 cells with mix culture are represented by small grid | 185 |
| Figure 5-28: Graphical representation of the best solution for experiment CP::A.30_oys+A.30_sca (1 month) - 16 cells with mix culture are represented by small grid | 185 |
| Figure 5-29: Graphical representation of the best solution for experiment DP::BC.30_oys+BC.30_sca (1 month) - 14 cells with mix culture are represented by small grid | 186 |
| Figure 5-30: Graphical representation of the best solutions for experiments DM::IO.60_oys and DM::I.30_oys+O.30_oys (1.5 year)..... | 188 |
| Figure 5-31: Graphical representation of the best solutions for experiments DM::IO.60_sca and DM::I.30_sca+O.30_sca (1.5 year)..... | 189 |
| Figure 5-32: Graphical representation of the best solutions for experiments DP::I.30_oys+O.30_sca and DP::I.30_sca+O.30_oys (1.5 year)..... | 190 |
| Figure 5-33: Graphical representation of best solution for experiment DP::IO.30_oys+IO.30_sca (1.5 year)..... | 191 |
| Figure 5-34: Carrying capacity studies for oysters (density = 55 individuals/m ²) | 192 |
| Figure 5-35: Carrying capacity studies for oysters (density = 110 individuals/m ²) | 193 |
| Figure 5-36: Ria Formosa clam farming areas (white cells) | 194 |
| Figure 5-37: PCM matrices defined to AHP analysis | 194 |
| Figure 5-38: Graphical representation of AHP scenarios analysis: (a) fixed prices; (b) elastic prices | 197 |

1 Introduction and Objectives

1.1 Motivation

Human activities always had a prominent influence on the natural environment. Today, there is a growing awareness that the decisions and actions taken on the environment have consequences that can be very penalizing in the future, for the environment and for the living beings, particularly human beings. This is one of the reasons why environmental decision support systems began to have increasing interest in the scientific and business activities.

Coastal ecosystems always performed an important role in the life of human beings. They represent a small part of the area and volume of the oceans but, because they are the interface between land and sea, they allow an enormous amount of possible activities for work and leisure and guarantee several basic services to humanity such as food production, waste water treatment, shelter for harbour activities, leisure places, etc.

Over the last decades, human population migrated intensively from inland towards coastal boundaries and, nowadays, at least 60% of the human population lives within 60km from the sea (Watson et al., 1996). These numbers are also relevant in Portugal where almost 80% of the population lives within 50km from the sea (INE, 2008) - Figure 1-1.

Fourteen of the seventeen largest megacities are located along coasts. Coastal regions contain the lion's share of humanity's infrastructure. The activities of human society in industry, transportation and trade, energy processing, tourism, recreation, communications and services are all concentrated along coasts (Olsen et al., 2009). In recent decades, after some ecological

disasters, scientists and the various stakeholders are becoming aware that they must work together to ensure the appropriate management of those areas in order to maintain the overall quality of the ecosystems compatible with development strategies – that is what is commonly called Integrated Coastal Zone Management - ICZM (MAOTDR, 2007).

Aquaculture, touristic and harbour activities, urban development and human leisure converge to the coastal ecosystems, forcing loads of fresh water inputs, rich in organic and mineral nutrients derived from agricultural, urban and industrial effluents and domestic sewage (Duarte et al., 2007).

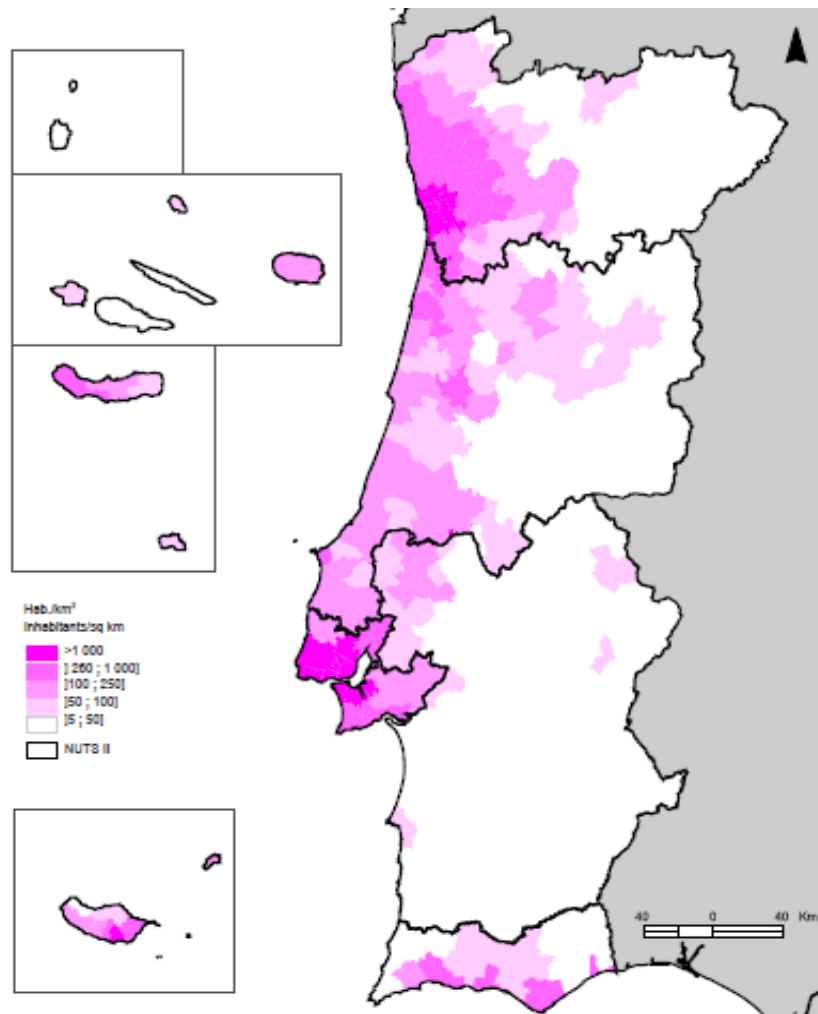


Figure 1-1: Population density - Portugal 2007 (INE, 2008: 44)

Coastal regions generate a disproportionate share of the global consumption of the manmade and natural resources, and the resulting generation of wastes. How humanity manages its activities and the impacts we produce on coastal ecosystems is one of the great challenges of the twenty-first century (Olsen et al., 2009). The sustainable development strategies for these ecosystems must include all the known interests in each region and should explain,

undoubtedly, why some actions must be forbidden, why some decisions must be taken and the benefits achieved by these options in the medium or long term for each ecosystem. These strategies may include short-term targets (when ecosystem threats may lead to immediate collapse in terms of ecological balance), but should be designed for future generations. The inclusion of the system stakeholders' in the decision-making process is crucial to ensure the commitment of the population to the decisions taken, even if their consequences may have some immediate negative effect on the day life of each community.

The use of simulation and ecological models is becoming a widespread tool in the management of coastal ecosystems towards a sustainable development strategy. Therefore, it is important to integrate human reasoning in the models, decisions and actions over the ecological systems. This task is very hard to include and none of the traditional simulation processes and models includes "human behaviour" because it is not easy to model, even if human decisions are limited by legal regulations and laws.

The fragmented legal framing of the administration responsibilities facilitates an overlap of competences between different entities, leaving a simple question like: "Is it advisable to enlarge an opened navigation channel (due to tourism pressure and boat navigation demands)?" without an immediate answer. A positive answer to the previous question raises another complex one: "What will be the future consequences and/or benefits?"

The huge number of possible combinations, generated by the different management decisions and options, the opposite interests of the stakeholders and the institutional authorities, and the slowness of the decision processes, increases the difficulty to implement automatic real-time management policies (Pereira et al., 2007).

The International Geosphere-Biosphere Programme (IGBP) launched in 1993 the Land-Ocean Interactions in the Coastal Zone (LOICZ) project (<http://www.loicz.org>), integrating scientists from across the globe to investigate changes in the biology, chemistry and physics of the coastal zone. The human dimensions were addressed in 2003, with the inclusion of research in areas like social, political and economic sciences. One of its objectives is the use of "research results to explore the role humans play in the coastal zone, their vulnerability to changing environments, and the options to protect coasts for future generations".

The European Union is contributing also to the LOICZ project and is being expend considerable financial and political effort in gaining a better understanding of the factors that affect European coastal waters, their interaction with river basins and the effect of land-use on

resulting fluxes of contaminants to coastal zones. The DITTY project was one European Project integrated in this effort and attained to develop the scientific and operational bases for a sustained and rational utilization of the available resources in Southern European Lagoons, taking into account all the relevant impacts from agriculture, urban and economic activities that affect the aquatic environment, by developing information technology tools tailored for these type of ecosystems (EC, 2003).

In the DITTY project, the participation of economists and stakeholders, in parallel with researchers specialized in the area, put considerable emphasis on the modelling phase as well as on detailed socio-economic assessment of management options. During the project implementation, it became evident the need to further explore the potential hybridization between ecological modelling and computer sciences, namely the provision of computer frameworks to integrate models developed by different research teams in different computer platforms and programming languages, exchange data generated by those models during the simulation phase, and develop the basis for common information technology tools and environmental decision support systems. It was the kick-off for the work presented in this document.

To corroborate this potential, during the first decade of the twenty-first century several journals appear in this hybrid area, emphasizing the integration of artificial intelligence techniques with environmental sciences – journals like “Ecological Informatics” and “Ecological Complexity” have joined the wide established journals edited by Elsevier, “Ecological Modelling” and “Ecological Engineering”, among others. Also the list of conferences around the environmental and sustainable development themes increased significantly, and every week one international conference happens anywhere in the world (<http://www.conferencealerts.com/environment.htm>).

1.2 Objectives

The main objective of this work is to develop an integrated computational system for modelling, optimizing and managing coastal ecosystems with the help of intelligent agents, designed to be used by the end users of the sites (non-experts of environmental sciences) and capable of realistically simulate complex systems, integrating other simulators in order to ensure compatibility and portability of models and processes.

For that, six specific objectives must be achieved by the research, answering the following questions:

- Is it possible to build a simulator for realistic ecosystem simulations, with the ability to include or remove sub-processes of simulation in accordance with the modelling goals?
- Is it possible to build a flexible simulator capable of simulating distinct ecological systems and/or distinct system characteristics with minimum change in the core simulation modules?
- Is it possible to integrate modelled processes, built in legated platforms, with new simulators and computer systems, sharing the mathematical knowledge of different research teams?
- Is it possible to optimize the simulation of complex ecological systems using intelligent agents and optimization and machine learning methodologies?
- Is it possible to combine the use of mathematical models with multiple and opposite decision criteria in the effective management of real coastal ecosystems?
- Is it possible to integrate all the previous into an harmonious intelligent simulation system, easy to extend and easy-to-use by the human decision makers?

1.3 Document Organization

This document is divided into six chapters and is organized as follows:

- In this chapter the work was introduced and its main objectives presented.
- The next two chapters present the background necessary to understand all the scientific and technological areas that are involved in this project. The state-of-the-art of ecological modelling and simulation is presented in chapter 2, agent-based simulation, optimization and machine learning techniques, and decision support systems are presented in chapter 3.
- Chapter 4 describes the methodology followed in this work, presenting the applications built to implement the ideas proposed in the project (simulator, visualizer, agents and decision support system (DSS)), the definition and the implementation of the framework needed to establish the network for the applications' communications (EcoSimNet platform and ECOLANG messages). The proposed DSS (based on the AHP methodology) is described more deeply as well as the strategy followed for the generation and the simulation of several different scenarios.
- Chapter 5 presents some results of relevant outputs generated by the simulation experiments and opens a discussion/analysis of the results obtained.

- The document ends with a chapter dedicated to the conclusions and to point for future possible developments.
- In the annexes the reader can find the user's manual of the simulator application, the detailed diagrams of the EcoDynamo simulation classes, the detailed description of the ECOLANG messages and protocol, and the description of Farmer Agent configuration files used in the work.

2 Concepts and State-of-the-Art: Modelling and Simulation

“The whole is simpler than the sum of its parts.”

Willard Gibbs

2.1 Introduction

It is difficult to date historically the beginning of the usage of models by humans – a simple sketch, or a scheme, with directions to some destination, may be seen as models. A road map may be seen as a more sophisticated model and, as a function of its scale, it could represent roads with more or less detail.

A model is always a simplification of reality and will “never contains all the features of the real system, because then it would be the real system itself” (Jørgensen and Bendoricchio 2001). For a model, to be understandable and useful, it must contain the characteristic features that are essential in the context of the problem to be solved, discarding those that will complicate the system without any advantages for the intended purposes.

Engineering is a wide field where models are fruitfully designed and applied. Each model has always one main objective. Therefore, many different models could be developed for the same subject, selecting the appropriate version according to the goals the modeller wants to achieve.

Models can be physical (miniaturized cosmos) or described by mathematical relations and equations resuming the main characteristics of the system and run by a computer program.

The rapid progresses achieved in computers' hardware and software development over the last decades have exponentially increased the usage of mathematical models across almost all fields of science, and simulation is now widely used to test or predict researchers' theories. This is particularly relevant in the fields of physical, chemical, biological and ecological sciences, engineering, health and weather forecast domains.

Following sections present basic concepts and state-of-the-art related with simulation, models and ecological modelling.

2.2 Models and Simulation

Human beings always tried to understand the environment and to explain the laws or rules governing its operation. While trying to explain the reality they realized that nature is very complex and difficult to explain as a whole. Not giving up the idea, they began by trying to understand simple phenomena, or those that seemed simple.

The first step began by the tentative to replicate natural phenomena, on a smaller scale, and then extract the understanding of its mode of operation – it was the advent of **physical models**. It was successful in many cases, unsuccessful in many more, but more complex processes of nature continued unexplained, because no physical model could be built to fit the objective. Over time, physicists and mathematicians proposed analytical expressions to explain some of those real processes. Simple systems of equations began to be constructed to represent the understanding of the processes and the knowledge acquired about them – it was the advent of **mathematical models**.

As presented in the introduction of this chapter, assumptions made about real or imaginary systems and their behaviour conducted the researchers to build mathematical models to understand how the systems work or to build new systems to do something imagined. Models have been constructed for almost every possible system - factories, communications and computer networks, integrated circuits, highway systems, flight dynamics, biological systems, national economies, social interactions, and imaginary worlds. In each of these environments, a model of the system has proved to be more cost effective, less dangerous, faster, or otherwise more practical than experimenting with real system. The process of designing models and conducting experiments with those models is called **simulation** (Smith, 1998b).

A model doesn't capture all the system reality and complexity, as it is impossible for a mathematical model to represent them. Typically, a model tries to describe, as simply as

possible, the processes under study, discarding all the characteristics that are not important for that purposes. The researcher adjusts the functions and **parameters** of the mathematical model until an acceptable degree of fidelity between the model and the system is achieved. This degree of fidelity is revealed by the results generated by the simulation model (Jørgensen and Bendoricchio, 2001).

As simulations are created from the minds of human designers, and even though every effort is made to ensure accuracy, compromises are always made and mistakes are inevitable. All simulations must be tested to establish their accuracy and appropriateness for the specific problems they have been developed to solve. Some authors (Sargent, 1991; Smith, 1998b) define this as a **three-phase process**: verification, validation and accreditation (VV&A). These phases are applied to the simulation development cycle and assume that the real world system to be replicated is identified, and a conceptual model of it is defined. The conceptual model is then encoded as a computer program. Sargent (1991) popularized the modelling process schema (Figure 2-1) where the three items (system, mathematical model and computerized model) form the vertices of a triangle where the VV&A phases are used to ensure that the transformation from one point to the next is accurate.

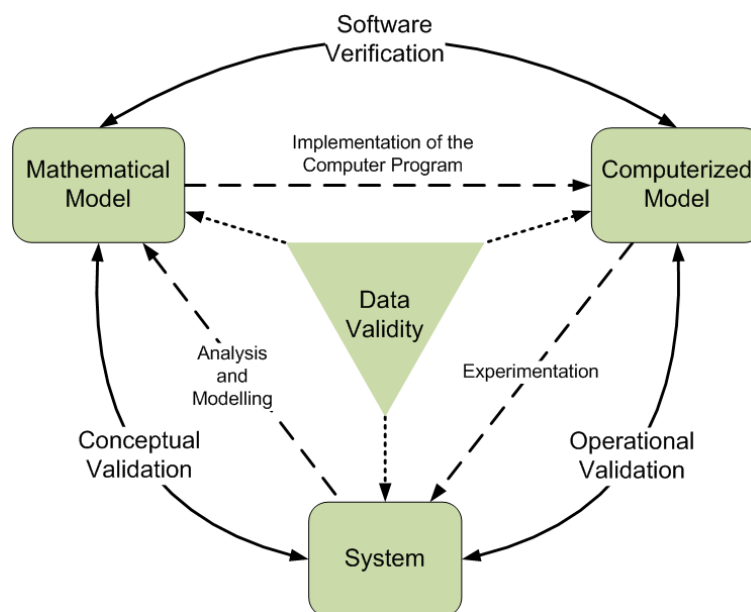


Figure 2-1: Simulation, Verification and Validation – adapted from (Sargent, 1991)

The **system** (real or proposed) to be modelled is the problem entity; the **mathematical representation** of the system is the conceptual model, developed for a particular study; and the **computerized model** is the mathematical model implemented on a computer as a program. The analysis and modelling phase helps in the construction of the conceptual model. The computer programming and implementation phase builds the computerized model. The

experimentation phase conducts computer experiments on the computerized model to obtain inferences about the problem entity and completes the feedback loop of the system.

Validation (or *conceptual model validity*) is the process of determining the extent to which the conceptual model is a reasonable representation of the proposed system. This phase is often described as answering the question: “Are we building the right product?”

Verification (or *computerized model verification*) is the process of determining that the software developed is an accurate implementation of the conceptual model. This phase is often described as answering the question: “Are we building the product right?”

Accreditation (or *operational validity*) is the determination that the simulation results are acceptable for the specified purpose over the domain of the model’s intended applicability. Each simulation addresses a specific class of problems and the solution obtained is only valid under the conditions imposed by those problems. This phase defines the set of problems for which the simulation is a good and useful tool.

Several versions of a model are usually developed, during the modelling process, in order to obtain a satisfactory valid model. The *data validity* represented in the figure is defined to ensure that the data necessary for model building, model evaluation and testing, and to conduct the model experiments to solve the problem, are adequate and correct.

Two fundamental questions are always faced by the computer scientists when they have to lead with a model: “how to describe the structure of a model”, and “how to present the results”?

The format of the results supplied by the model is of great importance and, in most cases, it is compatible with several commercial or open source software applications to manage the results and present the conclusions in an attractive and easy readable manner.

Simulations are often referred as either **continuous** or **discrete**. This discrimination is based on the manner in which the representation of the model (**state variables**) evolves. During continuous simulations the values of the state variables change continuously as time progresses. In discrete simulations the values of the state variables change when some event occurs, at distinct points in time (Smith, 1998b; Law, 2007). In practice, most simulations use both continuous and discrete state variables, and their classification is based on which one of these types is predominant.

As mentioned, a model only captures the important details of the real system or the system under study. The effects of the omitted details can be lost or aggregated into other variables included in the model. The inaccuracy introduced by either options must be evaluated and accepted by the developers. It can be the Achilles' heel of the model.

Another constraint for the application of simulation models could be the scarcity of input data to describe the behaviour of the system. This aspect is always addressed prior to the development of a model to reduce its impact when the model is completed.

These two limitations force the simulations to generate only approximate results and, most of the times, describe the system's behaviour statistically. Simulation is good to provide measurements of general trends, but not exact data for specific situations, turning it a valuable tool to help making decisions without the need of experimentation with the real system and without requiring the construction of the entire system. This is particularly important in the areas covered by the social sciences, where simulation tools are being used broadly to test new paradigms of relationships without real experimentation. Also in fields like nuclear energy, it is safer to analyze possible effects through simulation than with real experiments.

2.2.1 The Simulation Process

The last decades of the 20th century evolved the simulation art to an iterative process with feedbacks between the experimental and the modelling work (Figure 2-2).

Following the ideas documented by Smith (1998a), first it is necessary to explicitly **define** the **problem** that the model should address. The defined problem space fixes the objectives and requirements of the project and the required accuracy of the results. Also boundaries are defined between the problem and the environment. A model cannot be built without concrete definitions of hoped results.

The **definition** of one appropriate **conceptual model** includes the algorithms to be used that describe the system, inputs required and outputs generated. The limitations of the model, and assumptions made about the system, must be well documented and clearly defined to determine its appropriate uses. At this phase, all potential models are compared, until a single one is chosen that meets the objectives and requirements of the problem.

Once one solution has been chosen, it is necessary to **collect input data** to supply initial values for the parameters and information to the model. The data collected will also serve as valuable

information to validate the results of the simulations. It is, the most of the times, the most difficult task to achieve in the simulation process because it is the most prone to human error.

After that, the **software model** is **constructed** based on the solution adopted and the data collected. Mathematical functions and formulas are expressed in one computer language following the principles of good software engineering.

The phases that follow can be seen as the production phases: the **design experiments** phase must identify the most productive and accurate methods to run the simulations and generate the results; expensive simulation runs must be identified in order to avoid high costs and sparse results.

In the **execute simulation** phase, the simulations are performed to generate the output data. When the models follow Monte Carlo approaches, many hundreds or thousands of simulations must be done to collect statistically reliable results. While the model is executed, **output data** is **collected**, organised and stored.

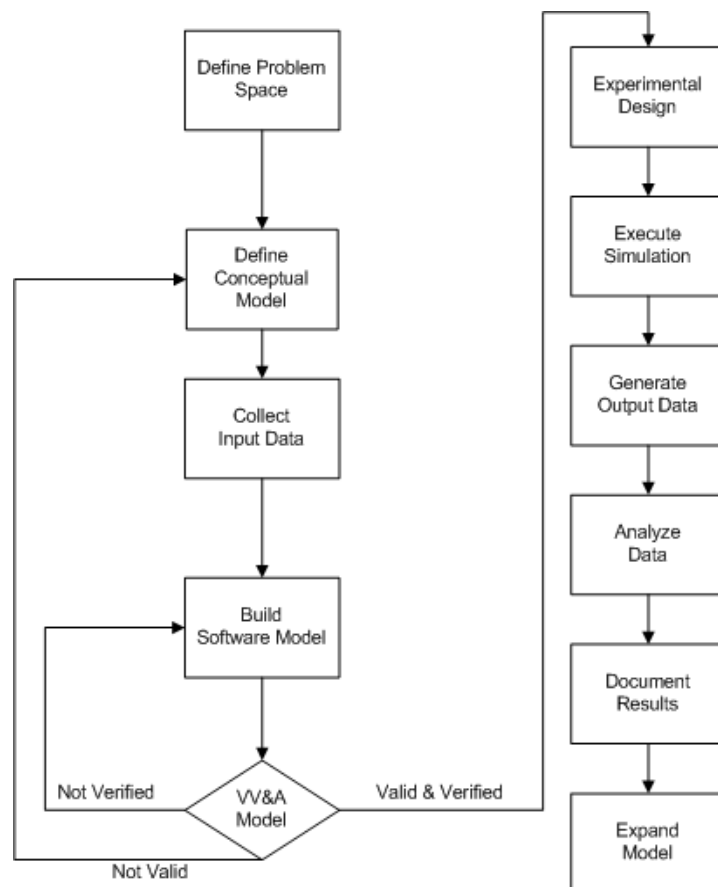


Figure 2-2: The Simulation Development Process – adapted from (Smith, 1998a)

The data collected during the execution phase are usually voluminous and distributed through time. Detailed **data analysis** must be performed to extract long-term trends and answers to

the questions that motivated the simulations. The information produced in this step can be combined in textual, tabular, graphical or animated forms. Modern applications and user interfaces have enriched the presentation of data to be easily understood by diverse audiences.

All the analysed **results** of the simulations must be **documented** and disseminated to interested parties in order to identify the fitness of the results with the expected answers to the initial requirements problem.

As simulation models are expensive and difficult to build, once a **model** is built it can be modified and **expanded** for use on related projects with new requirements, saving time and money.

With some slight differences, Figure 2-3 shows the proposal made by Law (2007) for the simulation process.

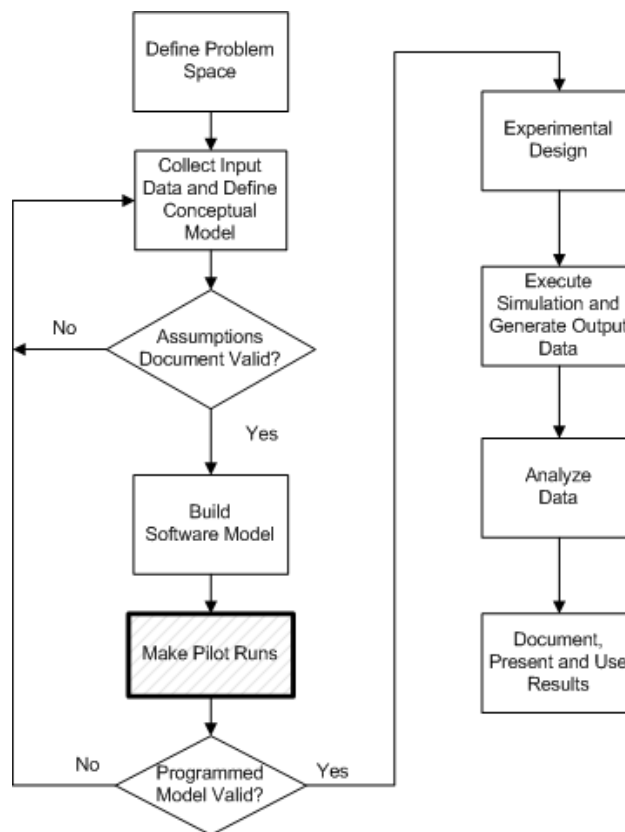


Figure 2-3: The Simulation Development Process – adapted from (Law, 2007)

In this diagram it is emphasized the need to run a prototype of the simulation model to produce results that must be checked for consistency with existing data for calibration. The acceptance of the simulation model prototype as valid occurs after a careful sensitivity analysis to determine the factors that influence the performance and results.

2.2.2 Continuous Simulation

Continuous simulation is used when the system is modelled over time and its representation made by state variables that change continuously as the time progresses. The simulation of continuous processes is usually represented by **differential equations** that reflect the rate of change of the state variables with time (Smith, 1998b; Law, 2007).

The initial values of the variables (value of the state variables at time 0) must be supplied to the model, in order to solve the differential equations and to give the values of the state variables over time. When analytical solutions for the differential equations are not possible, which is typically the case, numerical-analysis techniques are used to solve the differential equations.

There are many examples of this type of simulation. One of the most famous is the predator-prey model, also known as parasite-host or Lotka-Volterra (Volterra, 1926; Lotka, 1956), where two populations interact with each other and compete for survival in a region (Roughgarden, 1998). This is a very simple biological model where one biological species (prey) is passive and only eat, grows and reproduces itself, and the other (predator) depends on the prey as a food source: foxes-rabbits, sharks-prey fish, plant-herbivore, tumour cells-immune system, are some examples of these kinds of models.

2.2.3 Discrete Simulation

Discrete simulation is used when the system evolves over time and its representation is done by state variables that change instantaneously at separate points of time, caused by the emergence of events that change the state of the system. In this kind of simulation the modeller is focussed on the identification and description of the elements that characterize and are responsible for the system dynamics (Brito and Teixeira, 2001; Law, 2007).

The representation of the system is achieved by a set of **entities** that are responsible for **activities** implemented by the system. The activities represent the actions made by the entities to change the internal state of the system – the overall state of the system is the set of all states of the various entities that are part of the system.

This kind of simulations makes intensive use of concepts such as **events** and **queues of entities**. Each activity remains for a while and its conclusion will raise one or more events - one event is seen like one impulse that happens instantaneously and forces one system state transition. The system will rest in that state until a new event occurs. Events and activities are

two complementary views of the system dynamics. The queues of entities - typically implemented as waiting queues (first-in-first-out) or stacks (last-in-first-out) – are structures where each entity is placed while wait for the beginning of a new activity that will use it.

2.2.4 Statistical Simulation

When the complexity of the system is very high, and it is almost impossible to represent its behaviour by differential equations or through rules that describe its operation and activity, it is usual to apply statistical simulation on the system. This type of simulation does not operate on the model of the system but uses a set of **statistical distribution functions** to try to characterize the behaviour of the system (Johnson, 1987).

The mere fact that it is not mandatory the creation of a model for the system, makes the statistical simulation a paradigm very interesting among researchers, because the system is seen as a black box, eclipsing the internal processes, and the inputs and outputs are related through statistical distribution functions. This is a field where the Monte Carlo methods are used intensively (Law, 2007).

2.2.5 Stochastic Systems

There are simulation models where the values of some parameters are not constant over time – for example, the number of pedestrians, by minute, that wait for the green light in a semaphore to traverse one crosswalk in the city over a day. Even if a mean value could be calculated, it is interesting to simulate what happens in reality to reflect system dynamics. This can be achieved by replacing a fixed mean value of the parameter by a statistical distribution function that represents the arrival of the pedestrians to that crosswalk over the day. This method transforms the deterministic simulation process in a stochastic simulation process (Brito and Teixeira, 2001).

It is interesting to notice that the model in its conception remains deterministic, but the simulation process is stochastic. The stochastic behaviour of the parameters is included in the system through frequency tables, histograms or with the help of typical probability functions – for example, the Normal distribution, the Poisson distribution or exponential negative distributions are the most used (Law, 2007).

2.3 Ecological Modelling

Ecological models are simplified views of nature used to solve scientific and/or management problems. Ecological models only contain the characteristic features that are essential in the context of the problem to be solved or described. Ecological models may be considered a synthesis of what is known about the ecosystem with reference to the considered problem. As opposed to a statistical analysis - a model is able to translate the modeller knowledge about the processes of the system, formulated in mathematical equations, and component relationships and not only relationships between data (Jørgensen and Bendoricchio, 2001). For the same ecosystem there could be different ecological models, selected in accordance to the model's goals.

An ecosystem, following Jørgensen and Bendoricchio (2001), may be defined as:

“... a biotic and functional system or unit, which is able to sustain life and includes all biological and non-biological variables in that unit.”

The study of ecosystems involves many sciences and areas of knowledge because of its complexity and heterogeneity. It is difficult, or almost impossible, to capture all the relevant aspects that explain ecosystem functioning. To understand and model an ecological system as a whole, scientists consider that it must include, at least, three types of processes: physical, chemical and biological (Jørgensen and Bendoricchio, 2001).

In ecological models of aquatic systems, **physical processes** include flow and circulation patterns, mixing and dispersion of mass and heat, water temperature, settling of planktonic organisms and suspended matter, insulation and light penetration (Knauss, 1997; Jørgensen and Bendoricchio, 2001).

The **chemical processes** include hydrolysis, redox and acid-base reactions, adsorption and ion exchange processes. The simulation of these processes is very important for setting up a good model of a whole ecosystem and detailed descriptions of them are available and widely accepted by modellers (Jørgensen and Bendoricchio, 2001).

The **biological processes** include the biogeochemical cycles in aquatic environments, photosynthesis, algal, zooplankton, phytoplankton and fish population growth, and ecotoxicological processes (Jørgensen and Bendoricchio, 2001).

Chemical and physical processes are very well known when compared with the biological ones. The latter are much less established and, sometimes, only a rough description of their internal behaviour is known. The resulting model for an ecosystem always contains a trade-off between acceptable details of physical and chemical processes and a reasonable description of the bio-ecological processes.

Carefully designed models, which include important processes and components, still omit details that aren't important to the problem under consideration – many irrelevant details would cloud the main objectives of a model. However, these omitted details might have a strong influence on the predicted output those models produce (Scholten and Tol, 1998; Jørgensen and Bendoricchio, 2001).

2.3.1 Space and Time Resolution

For the modellers, one of the most important compromises is to find the optimal time and spatial scales of the model. Spatial grids acceptable for physical and chemical processes (10 to 100 metres) are very detailed for biological processes, and similarly, seconds, minutes or hours are good time scales for physical and chemical processes, but hours, days and months may be appropriate time scales for biotic components of an ecosystem (Jørgensen and Bendoricchio, 2001). Models with spatial resolution of a few square metres can be adequate for microbiologists but if the researchers focus on the study of large carnivores or nomad populations, a spatial resolution of thousands of square kilometres seems to be more adequate (Hutchinson, 1978).

The space division must account for variations in horizontal and vertical dimensions. 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) 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. Two examples of systems simulated by one-dimensional models are rivers (1D horizontal) and narrow deep lakes (1D vertical to study stratification).

When the system is large enough to present sensible variation of the properties, vertical and/or horizontal division is required and two or three-dimensional (2D or 3D) representations are more common. Models of deep large lakes, deep bays, dams or large river estuaries are examples of these representations.

Moving from 0D to 3D increases the morphological complexity of the model, and the model grid for an appropriate simulation of the physical processes.

2.3.2 Physical Processes

The principal physical processes in aquatic ecosystems include flow and circulation patterns, mixing and dispersion of mass and heat, water temperature, settling of planktonic organisms and suspended matter, insulation and light penetration (Jørgensen and Bendoricchio, 2001).

One of the most relevant physical processes is the **transport of mass** in the fluid media of ecosystems: air and water. This process influences the movement of nutrients, food and pollutants. It is important to know which concentration a substance can assume in a given place at a given time, and how a substance moves within the medium. This process is described almost by the same equations in each one of the fluids, with different values for the parameters that describe the substance moving in it.

Advection, diffusion and dispersion are the major processes of mass transport. **Advection** transports the substance solidly with the fluid in one direction without varying its concentration. The principle of conservation of mass applied to the fluid itself, or to a solute or suspended matter in the fluid, guarantees that the total mass of a conservative substance entering a fixed element of space in a given time must equal the increase in mass within the space, in that time.

Diffusion is the movement acquired to minimize concentration gradients in a fluid, from a region of high to a region of low concentration. It is accepted that horizontal mass diffusion in ecosystems is generally unimportant, but its mathematical formulation must be included because it constitutes the base for the turbulence in transport, which is much more related to ecological processes than horizontal mass transport. It also plays an important role in the vertical mass transport, for example, to explain the release of soluble substances from sediments.

Another important physical process, and a very primary principle of ecological modelling, is the **mass balance** – the fate of substances entering and leaving a system in various ways. The integration of the mass balance in ecological models is always simplified to make the system tractable. In aquatic systems one of the three assumptions is usually made:

- the system is completely mixed, **dominated by dispersion** and zero-dimensional, like a lake;

- the system is **dominated by advection**, like a river (substances entering a branch of the river are leaving it in the same sequence as they enter) or
- the system is **affected by both advection and dispersion**, like estuaries or coastal lagoons.

The **energetic factors** are the drivers for the evolution of ecosystems. *Solar energy* is the main source of energy for all ecosystems. It works as the most important forcing function for heat budget, photosynthesis, primary production and photolysis. It is well established the fate of solar radiation that flows through the atmosphere and reaches the earth's surface. Its intensity depends on the time of the year, the hour, the latitude of the place and the cloud cover (Brock, 1981). The equations that express the most important variables of this factor are globally accepted by the scientific community (Jørgensen and Bendoricchio, 2001).

Another important variable to the functioning of ecosystems is the daily solar exposure – **photoperiod** – which depends on solar declination. In aquatic ecosystems it is important to consider the **light extinction coefficient**, because what matters for water thermodynamics and for photosynthetic activity and, therefore, for primary production is the light that penetrates the surface of the water. The light that reaches the surface is attenuated as it enters the water, because the various dissolved and particulate materials scatter or absorb it. Thus, light intensity is a function of depth and water contents.

The **air and water temperatures** are other abiotic factors that drive the production of ecosystems. They are linked to solar radiation, but are influenced by factors like wind, humidity, cloud cover and pressure. At a seasonal time scale, temperature follows a deterministic behaviour driven by solar radiation, but at a short time scale of hours, days or weeks, temperature shows a stochastic behaviour driven by weather variations. The thermal capacity of the ecosystem (supplied by the large mass of air, water and land) contributes significantly to delay or reduce the influence of the variation in temperature due to solar exposure (Portela and Neves, 1994).

The physical processes that explain the movement of not dissolved particulate matter from the water column to the benthic species, and vice-versa, are called **settling** and **resuspension**. The physics of the settling phenomenon is described by classical mechanics, but the physiological state of phytoplankton, for instance, affects the sedimentation rate. Resuspension is the process that removes a particle from the sediment and moves it to the water body. It depends on factors like the sediment type, its grain dimension and consolidation state, the kinetic

energy in the water and the drag forces generated; the presence of biological material over the sediments increases their glue effect and difficult resuspension.

2.3.3 Chemical Processes

The principal chemical processes include hydrolysis, redox and acid-base reactions, adsorption and ion exchange processes.

It is important to notice that most **chemical reactions** occurring in the water ecosystems tend to be homogeneous – occur in a single phase (gas, liquid or solid). On the other hand, the time to complete one chemical reaction is much lower than the usual time step used to model ecosystem processes – so, the majority of ecological models deal with final steady-state **chemical equilibrium** of the system.

The **hydrolysis** processes proceed with water, hydrogen ions and hydroxide ions, and result in the introduction of a hydroxyl group OH^- in the structure of the compound. It is an important process of organic and inorganic compounds in the aquatic environments. In this group of processes are included the solubility of heavy metals and the organic pollutants reactions with water. The inclusion of the hydrolysis of some organic compounds requires dynamic models, instead of the steady-state approach of the chemical equilibrium, because the long half-time of the reactions with those compounds can be several times longer than the model time step.

The **redox** processes involves reduction-oxidation reactions with the participation of dominant inorganic ions present in the ecosystem. The reduction decreases the oxidation number of the component (molecule, atom or ion) and the oxidation increases the oxidation number. The existence or absence of oxygen in the aquatic environment can trigger redox reactions that modify the quality of the ecosystem – for example, one of the consequences of the eutrophication (anaerobic situation) in aquatic systems can be the release of phosphorous from sediments. Phosphorous in sediments is usually bound as iron phosphates; if the conditions are changed from aerobic to anaerobic, the anoxia caused by eutrophication will increase the phosphate available for algal growth (Jørgensen and Bendoricchio, 2001).

Because almost all processes in the environment depend on PH, the **acid-base** reactions are of great interest. Examples: the ammonia is toxic to fish and the ratio of ammonium to ammonia is known to be dependent on PH; the fertility of fish and zooplankton eggs is highly dependent on PH; the release of heavy metals ions from soil and sediments increases very rapidly with decreasing PH. Many models include computations and predictions of PH.

Adsorption is, by definition, a partitioning or separation process whereby a species (adsorbate) is transferred from the dissolved phase in a fluid solution onto the surface of a solid substance (adsorbent); this process may be explained by an electrical attraction to the solid surface of components with a minor electrical charge and by minor free energy of adsorbate compared to the adsorbent one (Jørgensen and Bendoricchio, 2001). Adsorption is usually coupled with **ion exchange** in the nature and a description of the processes is often included in water quality modelling.

2.3.4 Biological Processes

The biological processes are of great importance for ecological modelling and some are very complex.

The **biogeochemical cycles in aquatic environments** of macro constituents of organic matter are of great interest for ecological modelling, particularly the cycle of nutrients like carbon, oxygen, nitrogen, phosphorous and silicate.

Figure 2-4 was extracted from (Jørgensen and Bendoricchio, 2001) and represents the general biogeochemical cycle of nutrients in an aquatic environment. Compartments are indicated by boxes. Arrows show some of the processes moving nutrients from one compartment to another.

The **nitrogen cycle** is the most complex cycle of nutrients. Nitrogen cycle includes physical processes - reaeration (the passage of N_2 from air to water) adsorption/ desorption between sediment and water, and settling of particulate nitrogen - and chemical processes - mineralization of the organic form to the reduced inorganic form, the decay of organic matter via ammonia transformation and hydrolysis of the dissolved ammonia. One biological process typical of the nutrient cycle is the uptake of plants: ammonium and nitrate are taken up from plants in order to grow. In Figure 2-4 it is showed that the primary producers interact with the dissolved organic and inorganic via respiration and exudation, while the secondary producers interact via excretion. These last three processes are usually included in the nitrogen cycle model.

The **phosphorous cycle** is very similar to the nitrogen cycle. The most important processes in the phosphorous cycle are related with adsorption-desorption equilibrium between phosphorous in the sediment and in the pore water, followed by the diffusion to the water

column. As in the nitrogen cycle, the primary and secondary producers release the nutrient directly to the dissolved organic and inorganic pool via respiration, excretion and exudation.

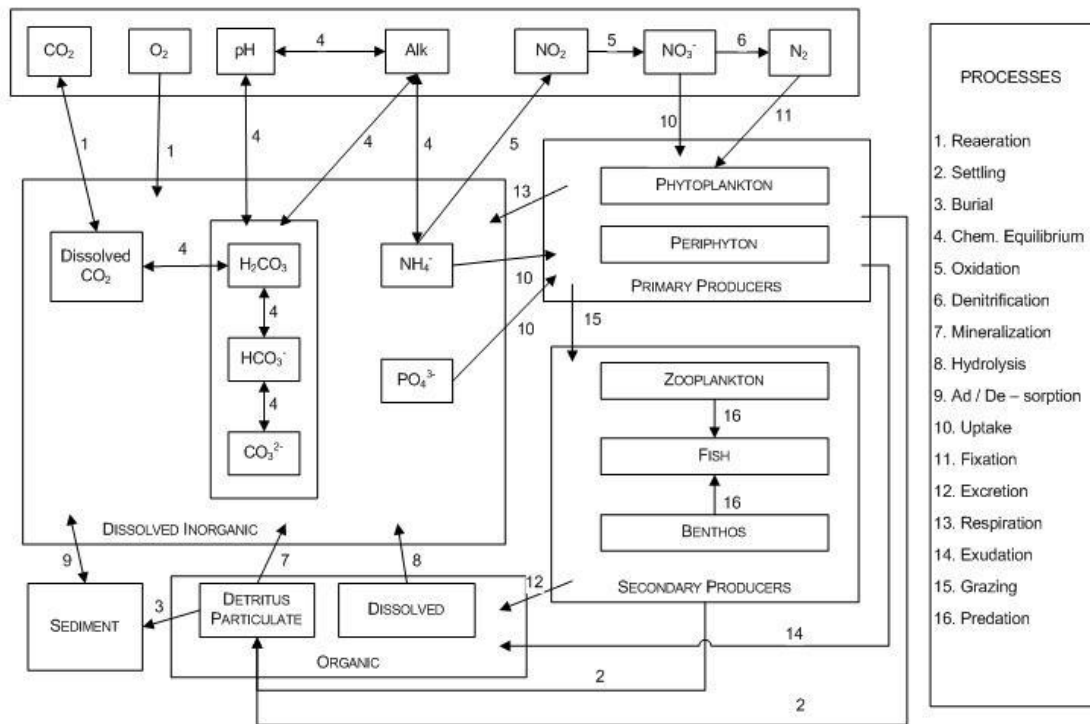


Figure 2-4: Biogeochemical cycle of nutrients in an aquatic environment (Jørgensen and Bendoricchio, 2001)

For biotic life the **oxygen** is very important; it cycles in the environment and enters processes of the other elements in consequence of chemical reactions, respiration and production by photosynthesis. In aquatic environments, oxygen is modelled as the concentration dissolved in water as gas, and the contribution to the general mass balance is guaranteed by the reaeration, consumption and production processes.

Different concentrations of oxygen in water and air raise the **reaeration** with a flow from air to water or vice versa, when the saturation in water is reached. The reaeration is a function of oxygen concentration in water, temperature, pressure and salinity, and also wind at the interface between water and air. As the temperature increases, the ability to maintain the oxygen dissolved in water decreases. When the oxygen concentration achieves very low values contemporaneously with high salinity and low pressures, it can be very harmful to the biota.

Another process entering the oxygen cycle is the **consumption**; it accounts for the microbial degradation of organic matter, which requires oxygen to oxidize all the reduced compounds that are the products of the general reaction of organic matter degradation. The consumption of oxygen in aquatic environments is mainly due to the respiration of primary and secondary

producers living in water, the oxidation of chemical compounds dissolved in water, the degradation of dissolved and suspended organic matter, the oxidation of nitrogen, and the oxidation of settled organic matter and respiration of benthic biota.

The **production** of oxygen is guaranteed by the photosynthesis process occurred in the phytoplankton. It is observed that the oxygen concentration is almost constant along the water column until the thermocline depth (about 5 metres). Below this depth the oxygen concentration drops quickly to low values reaching almost zero in the water-sediment interface, where the strong oxygen depletion is due to the anoxic sediments.

The **photosynthesis** is the key process in closing the cycles of oxygen and carbon, reducing the oxidized form of carbon (carbon dioxide) and producing oxygen. It represents the production of the biomass at the basic level of an ecosystem. Some external factors limit the photosynthesis: the availability of energy (light and temperature) and the existence of inorganic matter (carbon dioxide).

The growth of a population (algal, phytoplankton, fish, etc.) is always limited due to the availability of the resources in the environment: food, solar energy and space to grow. Of course, there are specific parameters for each species but those are the major constraints. For example, in the case of algal growth (**primary producer**) the general expression to calculate the variation of the biomass (measured by dry weight biomass or chlorophyll-a concentration or equivalent concentration of the more important nutrients) is function of the growth rate, respiration rate, exudation rate, non-predatory mortality rate, settling rate and loss due to grazing.

Ecosystems are complex systems where a food web can be identified. The primary producers of aquatic systems (for example, algae) are grazed by the upper levels of the web. To include secondary producers in the model it is advisable to simulate the long-term behaviour of the system. The zooplankton growth can be pointed as a **secondary producer** example. Simulating a complete biogeochemical cycle of zooplankton growth model includes the consideration of grazing and excretion processes relating nutrients cycle between algae and zooplankton, respiration, mortality and settling processes that transfer dead biomass to detritus, and the feedback of decomposition from detritus to nutrients.

In some models it is important to consider also the **ecotoxicological processes**. The biodegradation, the bioaccumulation and the equilibrium between spheres are the most common. The **biodegradation** is the biologically mediated conversion of organic compounds to

inorganic compounds. The **bioaccumulation** is described as the concentration of one compound in one of the spheres (atmosphere, lithosphere, hydrosphere and biosphere). There are processes that explain the transfers of components or elements between the spheres until **equilibrium** is achieved. Observation of trends in global changes of concentrations in the spheres might be of great importance as they may cause changes in life conditions on earth.

2.3.5 Modelling Process in Ecosystems

The modelling process in the environmental sciences needs, clearly, five components that must always be identified and defined: **forcing functions**, **state variables**, **mathematical equations**, **parameters** and **universal constants** (Jørgensen and Bendoricchio, 2001).

Forcing functions, or **external variables**, are functions of an external nature that influence the state of the ecosystem. If the modeller has control over these functions they are called control functions. For instance, in a model that includes eutrophication the inputs of nutrients can be a controlled function. The influence of tide in coastal ecosystems or climatic functions, which influence the biotic and abiotic components of the system, can be viewed as not controllable forcing functions.

The state of the ecosystem is described by the value of the **state variables**. The selection of these variables is, most of the times, obvious when the model is structured. In the eutrophication models the concentrations of nutrients and phytoplankton are fundamental variables. If the model is used for predicting the bivalve production, variables that reflect the biomass and the growth rate of the bivalve species are essential. If the model is used to ensure that the water quality of a lake or river is in accordance with the Water Framework Directive (EC, 2000), ammonia concentration is one of the obvious variables to consider.

The physical, chemical and biological processes are represented by **mathematical equations** that describe the relationships between the forcing functions and the state variables. If the same process exists in different contexts, it is natural that it can be described by the same equations in several models. But this may not always be true if the models have different levels of detail, leading to differences in the level of complexity of mathematical equations; on the other hand, there may be interest in the inclusion of other factors to help describing the behaviour of the process.

The **parameters** are coefficients in the mathematical representations of the processes. In a specific ecosystem their values are considered constants but in the literature many parameters are indicated as ranges instead of constants. This is particularly true for the biotic parameters –

for example the growth rate of a bivalve species. This is driven by the limited knowledge that exists for the values of the parameters in the biological processes; even if the parameters could assume constant values, in some ecological models that would be unrealistic because it does not consider the feedbacks of the real systems – the flexibility and adaptability of the ecosystems is inconsistent with the application of constant parameters in some models. Nowadays, some models attempt to use varying values for the parameters according to some ecological rules, in order to reflect this feedback influences.

Almost every model uses **universal constants**, such as gas constants, atomic weights, and gravity, latitude or longitude values.

One proposal for the modelling procedure of ecological systems was summarized by Jørgensen and Bendoricchio (2001) and Figure 2-5 reproduces the approach. In this summary it deserves emphasis the division in phases (observations, brainstorming, modelling, and management) that can facilitate the perception of the whole procedure and distinguish the intellectual effort of each step.

The first step is always the **definition of the problem**, and the bounding in time, space and subsystems. After grasp the big picture, it is important to identify the dynamic processes and their causal relationships. The complexity of the system should be well thought – sometimes a simple system is more accurate to translate the reactions of the real system than a complex one.

The **quantity and quality of the available data** is closely linked with the success of the calibration and validation of the model, and this is not a task easy to guarantee. The ideal would be to have real data available for all the selected state variables calculated by the model.

Another step that can be helpful for the conceptualization of the complexity of the model would be the **adjacency matrix**. It should be a simple matrix where all the state variables are placed and where it is indicated if there is a direct relationship between two variables. The matrix will assist the construction of the **conceptual model**. In practice, great part of the models is designed after the data collection, but ideally the conceptual model should determine what data are needed to develop the model.

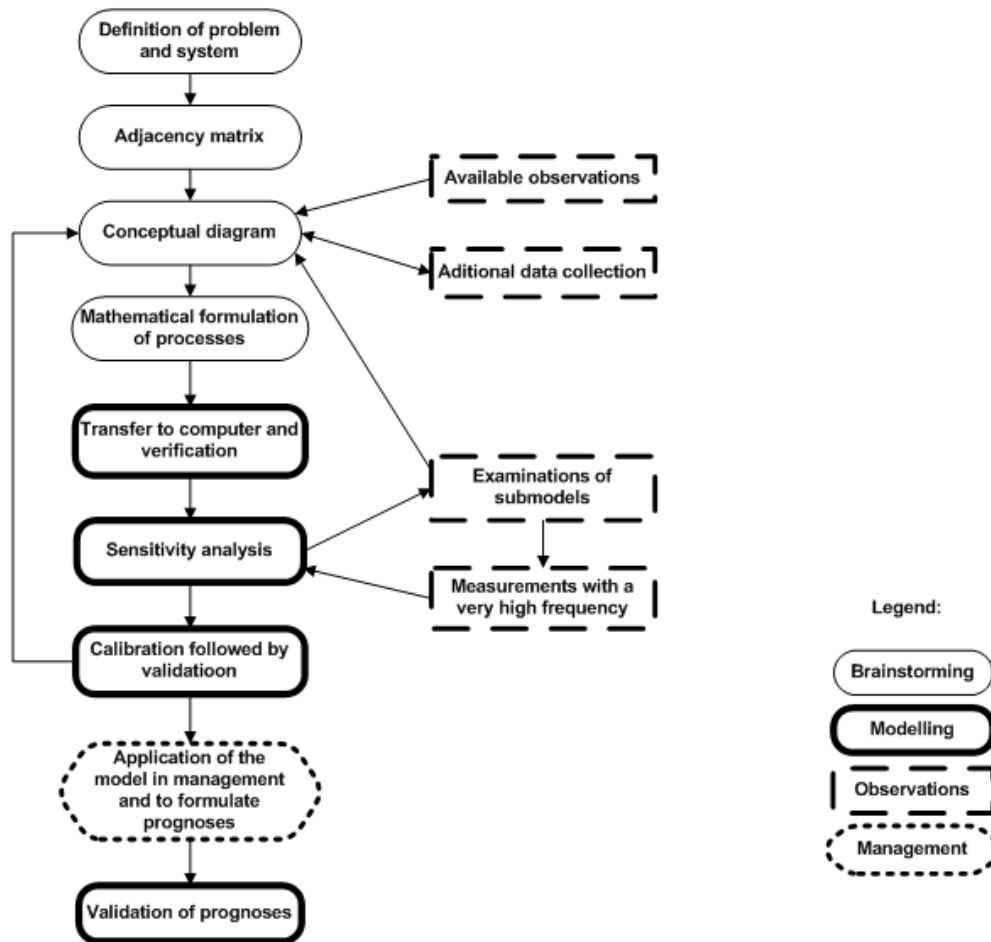


Figure 2-5: Modelling procedure for ecological systems - adapted from (Jørgensen and Bendoricchio, 2001)

Many processes could be enunciated by more than one **mathematical equation** and it is important to select the right ones to obtain the best results of the model. After the programming of the mathematical formulation, it is important to do a previous **verification** to the results shown by the model. It should be checked that the model is stable in the long term and react as expected.

The **sensitivity analysis** of the model is carried out by the modeller changing the parameters, the forcing functions or the submodels, and observing the variations in the values of the most important state variables of the system. The objective of this analysis is to get one overview of the most sensitive components of the model. A good knowledge on the certainty of the parameters and forcing functions is a providential aid to the **calibration** phase (crucial step for the model acceptance), where the parameters estimation is improved in order to validate the results of the model. The **validation** is guaranteed testing the fitness between the output generated by the simulation model against an independent set of collected data. This fitness is

indicated by some validation criteria, and while it is not achieved the process must feedback to the conceptual model and restarted from there.

When the model is validated, it is ready to be **used by the community** to do research, make prognoses or help the ecosystem management.

The modelling procedure described in the preceding paragraphs is similar to the ones presented in the section 2.2.1 , where it was referred that all modelling processes must agree with VV&A phases. The accreditation phase implies that the researchers' community accept the model and the simulations as good representations of the real system for the proposed objectives. Some authors refer three different significant steps in the modelling procedure – **verification**, **calibration** and **validation** (Jørgensen and Bendoricchio, 2001). These steps are similar to the former ones with slight differences - verification tests the internal logic of the model; calibration changes the values of the parameters belonging to the mathematical equations to adjust computed and observed data; validation performs objective tests over the model to see how outputs fit the collected and observed data.

These considerations are particularly relevant regarding ecological models, where the biological and some chemical processes have a large uncertainty associated with equation's parameters. After implementation, a first round of simulations is necessary **to verify** the internal logic of the model. Afterwards it is necessary **to calibrate** the model, i.e. make a second round of simulations to tune the internal model parameters in order to reproduce the observed data. This is a hard and tedious work requiring expertise and a very good understanding of the effects of the different parameters over the available variables (Scholten and Tol, 1998). When the interaction between the processes is intensive (like in the ecological models) this task is very time consuming. After calibration, the model is submitted to a third round of simulations **to validate** it, comparing the results generated with observed values that were not used in the calibration phase. Once the model is validated, it can be used as a predictive tool and simulations may be set up depending on the purposes for which the model was developed.

The *calibration phase* of the model consumes a lot of man power and it is modeller dependant. However there are automatic calibration procedures, based on systematic and exhaustive generation of parameter vectors and using some convergence and validation methods, like Controlled Random Search (CRS) (Scholten and Tol, 1998; Scholten et al., 1998) or linear regression techniques (Mesplé et al., 1996). These techniques require a large number of model runs, are computational intensive, and are not efficient when applied to complex systems.

Another approach can be followed - the development of a self-learning tool that implements the learning process of the modeller with machine learning techniques – the Calibration Agent (Pereira et al., 2004a).

2.3.6 Programming Languages in Ecological Simulation

Aquatic ecosystems modelling and simulation is a field where almost every programming language has representatives. Several modelling tools have been developed for the simulation of hydrodynamic and biogeochemical processes in aquatic ecosystems. Until the end of the 1970s, coupling hydrodynamic models to biogeochemical models was not common, and today problems linked to the different scales of interest remain. The time scale of hydrodynamic phenomena in coastal zones (seconds to hours) is much lower than that of biogeochemistry (few days to months). Over the last years, there has been an increasing tendency to couple hydrodynamic and biogeochemical models in a clear recognition of the importance of incorporating in one model the feedbacks between physical, chemical and biological processes.

Modern ecological models link together the physical, chemical, biological and ecological submodels simulating at different scales space and time steps, and even different types of models (static, dynamic or structurally dynamic models).

First ecological models were simulated based on structured programming: each application consists of a main program, where some global state variables describing the ecosystem under simulation are defined, and calls are made to several sub-routines, at each model time step. These subroutines calculate all processes represented in the model and the fluxes that affect each state variable. At the end of each simulation cycle, all state variables are updated as a function of the mentioned fluxes. In some cases, sub-routine calculations depend on general scope state variables that are defined and calculated elsewhere in the code. When this happens, it is difficult to reuse these subroutines in other source codes. In addition to these limitations, and depending on the compilers used, it is always a complex task to combine subroutines written in different source codes to build new models (Pereira et al., 2006).

With the birth of the object-oriented programming (OOP) languages, a new simulation style has emerged: the idea is to define objects representing the real life objects. Several groups of variables and processes, that may correspond roughly to some subroutines of the previous approach, can be aggregated in classes that generate objects in runtime, with no need to use “global” state variables because each object should handle its own variables and should be

able to represent itself and to control its inputs, outputs and inner behaviour. The advantage here is that, by eliminating global scope variables, it is easier to avoid programming errors, when a variable is changed at several different sites in a program. Apart from that, objects have several properties inherent to the OOP approach that make them very useful in ecological modelling: encapsulation, inheritance and polymorphism (Weiss, 2000; Lippman et al., 2005).

Encapsulation forces an object to reveal only the interfaces needed to interact with it providing public interfaces to the outside world; details that are not pertinent to the external use of the object should be hidden from other objects, preventing inadvertent usage that may corrupt its internal state. Moreover, the object implementation is separated from the interface and may evolve over time, in response to new knowledge or requirements, without requiring changes in its interface. This information hiding is the first step to modularity: it strengthens the existence of independent self-contained modules, which make their reusability easy in other applications.

The concept of **inheritance** was directly borrowed from biology – it is a feature that represents the “is a” relationships between different objects, allowing an object to inherit the attributes and methods of another object, providing support to extend or tailor new behaviours in derived objects maintaining the base behaviour of the parent - it is like borrow a group of existing attributes and methods from an existing object instead of re-inventing the wheel. For example, dogs and cats look different, have different behaviours and characteristics but have many things in common because both are mammals. The mammal properties could be placed in the mammal class, and dogs and cats just add their own features – it is not necessary to repeat the mammal attributes and behaviour.

Polymorphism allows two or more objects to respond to the same method in different manners – the “invoker” object communicates with different objects in a consistent manner without worrying about how many implementations of that method exist. Derived objects behave differently from the base object calling, at runtime, the appropriate method depending upon the type of object instantiated – different objects handle different types of data using an uniform interface. These features facilitate the building of more concise code and reduce the development cycle of a program.

One example of OOP approach, and moving again to aquatic ecosystems modelling domain, is briefly explained: one may define a phytoplankton object that has a set of parameters and methods to change its variables (e.g. biomass) by simulating relevant physiologic processes. If

the implementation of a particular model needs the definition of new phytoplankton species, differing from the existing object in the values of some physiologic parameters and behaviour, and everything else being equal, it is not necessary to repeat the code or variables from the existent phytoplankton to each new species – only the new parameters and corresponding behaviour should be included. With polymorphism, several dynamic instances of the phytoplankton species may be created, each with their own parameter values and corresponding behaviour while sharing the common parameters and behaviours. With this approach it is easier to build models of different complexity. Furthermore, if it is necessary to change the behaviour for one of the modelled species, a descendant object may be created inheriting all from its ascendant and overloading only the methods that produce the desired behaviour.

It is important to notice that the need to manage the model time step in each simulation remains unchanged with any of the methodologies described.

2.4 Coastal Models and Simulators

The coastal ecosystems science can be seen as the study of the inter-relationships existing in coastal ecological communities. To protect and manage the coastal resources effectively, it is essential to integrate the human activities, living organisms, physical and bio-chemical processes and natural phenomena.

Models of aquatic ecosystems include biogeochemical processes, such as photosynthesis, nutrient cycling and grazing, and transport processes. The former are responsible for local changes of state variables, such as concentration of chemical constituents and biomass of different species or groups of species. The latter are responsible for the transport of pelagic variables across model domain and boundaries.

The art of modelling is to make a model which includes all the properties in study, but no more than that. This is quite an intricate task and that's why there are so many different research teams, all over the world, designing different models for identical systems. Different models may include different processes, described on a different way with more or less detail. The degree of detail may be determined by the importance assumed for the different processes in a particular ecosystem and also by the available knowledge. For example, some mathematical models can use a simplified description of hydrodynamic transport processes but a very detailed description of benthic biologic processes (Baretta and Ruardij, 1988), whereas other models may include very deep hydrodynamic details and relatively simple ecosystem biological

model (Luyten et al., 1999). Even when such a basic, important and well established process as photosynthesis is considered, it is difficult to find many models using exactly the same approach – several different mathematical formulations may be used to describe the relationship between light intensity and production (Duarte, 2005).

In the last 40 years many theoretical studies have been developed to attempt to produce general rules for application in modelling of ecological systems with biological organisms. The Dynamic Energy Budget (DEB) theory (Kooijman, 1993) is a prominent theory on the way organisms manage their energy - this formal theory analyses the food and nutrient requirements organisms have, and how they incorporate the energy uptake and the use of substrates (food, nutrients, light) for maintenance, growth, maturation and reproduction. DEB theory is based in chemical and physical principles, says something about the energy that comes into and goes out of an organism, and it includes effects of temperature and chemical compounds in the microorganisms, animals and plants. For example, aging is presented as an effect of oxidation reactions in biological species. In practice, many well-known empirical models are included as special cases of the DEB theory.

One interesting view of biological organisms in the coastal areas was presented by Dumbauld et al. (2009) where bivalves aquaculture is viewed as a disturbance which modifies the estuarine system in three ways:

- Changes in material processes – bivalve process food and produce wastes;
- Addition of physical structure – aquaculture introduces the cultured organisms and physical anchoring structures;
- Pulse disturbances like harvest and bed maintenance disturb sediments – remove species in addition to the cultured organisms themselves, and change resource or habitat availability.

Different modelling teams tend to adopt different modelling tools. First approaches, and perhaps the most spread, make use of structured programming – the EMS/Dollard estuary model described by (Baretta and Ruardij, 1988) and the 3D Marine Coastal COHERENS model described by (Luyten et al., 1999) are two examples of this methodology. However, nowadays, the object-oriented software is gained prominence and is attracting more and more research teams – EcoWin2000 (Nunes et al., 2003; Nobre et al., 2005; Ferreira et al., 2008), MOHID (Miranda et al., 2000) and EcoDynamo (Pereira and Duarte, 2005); EcoWin2000 and EcoDynamo are written in C++ and recent modules of MOHID are written in object oriented Fortran-95.

What is the best approach? Well, it is not easy to perform benchmarking exercises with different models because of two main reasons (Pereira et al., 2006):

- There is not one modelling software suitable to solve all simulation challenges.
- Different research teams generally are very reluctant to work with other modelling software than their own, for a particular project, because of the time usually needed to learn, understand and handle a new modelling tool.

The obligation of Environmental Impact Assessments (EIA) in the current legislation of many countries, particularly in the European Union, is increasing the use of models in the environmental sciences. Some European Union Framework Directives - like the Water Framework Directive (EC, 2000) – already predict its usage, and it is natural that more environmental laws, in each country, begin the imposition of reference terms regarding the way different processes are simulated for aquatic ecosystems. This will enforce different research teams to work together, because EIA will be larger and complex, and will require multidisciplinary skills to be joined in order to obtain reliable results. Some research teams are actually working in integration tools and methodologies (Pereira et al., 2006) to allow modellers from different teams to share code, based on object oriented programming approach.

One simple presentation of watershed, hydrodynamic and biogeochemical modelling (three fundamentally different realms of modelling) follows.

2.4.1 Watershed Modelling

The description of the water movement in a river basin, both in terms of hydrological cycle and hydraulic routing, is the scope of the watershed modelling (Chapelle et al., 2005b). Many of these models also include the transport of the constituents, precipitation, dissolving, dispersion and chemical reactions, and changing concentration of the constituents as water flows along. Many of them accounts realistically for detailed watershed morphology, land use and land cover distribution, point sources and water protection/management structures (pumping, channels of circulation, artificial impoundments).

To provide real-time hydrologic forecasting, support watershed analysis, planning develop total maximum daily loads (TMDL), predict water quantity and quality planning and pollution control, many simulation models of such type are described in the literature (McDonald and Harbaugh, 1984; Haith and Shoemaker, 1987; Evans et al., 2002; Neitsch et al., 2002b; Smith et al., 2004; Chapelle et al., 2005b). Some models are highly specialized to simulate

environmental phenomena and components of pollution problems – predict localized pollutant transport – others incorporate more comprehensive assessment techniques – simulate pollutant loading, transport and transformation.

The watershed models are used to provide forcing inputs to hydrodynamic and biogeochemical processes in coastal lagoons, and a model intercomparison analysis can be found in Chapelle et al. (2005b).

2.4.2 Hydrodynamic Modelling

Hydrodynamic models are tools built to describe and forecast fluid dynamics and related variables for a given aquatic environment – ocean, continental shelf, lagoon, etc. The base for the mathematical modelling of geophysical fluid dynamics are the Navier-Stokes' equations, that are non-linear partial differential equations very difficult to solve. In the field of oceanography, mathematical models often use a simplified form of the equations, known as Boussinesq equations (Chapelle et al., 2005b).

Reports referring three-dimensional (3D) hydrodynamic coastal models appeared in the late 70s. The increasing availability of computer resources needed to run the models, triggered recent developments and widespread applications of Boussinesq-type equation models for studying wave propagation. The equations derived by Peregrine (1967) are often referred to as the standard Boussinesq equations - they describe the nonlinear transformation of irregular and multidirectional waves in shallow waters.

The main differences between available three-dimensional hydrodynamic models rely on (Pereira et al., 2006):

- The representation of the vertical dimension;
- The numerical treatment given to turbulence;
- The numerical methods used to solve the partial differential equations, such as finite differences or finite elements methods.

The representation of the vertical dimension can use space transformation ("sigma models") or work without any space transformation ("Z coordinate models").

The "sigma model" use the transformation of the vertical dimension developed by Phillips (1957) for meteorological models - the varying vertical coordinate in the physical domain is mapped to a uniform transformed space where sigma (σ) spans from 0 to 1. These models,

defined as “terrain-following” models, consider a flat bottom where boundary conditions are applied. The general used transformation is:

$$t = t^* \quad x = x^* \quad y = y^* \quad \sigma = \frac{z^* + h}{D} \quad (1)$$

Where:

- (x, y, σ, t) are the space and time coordinates in the sigma coordinate system;
- (x^*, y^*, z^*, t^*) are the space and time coordinates in the original coordinate system;
- D is the total water depth;
- h is the depth measured from $z^*=0$.

The major advantage of this system is the constant number of vertical layers over the whole computational field. One major disadvantage of this coordinate system is that σ will change with time if the total depth changes with time. Inverse transformation is needed for the calculation of the horizontal velocities or other variables of the model.

An intercomparison exercise on several water circulation models applied to the Mediterranean Sea, reported in Beckers et al. (2002), revealed that no model performed better than the others and that there was a similar correlation between model characteristics and modeller's skill in terms of results. A small presentation of six different hydrodynamic models follows.

COHERENS

The COHERENS model was developed by the Management Unit of the North Sea Mathematical Models (MUMM), Belgium, within the scope of the European Union Marine Science and Technology Programme during the 90's (Luyten et al., 1999) and made available for the scientific community. COHERENS is a 3D hydrodynamic model solving the Boussinesq equations with the hydrostatic assumption on a sigma space. COHERENS is also a model providing great care to solve turbulent phenomena and it proposes numerous different turbulent schemas. Special subroutines cope with the application of boundary conditions. COHERENS includes also other facilities, such as particle tracking, sediment movements as deposition and erosion of suspended organic and inorganic material, biogeochemical sub-models, etc.

The COHERENS software solves an advection-diffusion equation of the form:

$$\frac{\partial Y}{\partial t} + \nabla(uY) + \nabla(S_Y Y) = [P(Y) - D(Y)] + \nabla(v \nabla Y) \quad (2)$$

Where:

Y is a given scalar – temperature, salinity or a biogeochemical state variable;

u is the velocity vector;

S_Y the specific movement of Y ;

v the parameter for diffusion;

$P(Y)$ and $D(Y)$ are any biogeochemical production or destruction terms, respectively.

More information on equations can be found in (Luyten et al., 1999).

COHERENS includes also a biological model characterizing primary production for a North Sea application and for Sacca di Goro lagoon, Italy (Chapelle et al., 2005b). The code, written in FORTRAN, is in the form of a collection of subroutines called by a main program, and is freely available from the COHERENS home page at the MUMM website (<http://www.mumm.ac.be/EN/Models/Coherens/>), with a complete documentation of more than 900 pages, describing all aspects of physics and the code, as well as some examples of uses including input and output files and a makefile skeleton. COHERENS model is a powerful tool to understand the physical and ecological processes and is easily implemented across a wide range of computing platforms (multiprocessor systems, UNIX workstations, Linux or DOS).

Delft3D

Another well known modelling system to investigate hydrodynamics, sediment transport, morphology and water quality is Delft3D, developed by Delft Hydraulics (<http://www.deltares.nl/>). It is decomposed in several modules, not only hydrodynamics but also directed to higher levels of marine ecosystems:

- Delft3D-FLOW: is the heart of Delft3D and it simulates the multidimensional (2D or 3D) hydrodynamic processes – calculates non-steady flow and transport phenomena resulting from tidal and meteorological forcing on a curvilinear, boundary fitted grid, including the effect of density differences due to a non-uniform temperature and salinity distribution. The application is used to predict tidal, and wind induced flows in shallow areas, coastal areas, estuaries, rivers and lakes. In three dimensional simulations, the vertical grid is defined following the sigma coordinate approach. It has the possibility to decompose the domain in sub domains and take advantage of parallel processing, increasing flexibility, accuracy and efficiency. The Delft3d-Parflow

is an operational version of Delft3D-Flow that enables the coupling of models with different dimensions (e.g. 2D with 3D models).

- Delft3D-WAQ: is a general water quality module used to describe a wide range of water quality processes. It includes a library with the most usual processes considered in water quality problems like circulation and phytoplankton production. This module integrates the complexity of dynamic variability linked to physical processes with the processes that govern biogeochemistry.
- Delft3D-SED: is the module responsible for the transport of cohesive and non-cohesive sediments to study sediment/erosion patterns. Generally used to calculate the short-term transport of sediment and sand when the effect of bottom topography changes on flow conditions can be neglected.
- Delft3D-PART: this module is applicable in all geographic regions where the FLOW method is applied, and simulates the transport processes and simple chemical reactions following particles in three spatial dimensions and over time – it is a Lagrangian simulation, opposed to a Eulerian approach that simulates concentrations in fixed sites.
- Delft3D-ECO: module that simulates the biochemical and biological processes related with algal growth and nutrient dynamics relevant to predict eutrophication phenomena. It uses more detailed processes than the WAQ module and it is essential when the effects of management measures must be predicted.
- Delft3D-WAVE: this module is used to simulate the propagation and transformation of random, short-crested, wind generated waves in coastal waters which extend to estuaries, tidal inlets, barrier islands with tidal flats and channels. Design studies for offshore and harbour installations, and coastal development and management are the obvious users of this module.
- Delft3D-QUICKIN: module that generates and manipulate model bathymetries from the depth samples (raw acquired data) to the grid resolution of the model.
- Delft3D-RGFGRID: this module generates orthogonal, curvilinear grids of variable size for the FLOW module allowing an high resolution grid size in the area of interest and a lower resolution in the model boundaries.

POM

The Princeton Ocean Model (POM) is a general circulation model, written in FORTRAN, where the equations of motion, mass and heat conservation are discretized on a staggered Arakawa-

C (Arakawa and Lamb, 1977) finite difference grid and employing the sigma-coordinate (terrain-following) system for the discretisation of the vertical dimension (Blumberg and Mellor, 1987). It is a free surface ocean model with embedded turbulence and wave sub-models, with wet-dry capability and able to simulate a wide-range of problems, like circulation and mixing processes in rivers, estuaries, shelf and slope, lakes, semi-enclosed seas and open and global ocean. The model also calculates temperature, salinity, 3D velocity fields and the turbulent kinetic energy (Chapelle et al., 2005b).

Input and output data are described in text files, in the form of matrices, facilitating the visualization of the results in any application with graphical capabilities, like MatLab®.

The Princeton Ocean Model was developed at the Princeton University, New Jersey, U.S., and is freely available at <http://www.aos.princeton.edu/WWWPUBLIC/htdocs.pom/index.html>, the POM website.

MARS3D

The MARS3D model is a 3D hydrodynamic model (Lazure and Jegou, 1998; Lazure and Dumas, 2008) developed at IFREMER (French Research Institute for Exploitation of the Sea). MARS3D, that stands for 3D hydrodynamic Model for Applications at Regional Scale, is subjected to Boussinesq and hydrostatic approximation, forced by river run-off, wind speed and direction, and uses a spatial discretization with a staggered Arakawa-C grid (Arakawa and Lamb, 1977) and vertical sigma coordinates. The model runs on Linux/Unix platforms and has been used mostly for studying the Atlantic seaboard and the Mediterranean sea. The visualization of the model outputs can be done with a GIS platform or with VisuMars, a post-processor graphical application developed also by IFREMER.

MOHID

MOHID (<http://www.mohid.com/>) is a three-dimensional water modelling system developed by a Portuguese team – Marine and Environmental Technology Research Centre (MARETEC) at Instituto Superior Técnico, Technical University of Lisbon. The use of object oriented programming allows the adoption of an integrated modelling philosophy, considering physical and biogeochemical processes, at different scales (nested models) and systems (watersheds and estuaries) (Neves, 1985; Miranda et al., 2000).

MOHID is divided in three tools – MOHID Water, MOHID Land and MOHID Soil – to study the water cycle in different variants. A common framework is shared, facilitating the integration of

the mentioned tools. The usage of different time steps for different modules is one of the main advantages to harness the available computational power. The different tools can run in different processors, allowing parallel processing and communicating by MPI messages (Message Passing Interface), improving system performance.

MOHID Water is a three-dimensional numerical program to simulate surface water bodies (oceans, estuaries, reservoirs). The module covers hydrodynamics (circulation patterns and velocity fields that transports the water properties), waves (wave diffraction and refraction and its influence on sediment transport), sediment transport (vertical and horizontal movements and bottom physical processes), water quality and ecology (primary and secondary production and nutrients' cycle), plume dispersion and turbulence.

EcoWin2000

"EcoWin2000 is an ecological model for aquatic systems, developed using an object-oriented approach. It resolves hydrodynamics, biogeochemistry and can incorporate population dynamics for target species" (from EcoWin200 site: <http://www.ecowin2000.com>). This modelling tool was developed by the Portuguese team Geochemical and Ecological Modelling (<http://www.ecowin.org>) based at the Faculty of Sciences and Technology, New University of Lisbon, dedicated to the aquatic ecosystem research, and integrated in the Portuguese Institute of Marine Research – Centre for Ecological Modelling (IMAR-CEM). Initially developed as EcoWin (Ferreira, 1995) the application evolved and EcoWin2000 (E2K) provides a platform for integration with other models maintaining functionality of its own. It can be used for short-term and for long-term simulations. Rather than integrating sub-models, the various components act as self-contained objects and can be added/removed from the simulations.

The strength of this application resides in the biogeochemical model and the object-oriented approach that enables groups of objects to share state variables and, when desired, define new state variables or change the interactions among state variables without affecting the code already developed.

The hydrodynamic simulation is centred in the transport object with its advection-diffusion equations - salinity, water flow and velocities are some variables calculated by this object - but EcoWin2000 is tailored to import these values from other models specialized in hydrodynamics, like Delft3D.

2.4.3 Biogeochemical Modelling

The availability of equations based on physical laws of classical mechanics, known from the 18th century, is one of the main differences between hydrodynamic and biogeochemistry models. The biogeochemical phenomena are generally described by equations that are not universally accepted by the research community and are, in most cases, empirical or semi-empirical.

There are scarce intercomparison analyses between biogeochemical models, mainly due to the lack of universally accepted equations for the description of biogeochemical phenomena. One of the first studies comparing biogeochemical models was reported in Skogen and Moll (2000) concerning the primary production of the North Sea using two ecological models. Both models gave similar results regarding annual mean primary production, its variability and the influence of river inputs. The terms “biogeochemical modelling” and “ecological modelling” are used most of the times as synonyms, and refer to mathematical modelling of ecosystems including biogeochemistry, with emphasis on nutrient cycles, physiologic processes (such as respiration and feeding), population level processes (such as reproduction and mortality), and community level processes (such as predation and competition) (Chapelle et al., 2005b). Biogeochemical models consist of biotic compartments (different biological species or functional groups) and abiotic compartments (dissolved substances and suspended matter), represented by state variables and described by differential or difference equations governing the transfer of materials between them. The subjectivity of the modellers, regarding the importance of different processes and variables, guides the choice of the variables and the model internals (Jørgensen and Bendoricchio, 2001).

Ecological models group biological functional entities and make several simplifying assumptions about real systems. The generality of the authors – e.g. Jørgensen and Bendoricchio (2001), Gertsev and Gertseva (2004) – divide ecological models in stationary or time-dependent models. According to the same authors, the models can be classified as “models with lumped parameters”, when homogeneity is assumed along spatial coordinates (zero-dimensional models, 0D), or “models with distributed parameters”, when heterogeneity along spatial coordinates is considered (one- (1D), two- (2D) or three-dimensional (3D) models). Also, models can be considered continuous or discrete, according to the way time is represented, deterministic or stochastic, according to the type of mathematical relationships used, analytic or numeric, according to the way model equations are solved (cf. 2.2).

When spatial heterogeneity is considered, ecosystems may be divided in boxes of any size and shape (box-models) or are embedded in the same type of grids used in hydrodynamic models (grid-models). Exchange processes of pelagic variables between different boxes and between the ecosystem and its boundaries may be parameterized on the basis of steady-state balances of conservative state variables (Baretta and Ruardij, 1988) or from simulations with hydrodynamic models (Bacher, 1989; Raillard and Ménesguen, 1994; Ferreira et al., 1997).

There are no specific universal equations that can be used to determine how material is transferred between compartments of an ecosystem model. A general equation of population growth, which accommodates most of the limiting processes in a closed system, has been formalised by Wiegert (1979):

$$\frac{dx_j}{dt} = \sum_{i=1}^m (e_{ij}\tau_j p_{ij} f_{ij} X_j) - (\mu_j + \phi_j + \rho_j) X_j - \sum_{k=1}^m (\tau_k p_{jk} f_{jk} X_k) \quad (3)$$

The first sum represents the assimilated ingestion or uptake by species j from all other modelled species or abiotic sources. The middle term represents the losses due to physiological causes and natural death. The last sum represents the predation on species j by other species. The coefficients are defined as follows:

- e_{ij} - is the assimilation efficiency of species j using resource i ;
- τ_j - is the maximum specific ingestion/uptake rate of species j ;
- p_{ij} - is the preference of species j for resource i ;
- f_{ij} - is the limitation of ingestion/uptake of resource i by species j ;
- μ_j - is the specific loss rate due to natural mortality;
- ϕ_j - is the specific loss rate due to excretion;
- ρ_j - is the specific loss rate due to respiration.

These coefficients may depend on a variety of physiologic and behavioural interactions making them non-linear functions of the species or abiotic sources. The equations are not as well-defined as the physical equations of motion, because they are not based on known quantitative laws, as those available in physics. It is common to simplify most of these coefficients to either constant values, functions of time or space, or functions of the physical forcing (Taylor, 1993).

The type, quality and quantity of data available for a particular ecosystem, as well as some of its structural and functional characteristics, limit the number of processes that are represented in models, as well as the degree of detail used. In shallow-water ecosystems the relative importance of benthic processes tends to be larger than in deep-water ones. This may be one

of the reasons why there are more studies on benthic ecology in shallow-water ecosystems, and benthic species or species groups are generally included in models, with particular relevance to macroalgae, macrophytes and bivalve species. Since larger species are generally studied with more detail – those with more commercial importance – equations governing their growth include a lot of physiologic detail and, most of the times, there are a lot of equations and parameters for each species whereas, with other organisms, such as phytoplankton and zooplankton, several species are lumped into one variable (Chapelle et al., 2005b).

In the European project DITTY (EC, 2003) different biogeochemical models were compared. The examples focused (Ria Formosa, Portugal; Mar Menor, Spain; Etang de Thau, France; Sacca di Goro, Italy; Gulf of Gera, Greece) showed the diverse approaches employed, with some common features emerging among different models, regarding several biological variables and processes. Chapelle et al. (2005b) compared available models according to their state variables (water column and sediments), processes, time scale, forcing functions, coupling procedures, methodology of calibration/validation, sensitivity analysis, software, hardware and limitations.

Table 2-1 presents some examples of ecological models applied to aquatic ecosystems, ranging from 0D to 3D, with the dates of publication and the software or the programming language used in implementation.

Table 2-1: Examples of aquatic ecosystem models published between 1988 and 2004 (Pereira et al., 2006)

| Authors | Model dimensions | Model application |
|--------------------------------|------------------------|--|
| Baretta and Ruardij (1988) | 1D | Ems Estuary (Netherlands) ecosystem simulation (FORTRAN) |
| Fasham et al. (1990) | 0D | North Atlantic mixed layer ecosystem model (FORTRAN) |
| Bax and Eliassen (1990) | 0D | Modelling multispecies analysis in Balsfjord (Northern Norway) with SKEBUB model |
| Taylor (1993) | 0D | North Atlantic mixed layer ecosystem model |
| Ferreira (1995) | 1D | EcoWin applied to Tagus estuary (Portugal) and Carlingford Lough (Ireland) |
| Lanhart et al. (1995) | 3D | ERSEM II – Ecosystem model of the North Sea (FORTRAN) |
| van der Tol and Sholten (1997) | 2D horizontal | Ecosystem model of the Oosterschelde estuary (SMOES) (SENECA software) |
| Fulton et al. (2004) | 0D and 2D applications | Generic bay ecosystem models |

Several examples of ecological models, from very simple to very complex, can be found in the literature, namely in Bacher et al. (1998), Hawkins et al. (2002) and Duarte et al. (2003). Particularly complex is the last example: one ecological model developed for Sungo Bay, located in Shandong province, North-East of People's Republic of China.

2.5 Summary

In this chapter the fundamental concepts of simulation and modelling were presented, with particular emphasis on themes related to ecological modelling and, more specifically, the simulation of coastal ecosystems. Examples of hydrodynamic and biogeochemical models were highlighted, focusing on the description of the main physical, chemical and biological processes.

Also a description of programming languages used in ecological models was made, comparing the evolution of modelling techniques with the evolution of the programming languages paradigms. The object-oriented programming proposal is gaining some advantage in recent years because of its similarity with the modelling and behaviour of the living beings.

Many of the described tools are complementary and can be sequenced, adapted and integrated to simulate the different parts of an ecosystem, generating valuable results for the understanding of each system as a whole.

3 Concepts and State-of-the-Art: Artificial Intelligence and Optimization

“The ability to learn is one of the most fundamental attributes of intelligent behaviour”

(Michalski et al., 1984).

3.1 Introduction

Models are being used also as helpers for management purposes. The introduction of ecological modelling as a management tool began around 1970 (Jørgensen and Bendoricchio, 2001). The idea behind this concept was to identify the relationships between environmental science, ecology, economics, ecological modelling and environmental management and technology. At those days, only a small number of people, mostly researchers and ecologists, were concerned about what today is called sustainable development.

More recently, researchers in biological sciences began to use individual based models to distinguish the individual behaviour of biological species – the individual is the central object, and models explore the mechanisms through which population and ecosystem ecology arises from how individuals interact with each other and their environment. It was the birth of the individual-based models – IBM (Huston et al., 1988).

The establishment of some modern artificial intelligence techniques and applications, namely the emergence of the concept of artificial agents (Wooldridge, 1999), multi-agent systems (Huhns and Stephens, 1999), and distributed artificial intelligence (Ferber, 1999; Weiss, 1999)

and machine learning techniques (Michalski et al., 1984), allow the researchers to build individuals with more sophistication, varying their behavioural rules, defining different internal models for the external world, and retaining memory of past events influencing their decisions. The computational advances made possible the emergence of a new approach for simulation known by many names – the agent-based modelling (ABM), agent-based simulation (ABS) or agent-based modelling and simulation (ABMS) (Macal and North, 2006).

The integration of agents in the simulation system introduces a new level of complexity, because the agents have self interests and try to guide the simulations in order to maximize their own objectives. The existence of opposite interests can conduct to antagonistic results after several simulations' runs. One way to solve these conflicts is the addition of a decision support module to the system, to weight the several interests in game and help the managers to take decisions about the ecosystem.

Following sections present concepts and state-of-the-art on Artificial Intelligence topic, Machine Learning and optimization techniques, as well as decision support and management issues related with ecological models.

3.2 Artificial Intelligence and “Fifth Generation” Models

Models of ecosystems attempt to capture their intrinsic characteristics. However, ecosystems differ from most other systems by being extremely adaptive, having both the ability of self-organization and a large number of feedback mechanisms. How can the researcher build models that are able to reflect these characteristics? Reliable prognoses can only be done by models with a correct description of ecosystem properties, including a proper description of adaptation capabilities, including shifts in species composition. Models including a dynamic description of model parameters to mimic self-organization are sometimes called “fifth generation” models (Jørgensen and Bendoricchio, 2001).

Ecosystems are irreducible systems (Wolfram, 1984; Jørgensen, 1992a, 1992b): it is almost impossible to design simple experiments to reveal a biological relationship in all its details to transfer from one ecological situation and one ecosystem to a different situation in another ecosystem. For instance, the growth of living organisms is dependent on many interacting factors, which are functions of time and of feedback mechanisms that regulate all the factors and rates. This complexity prohibits the reduction to simple relationships that can be used repeatedly. It is the extremely high number of feedbacks and regulations that makes possible

for living organisms and populations to survive and reproduce in spite of changes in external conditions. Furthermore, the feedbacks are constantly changing and the adaptation itself is adaptable – if a regulation is not sufficient, another regulation process higher in the hierarchy of feedbacks will take over (Jørgensen and Bendoricchio, 2001).

Ecosystems show a high degree of heterogeneity in space and time, and all their components are steadily moving and their properties steadily modifying – one ecosystem never returns to the same situation again. Every point is different from any other point and offers different conditions for the various life forms. This enormous heterogeneity explains the high number of different species on earth. Because ecosystems are not homogeneous in relation to properties concerning matter, energy and information, they may be considered anisotropic - they exhibit properties with different values, when measured along axes in different directions. These variations in time and space make it particularly difficult to model ecosystems and to capture their essential features (Jørgensen and Bendoricchio, 2001).

The application of new tools and methods for model building became necessary as a result of limitations of available models. Object-oriented models boost the introduction of individual-based models and helped artificial intelligence to enter in the world of modelling giving way to agent-based models, fuzzy models and knowledge-based models.

The world is amazingly complex. Systems that need to be analyzed are becoming more and more complex: decision-making needs to be decentralized (e.g. deregulated electric power industry), the systems approach design limits (e.g. transportation networks) and there are an increasing number of interdependencies between physical and economic infrastructures (electricity, natural gas, telecommunications). New tools, toolkits and modelling approaches had to be developed to analyze the intrinsic complex systems that always existed but had never been analyzed due to the lack of tools - social systems and networks, economic markets and the diversity among economic agents. The organization of data into databases at fine level of granularity (micro-data) and the advancing computational power helped supporting micro-simulations that would not have been plausible just a couple of years ago.

In ecology, the individuality of the individuals and the locality of their interactions, within a population, are two basic principles that must be present in all ecological models (Jørgensen and Bendoricchio, 2001). Agent-based modelling (ABM) is a computer based methodology in which one set of independent agents interacts with the environment and with each other (representing the dynamics of complex systems with many interacting parts).

The integration of agents software to computer simulation gave rise to the so-called agent-based simulation systems, allowing the introduction of mechanisms more similar to the behaviour and development of living beings, including their adaptability to new environments and natural conditions. This adaptability to new environments is integrated into the agent's software through the use of optimization and machine learning techniques. This new field is grounded mainly in the biological and social sciences.

Agent-based modelling and simulation (ABMS) appears as a new paradigm because it claims to separate the rules of the researchers and the business decision makers. The assumption to this approach is that the businesses use computers to support decision-making and researchers use electronic laboratories to support their research (Macal and North, 2006).

3.2.1 Agent Definition

Although there is no universal agreement on the precise definition of the term “agent”, available definitions tend to agree in more points than they disagree (Macal and North, 2006). Some relevant definitions, from the literature, follow:

“A person or business authorized to act on another's behalf.” [dictionary.com]

“One empowered to act for or represent another.” [www.thefreedictionary.com]

“An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.” (Franklin and Graesser, 1997)

“An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.” (Jennings, 2000)

“An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objective.” (Wooldridge, 2002: 15)

“An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.” (Russel and Norvig, 2002: 32)

“An agent can be a physical or virtual entity that can act, perceive its environment (in a partial way) and communicate with others, is autonomous and has skills to achieve its goals and tendencies.” (Ferber, 1999)

“An agent is just something that acts (...). But computer agents are expected to have other attributes that distinguish them from mere ‘programs’, such as operating under autonomous control, perceiving their environment, persisting over a prolonged time period, adapting to change, and being capable of taking on another’s goals. A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.” (Russel and Norvig, 2002: 4)

“Agent programs differ from regular software mainly by what can best be described as a sense of themselves as independent entities. An ideal agent knows what its goal is and will strive to achieve it. An agent should also be robust and adaptive, capable of learning from experience and responding to unforeseen situations with a repertoire of different methods. Finally, it should be autonomous so that it can sense the current state of its environment and act independently to make progress toward its goal.” (Maes, 1996)

“An autonomous, reactive, pro-active computer system, typically with a central locus of control, that is at least able to communicate with other agents via some kind of communication language. Another common view of an agent is that of an active object or a bounded process with the ability to perceive, reason, and act.” (Weiss, 1999: 583)

“An agent is a computational system, situated in a given environment, which has the perception of that environment through sensors, has capacity of decision, acts autonomously in that environment and has capabilities for high-level communication with other agents and/or humans, in order to achieve a goal or to perform a function for which it was designed.” (Reis, 2003)

“(…) intelligent agents will be a key technology as computing systems become ever more distributed, interconnected, and open. In such environments, the ability of agents to autonomously plan and pursue their actions and goals, to cooperate, coordinate, and negotiate with others, and to respond flexibly and intelligently to dynamic and unpredictable situations will lead to improvements in the quality and sophistication of software systems (...)” (Wooldridge and Jennings, 1995)

One different, and curious, definition of agent was done by Ted Selker, from IBM, during the 90's:

"An agent is a software thing that knows how to do things that you could probably do yourself if you had the time."

Some scientists consider that a formal definition is not easy to establish, and prefer to distinguish two general notions of the term "agent": the weak and the strong concepts; the former one relatively uncontroversial and the latter more controversial (Wooldridge and Jennings, 1995; Hermans, 1996). The following properties are the basic and uncontroversial ones an agent must reveal (weak notion):

- **Autonomy:** agent operates without the direct intervention of humans or others, and have some kind of control over its actions and internal state;
- **Social ability:** agent interacts with other agents, and possibly humans, via some kind of agent communication language;
- **Reactivity:** agent perceives its environment and responds in a timely fashion to changes that occur in it;
- **Pro-activeness:** agent is able to exhibit behaviour by taking the initiative;
- **Temporal continuity:** agent is continuously running processes, exchanging between active/passive processing.

The strong notion requires that an agent must have properties that are more usually applied to humans:

- **Rationality:** an agent will act in order to achieve its own goals;
- **Adaptivity:** an agent must be able to adjust itself to the habits, working methods and preferences of its user;
- **Benevolence:** is the assumption that an agent does not have conflicting objectives and will always try to do what it is asked for;
- **Collaboration:** an agent should not unthinkingly accept (and execute) certain orders that could put in danger the environment or damage other agents;
- **Mobility:** an agent can have the ability to move around the environment.

From a practical point of view, and from the definitions above, Macal and North (2006) consider that agents must reveal the following characteristics:

- An agent must be **identifiable and self-contained**. It is discrete, with a set of characteristics and rules governing its behaviour and decision-making capability, has a boundary and one can easily determine whether something is part, or not, of an agent.
- An agent should be **goal-oriented**, having goals to achieve with respect to its behaviour. An agent can compare the outcome of its behaviour relative to its goals (not necessarily objectives to maximize).
- An agent is **situated**, living in an environment in which it can interact with other agents. Agents have communication skills and capabilities to respond to the environment, recognize and distinguish cooperative and antagonist agents.
- An agent is **autonomous and self-directed**. It must function independently in its environment and in the interaction with other agents.
- An agent must be **flexible, with ability to learn** and adapt its behaviours based on its experience. This requires some kind of memory and the existence of processes that modify its rules of behaviour.

3.2.2 Multi-Agent Systems Definition

After the agent's definition it is important to define what is understood by multi-agent systems. Once again, there are several definitions in the literature:

"A multi-agent system is a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity." (Durfee et al., 1989)

The definition given by Jacques Ferber (1999) seems to be the more meaningful for the research in ecology and environmental sciences:

"A multi-agent system (MAS) contains an environment, objects and agents (the agents being the only ones to act), relations between all the entities, a set of operations that can be performed by the entities, and the changes of the universe in time and due to these actions." (Ferber, 1999)

Sometimes one image is better than one thousand words and Figure 3-1, from Jennings (2000), illustrates the typical structure of a multi-agent system.

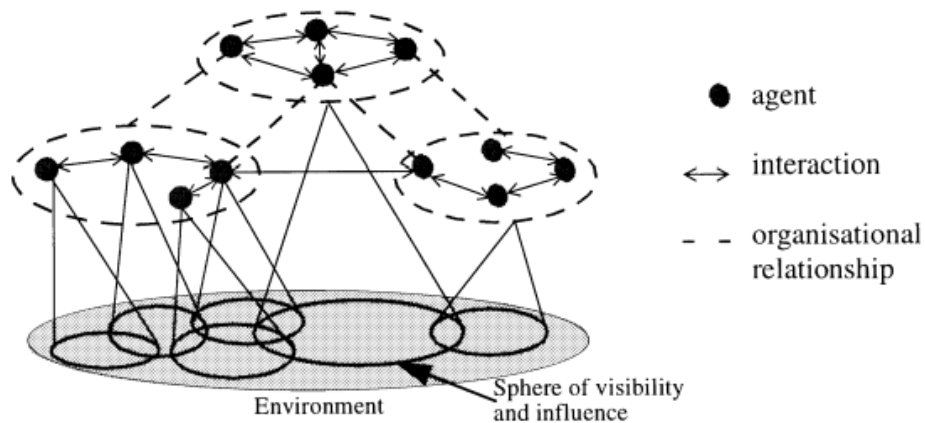


Figure 3-1: Typical structure of a multi-agent system (Jennings, 2000)

“The system contains a number of agents, which interact with one another through communication. The agents are able to act in an environment; different agents have different ‘spheres of influence’, in the sense that they will have control over – at least be able to influence – different parts of the environment. These spheres of influence may coincide in some cases, which may give rise to dependency relationships between the agents.” (Wooldridge, 2002: 105)

From the above definitions, six components can be identified in one MAS (Bousquet and Le Page, 2004):

- An environment, usually a space;
- A set of situated objects – at a given moment it is possible to associate any object with a position in the environment;
- A set of agents (maybe a subset of objects), representing the active entities in the system;
- A set of relations, that link objects (and agents) to one another;
- A set of operations, making it possible for the agents to perceive, produce, transform and manipulate objects;
- Operators with the task of representing the application of the agents’ actions and the reaction of the world to this attempt of modification (laws of the universe).

Michael Wooldridge points the autonomy and the agent’s view of the world (environment) as the more relevant characteristics of an agent to be part of a MAS (Wooldridge, 2002). Even if the communication capacity of an agent is reduced, it can make part of one multi-agent system because the result of its actions will be reflected in environmental changes and will be perceived by the other components of the MAS, as well as the agent perceives changes in the

environment as a result of actions from other agents. Also important is the understanding of the type of interactions that take place between the agents.

Typically, multi-agent systems research refers to software agents. However, the agents in a multi-agent system could be robots, humans or human teams. A multi-agent system may contain combined human-agent teams. Multi-agent systems can manifest self-organization and complex behaviours even when the individual strategies of all their agents are simple.

Usually, MAS rely on a bottom-up approach, and one issue in MAS is formalizing the necessary coordination among agents answering questions in key areas like (Bousquet and Le Page, 2004):

- Decision-making - What mechanisms of decision-making are available to the agents? How does agent relate perceptions, representations, and actions?
- Control - Is there any hierarchical relationship among agents? Are they synchronized?
- Communication - What kinds of messages do agents exchange? What is the messages' syntax?

As seen in previous section (cf. 3.2.1) agents could be classified as showing weak or strong notions of agency, according to their attributes (Wooldridge and Jennings, 1995). The weak notion of agency, which comes from Distributed Computing and Distributed Artificial Intelligence, sees agents as a paradigm of network based cooperative automation. MAS can simplify problem-solving by dividing the necessary knowledge into subunits, by associating an intelligent independent agent to each subunit, and by coordinating the agents' activity. The strong notion of agency, from Artificial Intelligence (AI), leads toward an anthropomorphic view where agents are seen as conscious, cognitive entities that have feelings, perceptions and emotions just like humans.

Since the beginning, fundamental research is being conducted on the problems associated with the architecture of the agents, the representation of agents' decisions and protocols for communication.

Agents Architectures

The internal structure of an agent is dependent on the objectives that the agent must comply. Reactive agents represent, perhaps, the most widely implemented architecture, but when human actions are considered in the ecosystem, architectures place more emphasis on deliberation (Bousquet and Le Page, 2004). Production rules are very often used to simulate

the deduction process of an agent facing environmental stimuli, and are usually mixed with parameterized functions or other kinds of formalizations, generating more structured architectures capable to process competitive tasks or belief-desire-intention architectures.

Architectures based on the evolutionary metaphor use algorithms mimed from the theory of evolution and designed to understand adaptation, like genetic algorithms (Holland, 1975).

Architectures for competitive tasks have many links with robotics and animats - the idea is to consider that different input stimuli activate different tasks. Tasks could be activated in parallel creating a behaviour system that can be complex. Each task typically is weighted and has a threshold value for its activation; the activation level is calculated taking into account the intensity of the stimuli, for example (extracted from Bousquet and Le Page (2004)):

$$a_i(t) = \frac{w_i(t)}{\sum_{j=1}^n w_j(t)} x_i(t) \quad (4)$$

where,

$a_i(t)$ is the activation level of task i in instant t ;

$w_i(t)$ is the weight of task i in instant t ;

$x_i(t)$ is the stimulus intensity i in instant t ;

n is the number of tasks.

If the activation level, calculated for the task, is greater than its threshold level, the task is activated. The *subsumption architecture*, proposed by Brooks (1991), is a possible architecture designed to select tasks. It embodies the fundamental ideas of decomposition into layers of task achieving behaviours, and different activities are represented by different levels. In this architecture, each layer is composed by a simple finite state machine, with a handful of states, internal variables and timers, which run asynchronously sending and receiving messages. There is no central locus of control and the finite state machines are data-driven by the received messages. The arrival of messages or the expiration of a designated time period causes the finite state machines to change state. Layers are controlled by *suppression* and *inhibition* mechanisms. The upper levels are capable of suppressing the inputs of the lower levels and inhibiting their outputs (Brooks, 1991). It is important to notice that this architecture has few applications in ecology, due to its simplicity.

Architectures based on neural networks emphasises the learning capacity of the agents. The perception-action relation is modelled by a network whose connections evolve. In an early work published in 1992, Collins and Jefferson endow ants with neural network so that they are

capable of learning (Collins and Jefferson, 1992). Although each individual ant performs typically only 20 to 42 distinct behaviours, the objective of the study focus on two central tasks: the search for food and its recovery to a central location. Several neural networks are embedded in the ant to enable the execution of different tasks, namely one to learn to explore, and one to learn to transport. Another work with application of these methods is found in Dagorn et al. (2000), who seeks to understand the movements of predatory fishes within their environment. In this example, artificial individuals were compared with real fishes (three species of tuna, three species of billfishes and one species of shark) in pelagic regions of the tropical oceans. When evolution is the focus of the research, coupling neural networks with genetic algorithms is becoming increasingly common (Chen and Hare, 2006).

In his work on bird flight, Reynolds makes the agent's decision dependent from the additions of physical forces. It applies vector calculation in force fields resulting from the attraction or repulsion of other agents (Reynolds, 1987). In his approach, agents represent elementary particles. This type of modelling is easily found in fields related to fluid dynamics – water flow, crowd dynamics, urban traffic flow, or mass animal movements. In some resource management applications the decision-making processes of economically rational agents is simulated. These agents can use models based on operational research (gradient calculation) or microeconomics (utility maximization) to obtain an optimal solution in the presence of constraints. Also, agents can use decision-making methods based on multi-criteria analysis. In any of these examples the architecture is based on parameterized functions, which makes difficult the characterization of an individual rationality of the agent (Bousquet and Le Page, 2004).

The belief-desire-intention (BDI) architecture, which comprises objectives, representation, and involvement in individual or collective actions, is another base for the cognitive agents. However, most applications for ecological problems pay attention mostly on understanding the coordination or the relations of the agents with the environment. Very simple BDI agents are used in ecological systems and the spatial representation is, beyond question, the primordial cognitive dimension that must be present in all ecological systems' applications. The most widely used method for the spatial representation is the memorization of space and resources building a mental map of the environment. In social sciences the notion of reputation is another cognitive dimension used to model agent's interactions. Agents may have beliefs about other agents, depending on their individual experience or as a result of their social reputation. These beliefs may be used to guide their level of commitment to collective

resource management (Rouchier et al., 2001; Mathevet et al., 2003; Bousquet and Le Page, 2004).

Agents Interactions

The interactions between agents are an essential part of each multi-agent system. Applied to ecology, three major types of interactions may occur: communication among agents, physical interactions (growing, eating, putting), and interactions mediated by the environment.

Unlike in other areas of research involving agents, in ecological applications it is not very common the use of direct interaction through the exchange of messages between agents. One of the few studies documented is from Baray (1998) where predators communicate to surround a prey. Another interesting work involves agents representing humans as part of the ecosystem and simulates negotiations among Bolivian farmers and exchanges of contracts, goods, and services (Paz Betancourt et al., 1996).

Physical interactions, like predating and eating, are often used in ecological models. The most used kind of interactions are those mediated by the environment, namely, the relation between organisms and their environment - the physical space and the resources. Each agent's action transforms the common environment with a retroactive effect on other agents – the dynamics of the environment can be used as a medium for collective adaptation (Bousquet and Le Page, 2004). Many applications of this approach (swarm intelligence, co-evolution, world model) are published and, basically, use the environment as a set of signals for movement, reproduction and nourishment.

Development Tools

MAS applications are generally developed with an object-oriented language and some of them use platforms, that can be dedicated or generic.

Generic frameworks like the well known Swarm (Minar et al., 1996; Bonabeau et al., 1999), StarLogo¹ (Resnick, 1996) and NetLogo² (Gilbert and Troitzsch, 2005; Vidal, 2007) are robust and user-friendly, and are widely documented. To describe the cognitive aspects of the agents

¹ <http://education.mit.edu/starlogo/>

² <http://ccl.northwestern.edu/netlogo/>

some modelling languages also exist, like SDML³ (Strictly Declarative Modelling Language), a modelling language where all knowledge is declarative, and represented on rule bases and databases.

Examples of dedicated platforms are much more numerous: they are structurally-fixed models, equipped with one user interface through which the user sets the parameter values and analyses the results of the simulations. For example, MANTA (Drogoul and Ferber, 1992) is a dated system focused on problems of foraging or task specialisation in a society of insects; BacSim (Kreft et al., 1998) models microbiological bacterial colony growth and is available online⁴ for demonstration; Formosaic (Liu and Ashton, 1998) deals with fragmented forestry dynamic, simulating forest dynamics in landscape mosaics; OSMOSE (Shin and Cury, 2001) generates virtual fish species interacting via predation. All dedicated platforms try to explain a concrete problem but most of them maintain a distance between the modeller and the end-user, with a heavy negative consequence: great part of many excellent models have been quickly forgotten due to the lack of evolution after being supplied to the end-users (Lorek and Sonnenschein, 1998).

Platforms for social and ecological simulations provide utility programs to simulate ecosystems and resource management problems. Frameworks like Ecosim (Lorek and Sonnenschein, 1998), use C++ objects and a simulation engine based on discrete events, or Cormas (Bousquet et al., 1998), written in Smalltalk, are highly generic and can be directly linked with geographical information systems (GIS) to integrate the evolved spatial support.

3.2.3 Individual-Based Modelling

At the end of the 1980s a new approach was developed: the individual-based modelling (IBM). Two ideas were behind the development of this approach that intends to simulate the global consequences of local interactions of a population (Huston et al., 1988): first, the need to take into account the individual because of its genetic uniqueness and, second, the fact that each individual is situated and its interactions are local. Each individual is modelled as an object, and different objects represent different individuals, with unique genetic characteristics (Kreft et al., 1998; Jørgensen and Bendoricchio, 2001).

³ <http://cfpm.org/sdml/>

⁴ http://www.theobio.uni-bonn.de/people/jan_kreft/bacsim.html

When IBM was introduced it was well received and now a very large number of publications and experiments refer to it. The proceedings of the conference “Individual-based models and approaches in ecology” (DeAngelis and Gross, 1992), held in 1990, constitute the first fundamental literature on this issue.

Individuals might represent plants and animals in ecosystems, vehicles in traffic, or autonomous characters in animation and games. Traditional modelling techniques averaged the characteristics of the population, and models attempted to simulate changes in these characteristics. IBM contravenes this approach paying attention to each individual, modelled as an entity that interacts with the environment and the other individuals. The behaviour, characteristic parameters and possible interactions are defined individually, supplying means to track individual progress throughout the time.

If the individuality of the individuals is placed in one agent, a sub-type of this approach is referred as Agent-oriented Individual Based Modelling (AIBM).

The web site maintained by Craig Reynolds (<http://www.red3d.com/cwr/ibm.html>) is probably the most frequently referenced regarding IBM related research. Even though the website has not been, apparently, updated recently, it remains one of the most referenced by modellers using IBM. Online resources include software toolkits for implementing individual-based models, links to online IBM applications, and links to offline resources, journals, conferences and to academic and commercial laboratories and groups. Interesting are the wide areas where this technique is/was tested, from ecology and biology, human and social sciences, economics, traffic and vehicle simulations, animation and interactive multimedia.

IBM can be spatially explicit meaning that the individuals are associated with a location and can move around their environment – natural model for an animal in an ecological simulation. If the subject of the simulation is a plant, the individual doesn’t exhibit mobility.

3.2.4 Agent-Based Modelling and Simulation

When IBM is transformed in AIBM, this kind of simulation falls in the scope of multi-agent systems (MAS) simulations. However there are some differences between both. IBM was developed by ecologists to understand the role of diversity, or heterogeneity in ecosystems, introducing the notion of the individual – each "agent" corresponds to an autonomous individual in the simulated domain. MAS are highly influenced by the computer and social sciences, and give more emphasis on the decision-making process of the agents and to the social organization in which these individuals are embedded. Furthermore, as Bousquet and Le

Page note, “an agent is not necessarily an individual”, it can be the representation of any level of organization (Bousquet and Le Page, 2004).

Agent-based modelling and simulation (ABMS) is the paradigm that covers the more recent studies and advances in the field of modelling and simulation comprised with interacting autonomous agents. ABMS has strong roots in the fields of multi-agent systems, and robotics, from the field of Artificial Intelligence and Distributed Artificial Intelligence (DAI). Its main objectives are in modelling human social and organizational behaviour, and individual decision-making (Bonabeau, 2002). This forces the need to represent social interaction, collaboration, group behaviour and the emergence of higher order social structure. The objective became to reproduce the knowledge and reasoning of several heterogeneous agents that need to coordinate and jointly solve planning problems. Instead of concentrating only in the agent and its autonomy, the researchers focused on the organization of multiple agent interactions (Huhns and Stephens, 1999).

The theoretical foundations of ABMS derive from many fields of science (computer, complexity, systems dynamics, management, and social sciences) and from traditional modelling and simulation. Moreover the construction of ABMS systems is supported by the notion that systems are built “bottom-up”, contrasting to the “top-down” systems view taken by the Systems Dynamics (Macal and North, 2006). ABMS tends to be descriptive, modelling the individuals behaviour, instead of identify optimal behaviours, the traditional operations research approach.

ABMS has its historical roots in complex adaptive systems (CAS), originally motivated by research in adaptation and emergence of biological systems, which have the ability to self-organize and reorganize dynamically their components, in order to survive and excel in their environment. Properties like aggregation (enable groups’ formation), nonlinearity (inexistence of simple extrapolation), flows (allowing the transfer of resources and information between agents), and diversity (agents behave differently from one another) and mechanisms like tagging (each agent has its name and recognize it), internal models (agents reason about their world), and building blocks (each component is composed by simpler components) provide the basic tools to design agent-based models. Simple rules result in emergent organization and complex behaviours.

Examples of ABMS can be found in a very wide range of sciences and provide a method to reformulate certain questions in the social and natural sciences. For instance, in **social sciences**, agents represent people or groups of people that use cognitive psychology and game

theory to rationalize the strategies to establish relations; agents' relationships represent processes of social interaction (Gilbert and Troitzsch, 2005). In **anthropology**, recent developments are being done with large-scale agent-based simulations of ancient civilizations to help explain their growth and decline, based on archaeological data. The study reported by Kohler et al. (2005) attempts to examine the collapse of the Classical Maya civilization, mimicking the processes of population growth and resource usage and see how well the software's predictions coincide with the archaeological record. In **sociology**, attempts have been made to model social life as interactions among adaptive agents who influence one another in response to the influences they receive. In **linguistics**, ABMS have been used to provide the agents with language and to organize communication protocols. In **cognitive science**, agent-based models of emotion, cognition, and social behaviour are being developed based on the notion that a person's emotional state impacts its behaviour and its social interactions. **Economic sciences** were, perhaps, the field in which AIBM had more influence, and changed some classical assumptions of standard micro-economy (Macal and North, 2006):

- Economic agents are rational and are able to optimize their behaviour – do individuals and organizations really optimize? Observations on people and organizations made by Herbert Simon, Nobel Laureate in Economics (1978) and with pioneer works on the field of Artificial Intelligence, led him to develop the notion of “satisficing” (satisfy with suffice) (Manktelow, 2000), with agents choosing what even not being optimal will make them happy enough.
- Economic agents are homogeneous, having identical characteristics and rules of behaviour – agent diversity universally occurs in real-world and many natural organizations, from ecology to industry, are characterized by populations whose diversity maintains their stability and robustness.
- There are decreasing returns to scale from economic processes, decreasing marginal utility, marginal productivity, etc. – dynamic processes of rapid exponential growth in economic systems showed that positive feedbacks can create self-sustaining processes that quickly take a system from its starting point to a faraway state.
- The long-run equilibrium state of the system is the primary information of interest – the transient states that are encountered along the way to a long-run state are often of interest, and not all systems come to an equilibrium.

One permanent issue is the agents' interactions with a collective environment. The **ecology**, for which the environment is a fundamental question, must play a key role in specifying

concepts and developing appropriate tools, despite the scientific disciplines mobilized to examine this problem.

The rich breeding ground of this interdisciplinary movement interested several research groups to use multiagent systems in different ways, so multiagent systems are now a generic term (Ferber, 1999) for:

- Interacting physical agents (collective robotics);
- Systems of interactive software agents (softbots) used in distributed planning tasks, scheduling applications, and search engines;
- Simulations of multi-agent systems, also called multi-agent simulations.

The concepts and techniques of the last item are widely used in the ecological modelling practices, and references to ecology are frequent on the multi-agent simulation field. Works on interaction structures in bumble bee colonies, reported from Hogeweg and Hesper (1983), and the Craig Reynold's "Boids" model (Reynolds, 1987), which imitate the behaviour of groups of migrating birds, were experienced before the notion of MAS but the methodology behind the works seems to use the same approach. The first works registered representing the notion of reactive agents and emergent behaviour in the society are related with ant colonies and reported by several authors during the first years of 90's decade, namely Drougol and Ferber (1992). Environmental applications use MAS to model the interactions between social dynamics (society) and natural systems, normally devoted to ecosystem management like water management (Lansing and Kremer, 1993) or fisheries (Bousquet et al., 1994).

According Bousquet and Le Page (2004), with MAS and ecosystem management the paradigm shifts from "dynamics under constraints" to interactions, from a systemic to an organizational point of view and, in modelling tools, from stocks and flows to behaviour and interactions.

Agent modelling concerns with modelling agent interactions, as much as modelling agent behaviours, and the primary issues of agent interactions are who is connected to who and the mechanisms governing the nature of the interactions. The topology of the network, where the agents belong, defines the neighbourhood of each one and the potential social interaction patterns. Traditionally, the social networks did not change their structure over time or as a result of agent actions (static networks). But the real-world networks change their configuration and it is important to understand why they expand or shrink. Dynamic network analysis is a new field that incorporates the mechanisms of network growth and change, based on agent interactions processes – the objective is to understand the agent rules that govern

how networks are structured, and how quickly the information roam through the network (Macal and North, 2006).

3.3 Intelligent Optimization Techniques

Researchers involved with ecological systems are used to study and deal with problems of great complexity, derived mainly by the number of independent variables for which it must assign a value (instantiation) – the decision variables. When those problems are integrated with management issues, it is important to find the **best solution** from some set of available alternatives, the one for which a given criteria or objective function evaluates to an optimum value (maximum or minimum). Typically, the admissible values for the decision variables are constrained through a number of conditions, and the value assigned to some variables can constraint the value of others. A **solution** is an instantiation of values for all decision variables. The set of available alternatives is defined as the set of all feasible solutions or the **solution space**, and the number of independent variables that need an instantiation value will determine the complexity of the solutions.

This is known as an *optimization problem* and can be formalized in several ways. Following the formalization enounced by Moreira (2008) the optimization problem (P) can be defined as the tuple:

$$P = (X, D, C, f) \quad (5)$$

where:

$X = \{x_1, \dots, x_n\}$ is the set of decision variables;

$D = \{d_1, \dots, d_n\}$ is the variables' domains;

$C = \{c_1, \dots, c_m\}$ is the set of constraints among variables;

n is the number of decision variables;

m is the number of constraints;

f is the objective function, with $f: d_1 \times \dots \times d_n \rightarrow R$

The solution space (S) is defined as the set of all feasible solutions:

$$S = \{s = \{(x_1, v_1), \dots, (x_n, v_n)\} \mid v_i \in d_i, s \text{ satisfies } \forall c_k \in C\} \quad (6)$$

The optimization problem is solved when a feasible solution $s^* \in S$ satisfies the condition:

$$\forall s \in S, \exists s^* \in S: f(s^*) \leq f(s) \quad - \text{minimization} \quad (7)$$

$$\forall s \in S, \exists s^* \in S: f(s^*) \geq f(s) \quad - \text{maximization} \quad (8)$$

When the goal is to minimize the objective function it is used the formalization given by equation 7 – in this case, the objective function is referred as the *cost function* or the *penalty function*. Equation 8 is used when the goal is to maximize the objective function, also known as *utility function* or *benefit function*. It is not unusual to have a set of optimal solutions instead of an optimal solution.

There are several techniques and algorithms that can be applied to assist in finding the optimal solution, and their application depends on the type of problem under study. According to the variable's domain values, the optimization problem can be divided in two main categories: those where solutions are continuous (encoded as real numbers) and those where solutions are discrete (discrete values). The group of **combinatorial problems** where the objective is to find an optimal combination of solution components from a countable (finite or infinite) set, is considered a subset of the latter category, even if some of the decision variables have a continuous domain.

The general approach to solve difficult combinatorial problems is based on **search algorithms** – pick one initial solution from the **solution space** (search space) of the problem and generate and evaluate new solutions in order to find the best one. The generation of the new candidate solutions is what distinguishes the different search algorithms.

The most obvious possible algorithm is to test all the feasible solutions in the solution space and return those (or that) for which the value of the objective function has the minimum (or maximum) value. It is a very simple algorithm to implement using a systematic generation of all the solutions, guarantying the completeness of the search. This method is known as **brute-force search** or **exhaustive search** and is only practicable when the search domain is relatively small – as the instance size of the search domain increases, the run time to find the best solution increases exponentially, making it necessary to elaborate more refined algorithms that turns possible, in polynomial or bounded time, to reduce the number of generated and evaluated solutions without compromising the completeness of the solution.

All algorithms that run in polynomial time systematically prune the solution space, discarding those parts for which there is evidence that the solution cannot be there. Methods like branch and bound (Lawler and Wood, 1966; Realff and Stephanopoulos, 1998), A* algorithm (Hart et al., 1968) or dynamic programming (Bellman, 1957; Bellman, 1958; Bellman, 2003) proved to be more efficient than exhaustive search, but cannot provide a solution in polynomial time in certain problems.

Another obvious approach is **random search** that perambulate from the search space by picking randomly candidate solutions. It is a very inefficient method that does not guarantee completeness, but that is used initially by almost all other algorithms to generate the first or initial solution, and is the foundation of the **stochastic search** - used with some guidance is a very powerful method to successfully find a solution.

The paradigm of local search is to explore the neighbourhood of a given solution in order to find a better solution, improving it iteratively. The new solution is derived from the current one by a user-given **mutator procedure**. It is simple and can easily drive to a good solution, but the success of the search is directly related with the generation of the neighbours - the mutator procedure. This strategy executes a move to a neighbour if and only if the objective function progress in the right direction, stopping when no improvement is possible. Algorithms using this approach must be advised to detect and overcome local optima solutions, where the value of the objective function is better than in any of its neighbours, but is not the global optimum value.

The well known **Hill Climbing algorithm** (*uphill* or *downhill*, depending on whether the objective function pretends maximization or minimization) is easily coded but may lead to poor quality solutions, if the search space has many local optima or large uplands (*plateaux* – regions where all solutions generate the same value for the objective function). In this case, several runs of the algorithm can be executed, starting always with a random initial solution and analyzing the results. Accepting non-improving solutions to continuing, or varying the size of the neighbourhood, are alternative methods to surpass the existence of local optima.

3.3.1 Metaheuristics

A metaheuristic is a high-level algorithmic technique specialized to solve a specific optimization problem where it is hard to find an optimal solution in a bounded time (Black, 1998). It can also be seen as a high-level strategy approach that guides other heuristics in search for a feasible solution. A good **heuristic** usually indicates a shortcut to the right solution, giving a better solution and is, generally, very problem-dependent.

From the dictionary a heuristic is:

“a commonsense rule (or set of rules) intended to increase the probability of solving some problem.” [wordreference.com]

“a usually speculative formulation serving as a guide in the investigation or solution of a problem.” [dictionary.com]

“a problem-solving technique in which the most appropriate solution, of several found by alternative methods, is selected at successive stages of a program for use in the next step of the program.” [dictionary.com]

More sophisticated definitions, directed to optimization problems, can be found in the literature. Two definitions by Reese and Pearl were referenced in Moreira (2008):

“... heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises from two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices. A heuristic may be a rule of thumb that is used to guide one’s action.” (Pearl, 1984)

“...technique which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.” (Reeves, 1993)

Algorithms that use probabilistic decisions could be much more powerful if they associate more than one heuristic to their decision-making processes in order to find good solutions, decreasing the computational time to produce high quality solutions – this strategy is known as **metaheuristic**. Typically, algorithms that use sophisticated metaheuristics maintain a pool with several candidate solutions and memorize old solutions, avoiding wasting time with repeated solutions. There is no commonly accepted definition for the term metaheuristic, but in the last few years some researchers tried to propose a definition:

“A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.” (Osman and Laporte, 1996)

“A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions

at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method.” (Blum and Roli, 2003)

“Meta-heuristics are widely acknowledged as essential tools to address difficult problems in numerous and diverse fields (...). In fact, meta-heuristics often offer the only practical approach to solving complex problems of realistic scale. (...) Heuristics do not, in general, guaranty optimality. Moreover the performance often depends on the particular problem setting and data. Consequently, a major issue in meta-heuristic design and calibration is not only how to build them for maximum performance, but also how to make them robust, in the sense of offering a consistently high level of performance over a wide variety of problem settings and characteristics.” (Crainic and Toulouse, 2003)

Compared to exact search methods, such as branch-and-bound, metaheuristics cannot ensure a systematic exploration of the entire solution space, but they attempt to examine only parts where, according to certain criteria, the researcher believes good solutions may be found. Well designed metaheuristics avoid getting trapped in local optima or cycling over visited solutions and provide reasonable assurance that the search has not overlooked promising regions.

Like iterative searches, metaheuristics for optimization problems may be described as a “walk through neighbourhoods” (Crainic and Toulouse, 2003), a search trajectory through the search space of the problem at hand. Movements from a given solution to another on the neighbourhood, unlike the classical heuristics, do not necessarily improve the solution. Methods like Tabu Search (Glover, 1986, 1989, 1990; Glover and Laguna, 1997) and Simulated Annealing (Kirkpatrick et al., 1983; Siarry et al., 1997), or a mix of them (Mishra et al., 2005), implement one move at each iteration, while Evolutionary Computation/Genetic Algorithms (Holland, 1975) may generate several new candidate solutions (individuals) at each iteration (generation). The two last algorithms may inflect drastically the search trajectory of the solution, creating new “active” neighbourhood.

The fundamental desirable properties of metaheuristics include (Hansen and Mladenović, 2001; Blum and Roli, 2003):

- *Simplicity*: Metaheuristics are strategies that guide the search process on a simple and clear principle which should be widely applicable.
- *User-friendliness*: Heuristics should be well defined, easy to understand and easy to use.

- *Effectiveness*: Metaheuristic goal is to efficiently explore the search space in order to find good solutions (near-optimal solutions) in bounded CPU time.
- *Efficiency*: Metaheuristic algorithms are approximate and usually non-deterministic, and may incorporate mechanisms to avoid getting trapped in confined areas of the search space, providing optimal or near-optimal solutions for most realistic instances.
- *Coherence*: Metaheuristic algorithms range from simple local search procedures to complex learning processes, and the various steps of heuristics follow naturally from the principle of the metaheuristic.
- *Robustness*: Metaheuristics are not problem-specific and the basic concepts permit an abstract level description for various problems, giving good solutions for a variety of instances in each of these problems.
- *Innovation*: Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by an upper level strategy, leading to new type of applications.

Although each metaheuristic algorithm has its own behaviour and characteristics, a generic procedure can be defined, sharing some fundamental components and common elements (Crainic and Toulouse, 2003):

1. Initialization – a method to create an initial solution - s ;
2. Neighbourhood – each solution has a set of corresponding neighbours - $N(s)$;
3. Neighbourhood selection criteria – when more than one neighbourhood is defined, a rule must exist to decide which one will be selected;
4. Candidate selection – only one subset of moves is examined at each iteration. A selection criterion must specify how solutions are picked for inclusion in the candidate list;
5. Acceptance criterion – moves are evaluated and the best solution is selected;
6. Stopping criteria – metaheuristics must impose the criteria to stop the search: computing time, number of iterations, rate of improvement, etc. More than one criterion may be defined to control various phases of the search.

This generic metaheuristic procedure describes the main classes of metaheuristics: Evolutionary Computation (EC), Simulated Annealing (SA), and Tabu Search (TS), considered by some authors as belonging to the *local stochastic search algorithms*.

In short, metaheuristics are high level strategies for exploring search spaces by using different methods, where a dynamic balance between *diversification* and *intensification* must remain. **Diversification** refers to the broad exploration of the search space and **intensification** refers to exploitation of regions with high-quality solutions. The terms were originally used by Glover and Laguna (1997) and their initial meaning becomes more and more accepted by the whole field of metaheuristics. The balance between diversification and intensification is important to quickly identify regions with high-quality in the search space, and not to waste too much time in regions of the search space already explored or which provide low-quality solutions (Blum and Roli, 2003).

The next sections describe some well known optimization algorithms, namely the individual based, or trajectory methods, and the population based methods. The selection of the algorithms was influenced by their usage, or intended usage, in problems like ecosystem management and optimization.

3.3.2 Individual Search Metaheuristics

Search methods for optimization like Simulated Annealing, Tabu Search, Greedy Randomized Adaptive Search Procedure, or Variable Neighbourhood Search can be grouped in the Individual Search Optimization Algorithms or Trajectory Methods.

These methods work on one or several neighbourhood structures imposed on the members (solutions) of the search space. The term **trajectory methods** is also used because the search process performed by these methods is characterized by a trajectory in the search space, where a successor solution may or may not belong to the neighbourhood of the current solution. The algorithm starts from an initial solution (the initial state) and describes a trajectory in the search space, whose dynamics depends on the strategy used.

Simulated Annealing and Tabu Search algorithms described are the basic local search algorithms, with simple strategies, and the latter ones use more general complex and explorative strategies that may incorporate other algorithms as components.

Simulated Annealing (SA)

Simulated Annealing (SA) has inherited the name inspired by the steel cooling technique thermal annealing, which aims to obtain perfect crystallizations by a slow enough temperature reduction to give atoms the time to attain the lowest energy state – solid materials are first heated past melting point, and then gradually cooled back to a solid state. The *rate of cooling*

directly influences the structural properties of the final product. A detailed analogy with annealing in solids provides a framework for optimization of the properties of very large and complex systems. SA is an algorithm technique to find a good solution by trying random variations of the current solution. A worse variation is accepted as the new solution with a probability that decreases as the computation proceeds. The slower the cooling schedule, or rate of temperature's decrease, the more likely the algorithm is to find an optimal or near-optimal solution (Kirkpatrick et al., 1983).

SA algorithm attempts to use the same metaphor, and similar statistical process, to guide the search through the search space (Kirkpatrick et al., 1983; Eglese, 1990; Siarry et al., 1997). The **temperature control** determines the probability of accepting worse solutions to continue, escaping from local optima; the **cooling schedule** controls how this probability evolves: while the temperature is high (initially) many non-improving solutions are accepted, but as long as temperature decreases very few worse solutions are accepted towards the end.

One major disadvantage that makes sometimes awkward the use of SA, is its extremely slow convergence in very complex nonlinear optimization problems. But the structure of the algorithm facilitates its parallelization and there are several successful parallel SA implementations that greatly speedup its performance and generation of results (Ram et al., 1996; Crainic and Toulouse, 2003).

Tabu Search (TS)

The ability to memorize previous tested solutions can greatly improve the convergence to a solution in space search algorithms. Because SA is a repetitive process, it is possible to store and process information from previous solutions to guide and enhance its efficiency. This is the base concept of **Tabu Search (TS)** – the memory-based strategy used to guide the search phases of the procedure (Glover, 1989, 1990; Hansen and Jaumard, 1990; Glover and Laguna, 1997; Youssef et al., 2001). Pure TS heuristics explore locally the domain by moving from one solution to the best available solution in the neighbourhood – inferior quality solutions are accepted only as a strategy to escape from local optima. Short-term tabu memory records recent visited solutions to avoid repeating or inverting the search. Medium or long-term memories can record statistical information relative to the solutions found so far, to learn about the solution space (e.g. frequency of certain attributes in the best solutions) and guide the search. Moves based on these types of memories are named **intensification** (search

around a good solution) and **diversification** (search in a region of the solution space not yet explored).

Variable Neighbourhood Search (VNS)

The **Variable Neighbourhood Search** (VNS) is a metaheuristic proposed in Hansen and Mladenović (1999, 2001, 2003), which applies a strategy based on dynamically changing neighbourhood in the search. Using systematically this idea and a local search routine, leads to a new metaheuristic adapted to be widely applicable.

Contrary to other metaheuristics based on local search methods, like TS, VNS does not follow a *trajectory* but explores increasingly distant neighbourhoods of the current incumbent solution, and jumps from this solution to a new one if, and only if, an improvement has been made (Hansen and Mladenović, 2001). A local search algorithm is applied repeatedly to get from these neighbouring solutions the local optima. It is possible to use several neighbourhoods, constructing different neighbourhood structures and performing systematic searches.

In the initialization step of the VNS, a set of neighbourhood structures must be defined to start the search. The neighbourhoods are arbitrary, but often a sequence $|N_1| < |N_2| < \dots < |N_{k_{max}}|$ with increasing cardinality is defined. An initial solution x is generated and a stop condition is determined (e.g. maximum CPU time allowed, maximum number of iterations, maximum number of iterations between two improvements, etc.). While the stopping condition is not met, the main cycle is repeated. The main cycle has three phases: *shaking*, *local search* and *move*. In the **shaking phase**, a solution x' is randomly selected from the k th neighbourhood of the current solution x . The x' solution is the starting point of the **local search**, that can use any of the neighbourhood structures, and supplies, at the end, a new solution x'' that is compared with x . If it is better, x'' replaces x and the algorithm restarts with $k=1$ (**move**), otherwise k is incremented and a new shaking phase starts with a different neighbourhood. At the end of the main cycle, x is the best solution and is returned (Hansen and Mladenović, 2001).

The objective of the shaking phase is to disturb the solution providing a good starting point for the local search. When no improvements are achieved, the process of changing neighbourhoods supplies a diversification to the local search – possibly moving from a “bad” place to a “good” place in the search space and evolving to better solutions.

Reduced VNS (RVNS), Variable Neighbourhood Decomposition Search (VNDS) and Skewed VNS (SVNS) are some known extensions of the VNS aiming to solve more efficiently very large problem instances.

RVNS intended to increase speed (or effectiveness) at the possible cost of an increase in solution value. As local search routine is the most time-consuming phase of the basic VNS, in RVNS this phase is simply discarded (Hansen et al., 2001). Solutions are drawn at random in increasingly far neighbourhoods of the best-so-far solution and replace it if and only if they give a better objective function. RVNS is very helpful when the quickly obtainment of a good solution is desirable, even if it is not necessarily very close to the optimum.

VNDS includes a heuristic and a modification of the basic scheme within a successive approximations decomposition method (Hansen et al., 2001). For a given solution x , all but k variables are fixed in the local search phase. As in VNS, the search phase starts with a random solution x' , within which one unfixed variable is chosen at random; instead of performing local search in the whole solution space, with x' as the starting point, the search is done in the one-dimensional space of the unfixed variable that has been chosen. The new value for this variable is returned into the solution x'' and the objective function is updated. The move phase of the VNS is maintained: if the new solution is better than the incumbent, then a move is made (x is replaced by x'') and $k=1$ again; otherwise, k is incremented and a look for improvements is restarted now in the subspace with two unfixed variables, and so on. The only difference between the basic VNS and VNDS is in the local search phase – instead of applying the local search method in the whole solution space, at each iteration a sub-problem is solved in some subspace of the neighbourhood.

SVNS provides an efficient exploration of the “valleys” (or “mountains”) far from the incumbent solution. A good source of information for understanding how heuristics work could be the study of the topology of local optima and the corresponding valleys (or mountains) of the objective function of the problem considered. A study from Hansen et al. (2000) showed that local optima tend to coalesce at the bottom of large valleys (or at the top of large mountains).

When there are several valleys (or mountains) the VNS has a problem: few iterations will quickly determine the best solution in a large region (the bottom of a large valley, or the top of a large mountain in that region), but it will be very difficult to find a better solution near it. When the best solution is found in a large region it is necessary to go quite far to obtain an improved one. In other words, when the basic VNS moves far from incumbent solution, it

tends to degenerate into multi-start, which is not a good metaheuristic. One way, over many, to overcome this problem is to make some compensation for distance of the new solution from the incumbent solution; the test for acceptance of a move will evaluate the objective function compensated with that “measured” distance – this is the scheme for the skewed VNS (Hansen et al., 2000; Hansen and Mladenović, 2003).

Changes in the neighbourhood in the search for good solutions is a simple and powerful tool, and several authors have integrated other metaheuristics with the VNS method. Example of the alternatively use of VNS and tabu search to solve the Nurse rostering problem is found in Burke (2004). A multi-level TS is proposed in Kovačević-Vujčić et al. (1999) to solve continuous min-max problem, where each level represents a ball of different size in the vicinity of the current solution, i.e., different neighbourhoods, and a tabu list is constructed for each level.

Greedy Randomized Adaptive Search Procedures (GRASP)

Most of the problems found in industry and government are computationally intractable, by their nature, or sufficiently large to preclude the use of exact algorithms, and the use of heuristic methods are usually employed to find good solutions, not necessarily optimal. The **Greedy Randomized Adaptive Search Procedures** (GRASP) were developed during the 1980's (Feo and Resende, 1989) with the objective of solving, in bounded time, complex combinatorial optimization problems. These methods are based on empirical observed behaviours especially in industries.

A GRASP is an iterative process consisting of two phases: a *construction phase* and a *local search phase* or *improvement phase*. The best-so-far solution is kept as the result (Feo and Resende, 1995). In the **construction phase** a feasible solution is iteratively constructed, one element at a time, based on a greedy function that measures the benefit of selecting each element and order all elements in a candidate list – the *restricted candidate list* (RCL). The heuristic updates the benefits associated with every element, reflecting the changes brought on by the selection of the last element. A probabilistic component randomly chooses one of the best candidates, not necessarily the top candidate. The solution is then improved via local search, the list of solutions updated, and the iterative process is repeated until a termination criterion is satisfied, such as a maximum number of iterations have occurred, or a maximum computation time is attained.

The **local search phase** is launched in each iteration by successively replace the current solution by a better solution in the neighbourhood, and it terminates when no better solution

is detected in the neighbourhood. The local search phase can require exponential time from an arbitrary starting point so an efficient construction phase, which produces good initial solutions, can improve the efficiency of the improvement phase (Feo and Resende, 1989).

The randomized nature of the greedy construction phase - also referred as semi-greedy or randomized greedy (Hart and Shogan, 1987) - often generates a wide variability of solutions in the same amount of time, which facilitates the parallelization of the algorithm, and the obtainment of multiple solutions. The best of these GRASP solutions is generally significantly better than the single solution obtained by a single random starting point (Feo and Resende, 1995). Moreover, parallelization allows the use of different strategies for local searches, that can be improved with sophisticated techniques and algorithms, like those mentioned in the previous sections (Hart and Shogan, 1987).

Due to its random adaptive nature, it is intuitive that the usual way to increase the quality of the best-so-far solution is to augment the number of iterations.

3.3.3 Population Search Metaheuristics

Search methods for optimization like Evolutionary Computing (Genetic Algorithms, Scatter Search) or Ant Colony Optimization can be grouped in the Population Search Optimization Algorithms. These methods follow a different philosophy than the previously mentioned (section 3.3.2) – they incorporate a learning component in the sense they implicitly or explicitly try to learn correlations between decision variables to identify high-quality areas in the search space.

Population-based methods deal, in every iteration of the algorithm, with a set of solutions rather than a single solution, providing a natural and intrinsic way for the exploration of the search space. Obviously, the final performance of the algorithm depends strongly on the way the population is manipulated. In the Evolutionary Computing algorithms a population of individuals is modified by recombination and mutation operators, and in the Ant Algorithms a colony of artificial ants is used to construct solutions guided by pheromone trails and heuristic information.

Genetic Algorithms (GA)

Genetic algorithms (GA) belong to the larger class of evolutionary methods and were inspired by the evolution processes of biological organisms. The extraordinary book “On the Origin of Species”, from Charles Darwin, edited for the first time in 1859, introduced a new paradigm in

biological sciences about species evolution and natural selection - when natural populations are studied over many generations, they appear to evolve according to the principles of natural selection and survival of the fittest to produce “well adapted” individuals (Crainic and Toulouse, 2003). The algorithms that mimic this biological process are the Genetic Algorithms and are referred by numerous researchers (Holland, 1975; Cardon et al., 2000; Youssef et al., 2001; Amirjanov, 2006). With the evolution of the computers’ hardware and software the paradigm was enriched with hybridization and local heuristics.

The basic idea follows closely the philosophy of the theory of evolution: an **initial population** of solutions evolves by generating new individuals out of combinations of existing individuals. At each iteration, some individuals from the current population, are selected as parents for the generation of new individuals (**reproduction**) that will improve the population. The candidate selection of the parents is based only on the value of some **fitness function**, and new individuals are generated by the appliance of the **crossover operator** over the selected parents. **Mutation operators** intervene to modify the definition of individual characteristics of the new generation to improve their fitness and the diversity of the population. A **survival operator** is applied finally to determine which of the parents and offsprings advance to the next generation – in principle the more adapted to the environment. Good heuristics in the operators will loop through iterations, gradually converging to better solutions. The criteria for the algorithm’s termination can be the achievement of a maximum number of generations, the achievement of a sufficient fitness within a population, or the attainment of a maximum computation time. The GA terminology equates generation with one iteration of the algorithm, chromosome with a single solution, and population with the group of chromosomes currently active.

GA are easily integrated with other optimization algorithms, like hill climbing, furnishing schemes to move the population away from local optima.

Scatter Search (SS)

Scatter search (SS) and its generalized form called Path Relinking (Glover et al., 2000) is an evolutionary approach originated from strategies for creating composite decision rules and surrogate constraints. The scatter search methodology differs from other evolutionary procedures, like genetic algorithms, by providing unifying principles for joining solutions based on generalized path constructions (in both Euclidean and neighbourhood spaces) and by utilizing strategic designs where other approaches resort on randomization (Glover et al.,

2000, 2003). Foundations of scatter search are intimately related to tabu search metaheuristic, supplying SS additional advantages based on intensification and diversification that explore the use of adaptive memory and associative memory mechanisms.

Scatter search operates with a population of feasible solutions, the **reference set**, like other evolutionary methods, and employs procedures for combining these solutions to create new ones, the **trial solutions**. The most distinguishable feature of the evolutionary approaches is its intimate association with the tabu search (TS) metaheuristic, and its adoption of the principle that search can benefit by incorporating special forms of adaptive memory, along with procedures particularly designed for exploiting that memory. Unlike “populations” in genetic algorithms, the reference set of solutions in SS is relatively small, typically with 20 solutions or less.

One of the main aspects of SS is the creation of composite decisions, the manner in which it combines solutions and undertake advantages to exploit these combinations (Glover et al., 2000). The idea is to generate new rules by creating numerically weighted combinations of existing rules suitably restructured, so that their evaluations embodied a common metric. Information about the desirability of alternative choices is captured in different forms by different rules, and this can be effectively explored integrating a mechanism that combines several rules to generate new ones, instead of using the standard strategy of selecting different rules one at a time. Additionally, after reaching a local optimum, the method continues varying the values of the parameters to build combined rules and producing additional trial solutions.

Another important aspect of SS is the introduction of associated procedures to combine constraints – a mechanism that generate weighted combinations, called surrogate constraints (Glover et al., 2000). Subsets of constraints, considered to be most critical relative to trial solutions, are isolated and produce new weights that reflect the degree to which the component constraints are satisfied or violated. The principal function of the surrogate constraints is to provide ways to evaluate choices that could be used to generate or modify trial solutions (Glover et al., 2003).

In evolutionary algorithms two or more solutions are randomly chosen from the population and crossed over and combined to generate one or more offsprings. In contrast, SS chooses two or more elements of the reference set in a systematic way with the purpose of creating new solutions, guarantying the injection of diversity in the reference set.

The basic SS template could be resumed as follows (Glover et al., 2000, 2003):

- Generate a set of starting solution vectors, guarantying a certain degree of diversity, and define heuristic processes as an attempt to improve these solutions. Designate a subset of the “best” solution vectors to be the reference set (best as synonymous of diverse).
- Create new solutions as structured combinations of subsets of the current reference solutions.
- Apply the heuristics defined in the first step to improve the solutions generated in the last one. The result of this step may be feasible and unfeasible solutions.
- Extract a collection of the “best” improved solutions from the previous step, and add them to the reference set. The notion of the “best” must be broad, making the objective value one among several criteria for evaluating the merit of the newly created solutions. Repeat the last three steps until the reference set does not change.
- Diversify the reference set, restarting from the first step. The process ends when a specified iteration limit is achieved.

The structured combinations of the SS must be designed with the goal of creating weighted centres of selected subregions, generating dispersion patterns that project new centres into regions outside the original reference solutions, facilitating the desired diversity.

The strategies used to select the subsets that will be combined in the second step of the algorithm are designed to make use of a type of clustering that allow the construction of new solutions within and across the clusters.

The scatter search can be briefly referred as the join of the following methods (Glover et al., 2000, 2003):

- A *diversification generation method* – to generate a collection of diverse trial solutions.
- An *improvement method* – to transform a trial solution into one or more enhanced trial solutions.
- A *reference set update method* – to build and maintain a reference set consisting of the best solutions found (typically no more than 20), guarantying quality and diversity of the membership.
- A *subset generation method* – to produce a subset of the reference set solutions as a basis for creating combined solutions.

- A *solution combination method* – to transform a given subset of solutions produced by the subset generation method into one or more combined solution vectors.

Ant Colony Optimization (ACO)

One imaginative nature inspired metaheuristic was introduced in the 90's by Marco Dorigo and colleagues (Dorigo et al., 1991, 1996; Dorigo and Gambardella, 1997) to help achieving solutions for hard combinatorial optimization problems. An analogy with the way ant colonies function when looking for food has suggested the definition of the Ant System. **Ant Colony Optimization (ACO)** is one of the algorithms that belong to the ant colony algorithms family, covered by the **Swarm Intelligence** methods (Bonabeau et al., 1999), and is a probabilistic technique originally developed to search the optimal path in a graph.

Ant algorithms are multi-agent approaches to solve difficult combinatorial problems, like travelling salesman problem (TSP) or quadratic assignment problem (QAP), and were inspired by the observation of real ant colonies, in particular the interesting foraging behaviour of ant colonies and how ants can find shortest paths between food sources and their nest (Dorigo et al., 1999). The complex and well structured social behaviour revealed by the ant colonies emerges from relatively simple behaviours of the colony's individuals.

While walking from the nest to food sources, and vice-versa, real ants deposit on the ground a substance called pheromone (in varying quantities), forming in this way a pheromone trail, or a marked path. When an isolated ant detects a previously laid trail it can decide, with high probability, to follow it, reinforcing the trail with its own pheromone. The collective behaviour that emerges is a form of *positive feedback loop*, where the more ants follow a trail, the more attractive that trail becomes for being followed (Dorigo et al., 1991) – the probability with which an ant chooses a path increases with the number of ants that previously choose the same path. This basic behaviour is the basis for a cooperative interaction which leads to the emergence of shortest paths.

The **ant algorithms** are models derived from the study of artificial ant colonies, and are used as an optimization tool. They are based on a parameterized probabilistic model – the **pheromone model** – that is used to model the chemical pheromone trails (Blum and Roli, 2003). The artificial ants have some memory, they are not completely blind, and live in an environment where time is discrete. Solutions are constructed incrementally by the artificial ants, adding opportunely defined solution components to a partial solution under consideration. Results obtained with ant systems emphasize the presence of synergetic effects in the interaction of

ants and the quality of the solutions increases when the number of ants working on the problem grows, up to reach an optimal point.

The communication and cooperation between ants is possible by distributed memory implemented as pheromone deposited on edges of a graph. In any step of the algorithm each ant gives a sign of its activity by modifying the problem representation, changing the probability with which the same decision will be taken in the future. If an ant chooses between different options and the choice is being particularly good, then in future that choice will appear as more desirable than it was before. In the early stages of the process a heuristic should help the ants' decision but, as the experience gained by ants increases, that heuristic loses importance and more relevance is potentiated to the problem representation. In each step of the algorithm one procedure must update the pheromone values according to the quality of the solutions constructed. The basic method is to update all pheromone values accordingly to a pheromone evaporation rate. This is necessary to avoid the premature convergence of the algorithm to suboptimal regions of the search space, and it can be seen as a forgetting mechanism that allows the exploration of new regions in the search space, favouring the diversification strategy.

The swarm simulation system (Minar et al., 1996) was, perhaps, the first platform that made intensive use of the ant algorithm concepts – it is a multi-agent software platform for the simulation of complex adaptive systems, where the basic unit of simulation is an object called a “swarm” that contains a collection of agents executing a schedule of actions, and the representation of time. Swarms can be themselves agents and can be part of other swarms.

The **Ant Colony System** (ACS) (Dorigo and Gambardella, 1997) is a version of the ACO framework that has very good performance derived by two major differences in the algorithm implementation. The first one is related with the update of the pheromone trails – it is done offline, after all the ants had built a solution. Pheromone is added to the arcs used by the ant that found the best solution since the start of the algorithm. The second difference is in the way the ants decide which component to move next – the rule is called pseudo-random-proportional: while some moves are chosen deterministically (in a greedy manner), other moves are chosen by the usual rule - in a probabilistic way. These method forces ants update pheromone only online step-by-step, favouring the emergence of other solutions than the best so far.

Another well-known algorithm that extends the ACO metaheuristic is the **MAX-MIN Ant System** (*MMAS*) (Stützle and Hoos, 1997, 2000). Like ACS, the pheromone trails are only

updated offline, receiving additional pheromone the arcs that were used by the best ant iteration or the best ant since the start of the algorithm. The pheromone values are restricted to an interval $[\tau_{min}, \tau_{max}]$ and all the pheromone trails are initialized to their maximum value τ_{max} , preventing that the probability to construct a solution falls below a certain value greater than zero, never vanishing the chance of finding a global optimum.

Due to its simplicity, there were many successful implementations of parallelized versions of the algorithm – one example may be found in Stützle (1998). Since its genesis, the distribution of search activities over many simple agents (ants), that use very simple basic actions, eases the parallelization of the computational effort, exploring the emergence of global properties from the interaction of the agents.

3.3.4 Optimization Metaheuristics Discussion

The high performance achieved by the great majority of metaheuristics is driven by concepts like **intensification** and **diversification** (introduced earlier in section 3.3.1). Depending on the paradigm behind a particular metaheuristic, intensification and diversification are achieved in different ways. Every metaheuristic approach should be designed with the aim of effectively and efficiently explore the search space, in order to obtain a search algorithm clever enough to intensively explore areas of the search space with high quality solutions – **intensification** - and to move to unexplored areas of the search space when necessary – **diversification** (Glover and Laguna, 1997). These terms were presented in the definition of the Tabu Search metaheuristic and became more and more referenced in other fields. For instance, in Evolutionary Computation the term exploitation is related with intensification, and the term exploration is related with diversification, although based in short term strategies tied on randomness, while the original meaning of the terms emphasizes the notion of high level strategies based on the usage of memory.

Locality is the concept that emerges and gives meaning to intensification and diversification; **locality** implies the notion of area of the search space, or region, and it is something fuzzy because it depends on the characteristics of the search space and on the definition of **distance** in each search space.

Two definitions drawn from the literature about intensification and diversification:

*“The main difference between intensification and diversification is that during an intensification stage the search focuses on examining neighbours of elite solutions.
(...) The diversification stage on the other hand encourages the search process to*

examine unvisited regions and to generate solutions that differ in various significant ways from those seen before.” (Glover and Laguna, 1997)

“Intensification is to search carefully and intensively around good solutions found in the past search. Diversification, on the contrary, is to guide the search to unvisited regions. These terminologies are usually used to explain the basic elements of Tabu Search, but these are essential to all the metaheuristic algorithms. In other words, various metaheuristic ideas should be understood from the viewpoint of these two concepts, and metaheuristic algorithms should be designed so that intensification and diversification play balanced roles.” (Yagiura and Ibaraki, 2001)

There are also strategies explicitly aimed to dynamically change the balance between intensification and diversification during the search. A fairly simple strategy is used in Simulated Annealing (SA), where an increase in diversification and simultaneous decrease in intensification can be achieved by “reheating” the system and then cooling it down again (which corresponds to increasing parameter T and decreasing it again according to some scheme). Such scheme is called non-monotonic cooling scheme. Another example can be found in Ant Colony System (ACS) that uses an additional component to introduce diversification during the solution construction phase - an ant reduces the pheromone values on the nodes of the construction graph that it visits, while it is walking on the graph to build a solution. The effect of this action is reducing the probability of other ants to follow the same path. This update mechanism is called step-by-step online pheromone update rule, and its interplay with the other update rules leads to oscillating balance between intensification and diversification (Blum and Roli, 2003).

Most metaheuristic components have both effects: intensification and diversification. Following the objective function amplify the intensification effect, and diversification is achieved by the use of randomness or by following a criterion other than the objective function.

The appliance of techniques that enable the metaheuristics hybridization is a promising and actual field of research – some of the techniques described in the previous sections are hybridizations. Common forms of hybridizations are:

- including components from one metaheuristic into another – component exchange among metaheuristics;
- enabling two or more algorithms to exchange information - cooperative search;

- integrating metaheuristics and systematic methods.

The **component exchange among metaheuristics** is, perhaps, the most popular method and concerns to the use of trajectory methods in population-based methods. Population methods are very powerful on the concepts of recombining solutions to obtain new ones, facilitating the existence of larger steps in the search space than the steps done by trajectory methods. But, when good solutions appear, mechanisms must exist that influence the search in the hope of finding better solutions between those solutions and the current. One of the strengths of trajectory methods is the more structured way they explore a promising region in the search space, reducing the danger of being close to good solutions but missing them. Thus, metaheuristics hybrids that in some way manage to combine the strengths of population-based methods with the advantages of the trajectory methods are often very successful (Blum and Roli, 2003).

The **cooperative search** (Hogg and Huberman, 1993; Hogg and Williams, 1993) is a loose form of hybridization and consists of a search performed by different algorithms that exchange information about states, models, solutions or other search space characteristics. Cooperative search typically run search algorithms in parallel execution with a varying degree of communication – the algorithms can be different or instances of the same algorithm running with different parameters settings. Nowadays this technique has gained increasing attention because of the potential of parallel implementations of metaheuristics.

Integration of metaheuristics and systematic methods has produced effective algorithms when applied to real world problems. A very successful example is the combination of metaheuristics and Constraint Programming (CP) (Pesant and Gendreau, 1996, 1999). CP enables to model a combinatorial optimization problem constraining admissible values for variables and domains. Every constraint is associated to a filtering condition algorithm that deletes those values from a variable domain that do not contribute to feasible solutions. Constraints are activated as soon as the domain of any variable involved as been changed. Then the filtering algorithm is applied, propagating the constraint through the variables' domain until there are no more values to be removed, or at least one domain is empty. A CP system can be seen as the interaction between components (constraints) which communicate through shared variables (Blum and Roli, 2003).

There are three main approaches for the integration of metaheuristics and systematic techniques:

- A metaheuristic and a systematic method are sequentially applied (or executed interleaved). The solutions produced by the metaheuristic can be used as heuristic information by the systematic search, or the systematic algorithm can generate a partial solution that will be completed by the metaheuristic. It can be seen as an instance of cooperative search with rather loose integration.
- Metaheuristic use systematic methods to efficiently explore the neighbourhood, instead of simply enumerate the neighbours or randomly sampling the neighbourhood. This approach combines the advantages of a fast search space exploration made by the metaheuristic and the efficient neighbourhood exploration performed by the systematic method, and is effectively fruitful when the neighbourhood to explore is very large.
- Introduce strategies or concepts from either class of algorithms into the other. This approach preserves the search space exploration on a systematic method, but sacrifices the exhaustive nature of the search (Harvey and Ginsberg, 1995). The hybridization is usually achieved by integrating concepts developed for metaheuristics (e.g. probabilistic choices) into the systematic methods – using a probabilistic backtracking instead of a deterministic (chronological) backtracking, if the search tree algorithm is used.

Although metaheuristics are different – some of them are trajectory methods (SA, TS, VNS, GRASP) and others are population-based (GA, SS, ACO) – and are based on different philosophies, all the algorithms are based on intensification and diversification to efficiently explore a search space. It is possible to identify subtasks where some metaheuristics perform better than others, opening way to the hybridization of metaheuristics that can boost the performance of the search considerably.

3.4 Machine Learning Techniques

“Machine Learning is the study of computer algorithms that improve automatically through experience.”(Mitchell, 1997)

Machine Learning (ML) is the subfield of AI concerned with programs that learn from experience, i.e., the research on computational approaches to learning. Computer programs typically execute the procedures supplied to them, very efficiently, but do not self-improve with experience. Research in machine learning concerns with building computer programs able

to generate new knowledge or improve available knowledge, previously placed by human instructors, by using input information (Michalski and Kodratoff, 1990).

Intelligent agents are an example of self-learning software where perceptions should be used to improve agent's ability to act in the future, not only to act in the moment (Russel and Norvig, 2002). Learning results from the interaction between the agent and the world (objects, agents and environment), and from the perception by the agent of its own decision-making process.

The importance of the work in machine learning continues to appeal researchers in expert systems development, computer vision, speech understanding, problem solving, autonomous robotic, decision support systems, intelligent tutoring systems and so on. The inclusion of ML in ecological models is a tentative to apply concrete techniques to problems of practical significance, transferring the ML programs from university laboratories to the external world.

Russel and Norvig present a general model for learning agents (Russel and Norvig, 2002) with four essential components (Figure 3-2). While **the performance element** takes perceptions and decides on actions, the **learning element** takes some knowledge about the performance element and some feedback on how the agent is doing, and determines how the performance element should be modified to do better in the future. The **critic element** tells to the learning element how well the agent is doing, based on a performance standard given to the agent, and the **problem generator element** is responsible to suggest actions that will lead to new experiences.

The way how feedback provided to the agent is manipulated, against its actions, determines the kind of learning process used. Common algorithms are the **supervised learning** (agent has one function that predicts the outcome of an action - labelled examples - and the feedback generally tells the agent what the correct outcome is; it improves the function that maps actions to desired outputs), the **unsupervised learning** (agent has no hint about the outcome of an action; it builds a model from scratch – unlabelled examples) and the **reinforcement learning** (agent observes the impact of its action over the environment, and receives a reward or punishment from the environment that guides the learning algorithm in order to maximize the rewards). In the literature the **semi-supervised learning** is also referred as the combination of labelled and unlabelled examples to build the appropriate function for decision.

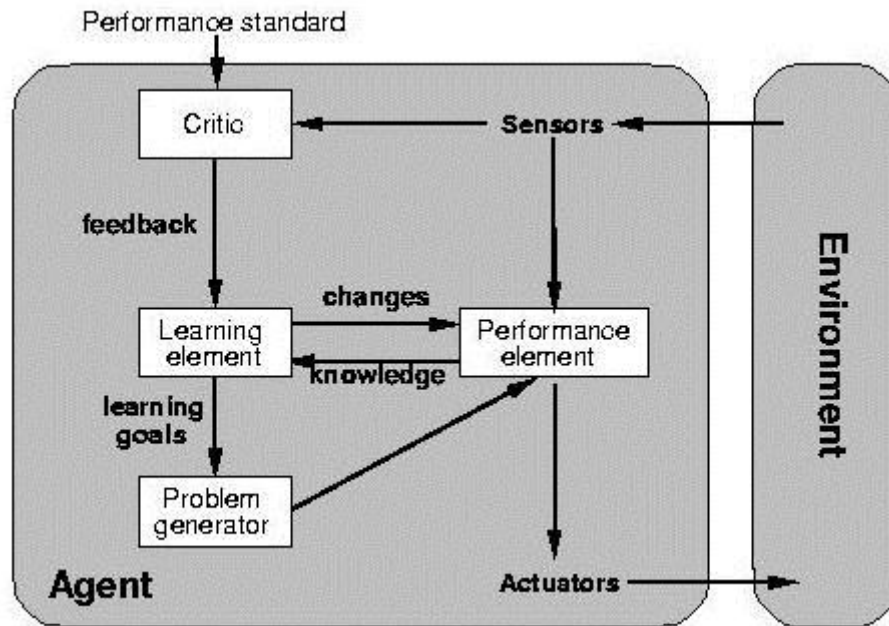


Figure 3-2: General model for learning agent (Russel and Norvig, 2002)

Examples of **supervised learning algorithms** and techniques include, among others, artificial neural networks, Bayesian statistics, case-base reasoning, decision tree learning or nearest neighbour algorithm, and applications where they are applied comprise bioinformatics, information retrieval, pattern recognition, object recognition in computer vision, handwriting and speech recognition, within many others. The typical output of these algorithms could be a class label (in classification) or a real number (in regression).

Artificial neural networks (ANN) are an abstraction from the current understanding of the functioning of the animal nervous system - synapses are simulated by nodes containing a transfer function (usually nonlinear sigmoid). Incident edges are divided into inputs and outputs; the inputs are summed proportionally according to weights attached to each edge, then the output values are generated by the transfer function. The feed-forward multi-layer perceptron, where the edges feed only forward from one level to the following, is the simplest representation and is the most commonly used; the layered architecture could have only a single internal layer. Each ANN must be trained before its usage in order to calculate/distribute the weights attached to each edge, and a wide variety of algorithms are available for minimising the error function; perhaps the most known and used algorithm is the back-propagation, that is a gradient descent algorithm for minimising the error function. Also several research works were carried out in order to reduce the excessive number of neurons and synaptic weights. Hu et al. (1991) relate one of those works where the authors replace the original weight matrix by a product of two smaller matrices so that the number of

multiplications required is reduced. To reduce the hidden units, they exploit the correlation among the outputs of the hidden neurons in the same layer.

The information embedded in a neural network is very machine dependent and this is an obstacle to extract meaningful knowledge from the inner data.

Decision tree induction is a traditional and well-respected method for generating predictive models from data (Quinlan, 1986). It proceeds by recursively partitioning the data according to values of attributes; most algorithms work from the root node of the tree downward, generating progressively more refined definitions for the classes being learnt. The algorithms are usually greedy, partitioning the data at each level based on which attribute gives the best value of whatever criterion is in use to judge the quality of data partitions. In most algorithms, the partitioning is continued until a further heuristic criterion, deliberately designed to generate overfitting, is triggered. The resulting tree is then pruned to give optimum generalisation performance on an independent test dataset. Decision tree induction may be extended to generate decision rules instead: the unpruned decision tree is converted into a logically equivalent rule set, then pruning is conducted on the rule set rather than the tree. Decision rule pruning can permit generalisations which are not available in decision tree pruning, and the resulting rules can sometimes be more comprehensible to human readers. The comprehensibility of decision tree learning comes at a significant cost: the language learnt by decision tree systems is not universal, so it may be impossible for the decision tree system to accurately fit a dataset simply because it cannot represent the information contained in the dataset.

Clustering and dimensionality reduction are, perhaps, the most known classic examples of **unsupervised learning** (Ghahramani, 2004), but it is also included in some artificial neural networks (ANN), blind source separation, generative topographic maps, among others. Applications with unsupervised learning simply receive inputs and do not obtain neither supervised target outputs nor rewards or punishments from the environment. Formal frameworks for unsupervised learning build representations of the input data that can be used for decision making, predicting future inputs, and so on. Even without any feedback from the environment the process uses probabilistic models to find patterns in the input data, and beyond what can be considered pure unstructured noise.

Along with the development of the ANNs, the **Fuzzy Set Theory** has been efficiently used for extracting information from ecological data. Inspired in the biological behaviour it is an extension of the classical meaning of the term “set”, and formulates specific logical and

arithmetical operations for processing imprecise and uncertain information (Zadeh, 1965; Zimmermann, 1987). The fuzzy rule-based models are often employed to capture the approximate mode of reasoning that plays an essential role in dealing with uncertain and imprecise data. The output of each rule is a linear function of the input variables and the final output of all rules is the weighted average of each rule's output. In ecological data, Adriaenssens et al. (2006) used fuzzy knowledge-based models for prediction of macro invertebrates in watercourses, Chen and Hare (2006) used a combination of neural networks and fuzzy logic models for analysis of the pacific halibut recruitment, and Salski and Holsten reported fuzzy and neuron-fuzzy approaches to modelling cattle grazing on pastures with low stocking rates (Salski and Holsten, 2006) or to modelling feeding of Greylag Geese on reed (Salski and Holsten, 2009).

Reinforcement learning (Kaelbling et al., 1996; Sutton and Barto, 1998) is closely related to the fields of decision theory (in statistics and management science), and control theory (in engineering). The fundamental problems studied in these fields are often formally equivalent, and the solutions very close, although different aspects of the problem and the solution are usually emphasized (Ghahramani, 2004). Environments with more than one machine (or computer) with learning capabilities can be much more dynamic, derived from the multiple adapting machines.

3.4.1 Ecological Modelling and Machine Learning Integration

Ecological modelling is concerned with the development of models that deal with the relationships among members of living communities, and between these communities and their abiotic environment. The latter can be highly nonlinear ecological models have to reflect this to be realistic. The ML technique of **artificial neural networks** is widely used to build models of population dynamics of algae (Recknagel et al., 1997), aquatic fauna and water quality (Schleiter et al., 1999), production in coastal areas with upwelling (Barciela et al., 1999), fish abundance and spatial occupancy (Brosse et al., 1999), phytoplankton primary production (Scardi and HardingJr, 1999) and prediction of zooplankton biomass variance (Aoki et al., 1999). However the models produced are like black-box models – hardly to inspect and understand.

The use of **symbolic machine learning** techniques allow the construction of models that can be inspected, modified, used and verified by researchers and experts, and have the potential to become part of the knowledge in the respective application domain. Symbolic ML methods

induce explicitly represented symbolic models from data (Džeroski, 2001) and numerous applications are being used in the field of ecological modelling, namely to model population dynamics - examples that include algal growth in Lagoon of Venice (Kompore and Džeroski, 1995; Kompore et al., 1997b) and in the Slovenian Lake of Bled (Kompore et al., 1997a) and phytoplankton growth in the Danish Lake Glumsoe (Todorovski et al., 1998).

Following this reasoning, Shan et al. (2006) report the use of ML techniques in a research to model a difficult problem – the spatial distribution of an endangered Australian marsupial, the southern brown bandicoot. Spatial phenomena and interaction in the real world are highly intricate, resulting in suspect and poor available theories, which make mathematical models very hard to build because of the absence of a good and strong theoretical knowledge. Four ML techniques – **decision trees/rules, neural networks, support vector machines and genetic algorithms** – were applied to the problem, and the exploration of the datasets gave interesting results. When the data available were highly regular and the effects strong, valuable insights were yielded. The challenges to the ML techniques were more evident when the datasets were small, with noisy data and weak domain theories. The experience reported surveyed over 300 sites in South Australia and registered data for the bandicoot population and surrounding factors, like vegetation, soil, fire history and geomorphology. The ML techniques were used to generate models, aiming to predict the abundance of the bandicoot related with geographical factors. The results obtained showed that decision trees/rules and genetic programming techniques directly yielded models with potentially human-comprehensible meaning, while neural networks and support vector machines did not.

3.5 Decision Support and Management Systems

Human activities have a prominent influence on the evolution of the natural environment, and this is accompanied, in recent years, by the awareness of environmental issues and responsibilities involved. This influence must be considered and integrated in the decision processes by environmental managers.

Managers should check that the decisions are feasible from a political and technical point-of-view, while simultaneously predict the reactions and effects between the biological elements and the environment. They must take into account the different temporal and spatial scale as well as biophysical, social and economic considerations, and resolve potential conflicts of interest. Policy makers must have some knowledge about ecosystem functioning, must manage the accumulated qualitative and quantitative information to adapt their conceptual

models to local management, and must select the appropriate management options to maximize the decision criteria.

Arguments like the previous are the basis for the increasingly developed Decision Support Systems (DSS) applied to environmental management to assist business decisions. The appearance of minicomputers, distributed computing and operating systems opened the way to computerized decision support systems. DSS applications evolve as the computer technology evolves, and the huge number of DSS frameworks, developed for dedicated problems, is almost faced as an image of each research team.

3.5.1 DSS Origins

The origins of the modern assisted decision making and planning are intrinsically related with the evolution of computers. The pioneering documented works that had been performed in the 1960's by several researchers can be referred as the birth of the interactive management decision systems. Particularly important is the experimental work done by Douglas Engelbart and colleagues that developed the first hypermedia/groupware system called NLS (oNLine System) (Engelbart, 1962) where the computer mouse and groupware were first presented. This system used the notion of hypertext, provided facilities for on-screen video teleconferencing and is considered a forerunner to group decision support system, derived from its interactive facilities.

It is generally regarded that the idea of decision support system was started with the work of Gorry and Scott-Morton's published in a Sloan Management Review article (Gorry and Scott-Morton, 1971). In the early years, the systems intended to replace the human decision maker but nowadays it is accepted that the systems focus on decisions and on supporting rather than replacing the decision-maker (Power, 2007).

Although first DSS's began to optimize a single criterion, it soon became clear that the interesting problems had conflicting interests and multidimensional characteristics, where multiple criteria antagonism exists. The modern DSS's embody the capacity of handling multicriteria evaluation methods and powerful tools to solve conflict management.

The difference between an optimization problem, like those related in section 3.3 , and a decision problem as focused in this section, is that while the former is aimed to find the best solution, the latter aims to determine if a solution with certain characteristics is acceptable or not – if it is a good decision. But what is a “good decision” or an “acceptable decision”?

"A decision is a position or opinion or judgement reached after consideration. The act of making up your mind about something." [www.wordreference.com]

From the definition, the decision needs a human actor and implies a rational decision process. The decision making process can be seen as a cognitive process (an outcome of mental processes) that leads to the selection of a course of action among several alternatives. A process of decision making is a process that guides the decision maker through a series of stages, from the identification and analysis of the problem to the formulation and construction of a number of alternative choices, in order to identify and define which of them is the most appropriate. It must explain the theoretical basis that supports each alternative.

In the technical literature there is no single accepted definition for what constitutes a Decision Support System. Some definitions of Decision Support Systems selected from several sources:

"A decision support system (DSS) is both a process and a tool for solving problems that are too complex for humans alone, but usually too qualitative for only computers. Multiple objectives can complicate the task of decision-making, especially when the objectives conflict. As a process, a DSS is a systematic method of leading decision-makers and other stakeholders through the task of considering all objectives and then evaluating options to identify a solution that best solves an explicit problem while satisfying as many objectives as possible." (Westphal, 2000)

"Abbreviated DSS, the term refers to an interactive computerized system that gathers and presents data from a wide range of sources, typically for business purposes. DSS applications are systems and subsystems that help people make decisions based on data that is culled from a wide range of sources.(...) DSS applications are not single information resources, such as a database or a program that graphically represents sales figures, but the combination of integrated resources working together." [www.webopedia.com]

"A properly designed DSS is an interface software-based system intended to help decision makers compile useful information from a combination of raw data, documents, personal knowledge, or business models to identify and solve problems and make decisions" [http://en.wikipedia.org/wiki/Decision_support_system]

"Decision support system is defined as an approach or methodology for supporting decision-making. It uses an interactive, flexible, and adaptable computer-based information system especially developed for supporting the solution for a specific

non-structured management system. It uses data, provides an easy user interface, and can incorporate the decision maker's own insights. In addition, a DSS usually uses models and is built (often by end users) by an interactive and iterative process (evolutionary prototyping process). It supports all phases of decision-making and may include a knowledge component." (Turban, 1995)

From the definitions, two important features that a DSS must supply are an interactive interface to the user and a graphical presentation of the results, to simplify the system's access by the human managers, usually people with less expertise in information systems but very experienced in analyzing results in the problem domain context.

Some literature establish five broad categories of DSS: model-driven, data-driven, communication-driven, document-driven and knowledge-driven (Power, 2007):

- Model-driven DSS use data and parameters provided by the user to stimulate a simulation model that generates results to assist decision makers in analyzing a situation. In general, this kind of systems doesn't need very large data bases.
- Data-driven DSS manipulate large data sets and generally provide tools tailored to specific tasks.
- Communication-driven DSS are designed to support team collaboration work on a shared task. These systems depend on communication technologies and can use video-conferencing, groupware software and so on.
- Document-driven DSS manipulate several unstructured information present in a huge number of document formats, and provide document retrieval and analysis. The diversity of document formats may include images, sound, hypertext documents, spreadsheets, text and scanned documents, video, etc.
- Knowledge-driven DSS are specialized problem-solving "expert" systems. Typically they have a great knowledge about a particular domain (stored as facts, rules and procedures), they understand the problems and supply a special skill to solve those problems. Many of these systems use AI techniques and algorithms.

Nowadays, the concept of web-based DSS is gaining position. The idea is to supply decision support information and tools to a manager using a "thin-client" web browser (Turban et al., 2001; Power, 2007). The decision maker doesn't need to have special software to analyze the data, since the use of web browsers evolved rapidly and the specification of HTML 2.0 facilitates the presentation of the results, generated by a DSS application hosted in a remote

computer server. The server DSS application can belong to any one of the former categories listed, changing only the presentation layer of the results that becomes accessible worldwide.

3.5.2 Architecture

The architectural components of a DSS can be defined as the database (or knowledge base), the model (decision context and user criteria) and the user interface. One different division can be made and identify components like **inputs** (data and characteristics to analyze), **user expertise** (inputs that require manual analysis by the user before the application runs), **outputs** (results generated by the application itself) and **decisions** (results selected by the DSS based on user criteria).

The definition of architecture for a decision support system to manage an ecosystem, entails to take into account the different needs and possible formulations of the decision problem. The diversity of views and opinions hinders any general classification of existing architectures and difficult the choice by the designer. Existing approaches can be distinguished by (Serment, 2007):

- Characteristics of the ecosystem in study and problem of interest for the decision makers: natural resources, aquaculture, fire prevention, ...
- Nature of the decision: purely environmental, political, economical, social, ...
- Objectives for the usage of the tool: anticipation, prediction, exploration, understanding, teaching...
- Objective of the decision: prediction, medium/long-term planning, supervision and monitoring, crisis management...
- The time scale of the studied phenomenon and the constraints of the decision-making: short, medium or long-term.
- The spatial scale of the studied phenomenon and the impact of the decisions: local, regional, global...
- The modelling paradigm used: individual-centred models, centralized models (distribution models), objects and agent-based models...

The complexity of temporal and spatial environmental dynamics, the amount of data collected and available and the knowledge about the ecosystem, multiply the management options and affects the development of the DSS. Environmental decision making is generally based on priority biological indicators, to check that the ecosystem is environmentally sustainable. The decision criteria also vary according to the decision makers, stakeholders and their interests.

The economic dimension and, more recently, the social aspects, also play a predominant role (Serment, 2007).

Ecosystem management is complicated by the close interlink between humans and the natural world. People are part of ecosystems and, as such, influence and are influenced by their structure and functioning. Ironically, environmental awareness frequently leads people to develop efforts to keep ecosystems “natural” using different sorts of “artificial” techniques. In any case, ecosystem management is a very complex issue that may benefit from decision support tools.

Environmental DSS (EDSS) must be intelligent systems improving the consistency, quality and reducing the time needed for environmental decision making. These systems help direct decision-makers (or decision making group) by facilitating the study of specific criteria for evaluating alternatives or justifying environmental decisions. The interaction between the various partners must be cooperative, friendly, fast and efficient. Some authors (Cortés et al., 2000) classify EDSS as a tool that combines different research areas, including Artificial Intelligence, Geographic Information Systems, Modelling and Simulation and Human Machine Interface.

3.5.3 Environmental Management

Over the last 20 years, the socio-political climate with regard to environmental plans and policy making processes have changed dramatically. Environmental legislation has changed worldwide, from a purely reactive system that attribute responsibility for damages to a pro-active concept of preventing damages through appropriate planning and mitigation (Fedra, 2005).

Participatory decision making processes influence EDSS design, which must include the basic principles of regulatory and legal framework for environmental management (Fedra, 2005):

- Precise preparation: governmental authorities should identify and consider all relevant factors and circumstances when preparing decisions.
- Trust: citizens must trust governmental authorities in such a way that the expectations of citizens are not shamed.
- Fair play: partiality may be out of question and citizens may not be denied any possibility for protection of their own interests.
- Motivation: every decision of governmental authorities should be accomplished by a correct motivation and explanation.

During the problem definition phase and the generation of alternatives, EDSS's must identify the major actors and stakeholders, including the profile of stakeholder institutions, in order to embed the key problem issues in the generated scenarios of development. Also the relevant criteria that will influence decision making should be identified. In a second phase the preference structures that lead the ranking and the selection of alternatives should be defined. The multi-criteria analysis required in the final step enforces a political exercise of negotiation and trade-off with subjective values and beliefs, involving the end users to reach practical results.

3.5.4 Multi-Criteria Analysis

The Analytic Hierarchy Process (AHP), introduced by Thomas Saaty (Saaty, 1980), provides a powerful and flexible tool that may be used to make decisions in situations where multiple and conflicting objectives/criteria are present, and both qualitative and quantitative aspects of a decision need to be considered. The AHP is effective in dealing with complex decision making, and may help the decision maker to set priorities and make the best decision, even if it is not a specialist in the scientific domain. By reducing complex decisions to a series of pair-wise comparisons, and then synthesizing the results, the AHP helps to capture both subjective and objective aspects of a decision. In addition, the AHP incorporates a useful technique for checking the consistency of the decision maker's evaluations, thus reducing the bias in the decision making process.

The AHP generates a weight for each evaluation criterion according to the information provided by the user (decision maker). Higher weights mean more important objectives. Next, the AHP evaluates all the scenarios on each criterion using the information provided by the user. Finally, the AHP combines the objective weights and the scenarios evaluations, thus determining a global score for each scenario and a subsequent ranking. The global score for a given scenario is a weighted sum of the scores it obtained on the single criteria.

The AHP considers a set of evaluation criteria (objectives), and a set of alternative options (scenarios) among which the best decision is to be made. The several alternative options should be characterized by indicators. Indicators come from different sources and are measured or, in the case of scenarios simulated by models, calculated from the simulation results. It is important to note that, since some of the objectives could be contrasting (e.g. socio-economic interests and environment preservation), it is not true in general that the best option is the one which maximizes each single criterion, rather the one which achieves the most suitable trade-off among the different criteria (Siena, 2005).

When the AHP tool is embedded in the DSS software, the user only concentrates its attention in how to compare the criteria. The relative importance between two criteria translates the evaluation (qualitative or quantitative) made by the user to relate the criteria pair and is measured according to a numerical scale (the larger value the more important is the first criterion compared to the second). The relation between the second and the first criterion is the inverse quantity. Each criterion is compared against all the others criteria resulting the pair-wise comparison in a square matrix – the pair-wise comparison matrix (PCM). With this process, the users, the stakeholders and the site actors, are involved in the decision process and committed to the alternative proposed as the best scenario for a sustainable solution. The AHP methodology offers the data for plausible explanation of the best decision.

To work properly the AHP needs two data sets:

- The inputs or indicators to the multicriteria analysis – a $n \times m$ matrix of indicators, with m decision criteria considered and n simulated scenarios. Each entry V_{jk} in the matrix is the value obtained in the simulation for the k criterion in the j scenario.
- The PCM – a $m \times m$ square matrix A , where each entry a_{jk} is the relative importance of criterion j over criterion k ; the entry a_{kj} is the inverse value. If $a_{jk} > 1$, then the criterion j is more important than the criterion k , while if $a_{jk} < 1$, then the criterion j is less important than the criterion k . If the two criteria have the same importance than $a_{jk} = 1$.

The multicriteria analysis is necessary when the optimum value on each column of the first matrix is not achieved in the same scenario for all criteria, which happens almost always.

The AHP is implemented in three consecutive steps, assuming m evaluation criteria considered and n scenarios evaluated:

- (i) Computing the vector of objective weights
- (ii) Computing the matrix of scenario scores
- (iii) Ranking the scenarios.

Computing the vector of objective weights

In order to compute the weights for the different objectives, the AHP starts by inspecting the PCM matrix. The relative importance between two criteria is measured according to a numerical scale (e.g. 1 to 9), as shown in Table 3-1, where it is assumed that the criterion j is equally or more important than the criterion k .

Table 3-1: Table of relative scores for PCM matrix

| Value of A_{jk} | Interpretation |
|-------------------|--|
| 1 | j and k are equally important |
| 3 | j is slightly more important than k |
| 5 | j is strongly more important than k |
| 7 | j is very strongly more important than k |
| 9 | j is absolutely more important than k |

The interpretations of the values are suggestive (Siena, 2005), and may translate the user's qualitative evaluations of the relative importance between two criteria into a numerical scale. The values in the matrix \mathbf{A} are pair-wise consistent, that is, the PCM elements verify the constraint:

$$a_{jk} * a_{kj} = 1 \text{ and } a_{jj} = 1, \text{ for all } j \quad (9)$$

The following step is to build the matrix \mathbf{A}_{norm} , the normalization of the matrix \mathbf{A} , making equal to 1 the sum of the entries in each column. Each entry in the matrix \mathbf{A}_{norm} is:

$$\overline{a}_{jk} = \frac{a_{jk}}{\sum_{l=1}^m a_{lk}} \quad (10)$$

The vector of objective weights \mathbf{w} (a m -dimensional column vector) is the average of the entries on each row of the normalized matrix:

$$w_j = \frac{\sum_{i=1}^n \overline{a}_{ji}}{m} \quad (11)$$

Computing the matrix of the scenario scores

The matrix of scenario scores is a $n \times m$ real matrix \mathbf{S} , where each entry s_{ik} represents the score of the scenario i with respect to the criterion k . From the matrix with the values of the indicators for the scenarios evaluated, a pair-wise comparison matrix $\mathbf{B}^{(k)}$ is built for each criterion k . Each matrix $\mathbf{B}^{(k)}$ is a $n \times n$ real matrix, where n is the number of scenarios evaluated. Each entry $b_{ih}^{(k)}$ represents the evaluation of the scenario i compared to the scenario h with respect to the criterion k : if $b_{ih}^{(k)} > 1$ then the scenario i is better than scenario h , if $b_{ih}^{(k)} < 1$ then the scenario h is better than scenario i . If the two scenarios are identical $b_{ih}^{(k)} = 1$, and the $\mathbf{B}^{(k)}$ entries verify the PCM constraint:

$$b_{ih}^{(k)} * b_{hi}^{(k)} = 1 \text{ and } b_{ii}^{(k)} = 1, \text{ for all } i \quad (12)$$

The AHP applies to each matrix $B^{(k)}$ the same two-step procedure described for the pair-wise comparison matrix A : divides each entry by the sum of the entries in the same column, and then averages the entries on each row, obtaining the score vectors $s^{(k)}$, each one containing the scores of the evaluated scenarios with respect to criterion k .

The matrix of the scenario scores S is obtained as:

$$S = [s^{(1)} \quad \dots \quad s^{(m)}] \quad (13)$$

Where the k -th column of S corresponds to $s^{(k)}$.

Ranking the scenarios

Once the weight vector w and the score matrix S have been computed, the vector v of global scores is obtained by multiplying S and w :

$$v = S * w \quad (14)$$

The i -th entry v_i of v represents the global score assigned by the AHP to the scenario i . As the final step, the scenario ranking is accomplished by ordering the global scores in decreasing order.

In practice, this approach implies a limitation in terms of the number of criteria and scenarios that can be analyzed efficiently, as the effort to elicit a preference structure grows exponentially with the dimensionality of the problem. When the alternative criteria does not exceed the number of nine the AHP methodology fits perfectly the help to decision making.

3.6 Summary

In this chapter, Artificial Intelligence concepts used in the context of ecological modelling and related applications were described, including the use of intelligent agents, optimization algorithms and Machine Learning techniques. The use of advanced concepts of Distributed Artificial Intelligence and the use of multi-agent systems were also explained.

This chapter ended with the presentation of some concepts in environmental decision support systems and their integration in the management of ecosystems. A strategy to solve the multiple criteria and conflicting objectives that involves the management of ecological systems was also presented – the AHP methodology.

A simulation system for coastal ecosystems only makes sense if it provides results and indicators relevant to the management of the ecosystem in question. Many of the tools described in this chapter can be adapted and integrated with the modelling tools, presented in the previous chapter, to simulate and manage natural ecosystems, but there is a lack of solutions with the integration done.

4 Methodology and Implementation

“Our sense of being a conscious agent who does things comes at a cost of being technically wrong all the time.”

Daniel Wegner (Wegner, 2003: 342)

4.1 Introduction

Ecosystem management is a difficult task for the local or national authorities. The objectives of environment preservation and socio-economic interests are typically contrasting and a trade-off between them must be achieved. The solution to this problem requires the development of interdisciplinary and multi-criteria approaches.

This work implements an Intelligent Simulation System that includes a mathematical model for simulating the processes of coastal ecosystems, an user-friendly graphical representation of the ecosystem (exportable to GIS), one farmer agent representing the aquaculture stakeholders and an environmental decision support system (EDSS) able to integrate more intelligent agents representing stakeholders, managers and different human interests over the ecosystem.

The complete simulation system was designed and developed to be applicable to any coastal ecosystem, starting with Sungo Bay (People’s Republic of China) and Ria Formosa (south of Portugal) models as prototypes or proof of concept. Every developed application (simulator, visualizer and agents) has communication skills, enabling the interchange of valuable

information between them, and providing a tool to help and support ecosystem management decisions in real scenarios simulations (Pereira et al., 2004b). Each application is able to “decide” what is the public information that may be exchanged with the others.

The core of the system is the EcoDynamo simulator (Pereira and Duarte, 2005), designed with portable Dynamic Link Libraries that simulate the ecological processes. It is a user-friendly software application focused in aquatic systems but adaptable to any kind of ecological simulations.

A specific language – ECOLANG (Pereira et al., 2005; Pereira, 2008) – was developed for the communications between all the applications that compose the system. The network established by the applications belonging to the Intelligent Simulation System, based in the simulator EcoDynamo and communicating using ECOLANG messages, was named EcoSimNet.

The agents constructed to be integrated in the simulation system, representing the interests of the stakeholders, local communities and authorities, share and implement Machine Learning methodologies and optimization techniques.

A small decision support procedure, with a simple and proved methodology - Analytic Hierarchy Process (Saaty, 1980) - was included in the system, integrating the contrasting and opposite interests of the several agents (Pereira et al., 2007).

4.2 Generic Network Infrastructure

The fundamental modules that a system must contain to help decision makers are: modelling and simulation (allowing the simulation of different management scenarios), data management and analysis (allowing the classification of simulated scenarios), visualization and integration with users. One advantageous approach is to design independent modules that interact with each other in a proper way. The maintenance of independence between the modules makes the system more scalable, flexible and robust, allowing the reutilization of previous developed modules in new solutions.

The **modelling module** starts with the definition of the problem and formulates the processes to study and the interactions between them. The **simulation module** enables the simulation of different scenarios using the modelling module and generates the results that can be saved for posterior analysis by the user or intelligent agent. The **visualization module** shows graphically the behaviour of the models, improving the interpretation of the results by the decision makers or users. The **analysis module** enables the comparison of the results generated by the

different simulated scenarios. The module responsible for the **integration with users** ensures the users access to the functionalities of the applications in the network, in an uniform user-friendly way. The **data management module** should ensure consistency and persistence of configuration data and generated results in different sources and formats. Although these modules are identified, their functionalities can be distributed by several applications and can be transversal to some or all of them.

Simulation with hydrodynamic and biogeochemical models is being used, with increasing frequency, to explain and/or confirm theories about ecosystems' behaviour, and to predict future developments, trends and scenarios of exploitation of the ecosystem under study. However, human activities and actions over the ecosystems are difficult to incorporate.

The use of autonomous agents (Weiss, 1999; Wooldridge, 2002), representing the institutional authorities and the local stakeholders, seems to be a promising way to implement the influence of humans over the simulated systems, and their learning about it. Agents interact with the simulator, generating scenarios for the model, capturing the results of the created scenarios, integrating environmental and socio-economic information and then evaluating the results (Pereira et al., 2004b, 2007).

This work developed a generic and complete multi-agent simulation system that includes an ecological simulator, a visualization tool and a farmer agent, and is opened to the inclusion of more autonomous agents representing intelligent entities (stakeholders and decision makers). Several machine learning techniques have been integrated in the agents providing them with artificial intelligence capabilities.

The interaction between all the components is made via ECOLANG (Pereira et al., 2005), a high level communication language, completely readable and understandable by the humans and by the applications, making it simple to trace the interactions between the different applications.

Figure 4-1 displays the proposal for the generic network infrastructure, or framework, which facilitates the integration of new modules, their manipulation and interaction. The Network Integration Infrastructure is composed by a physical layer (network connections between computers holding the different applications) and by a logical layer (ECOLANG messages exchanged and associated protocol).

This infrastructure offers to the final users the chance to choose the modules that better interpret their views for specific functionalities, integrating those views, and building a system that functions as a whole.

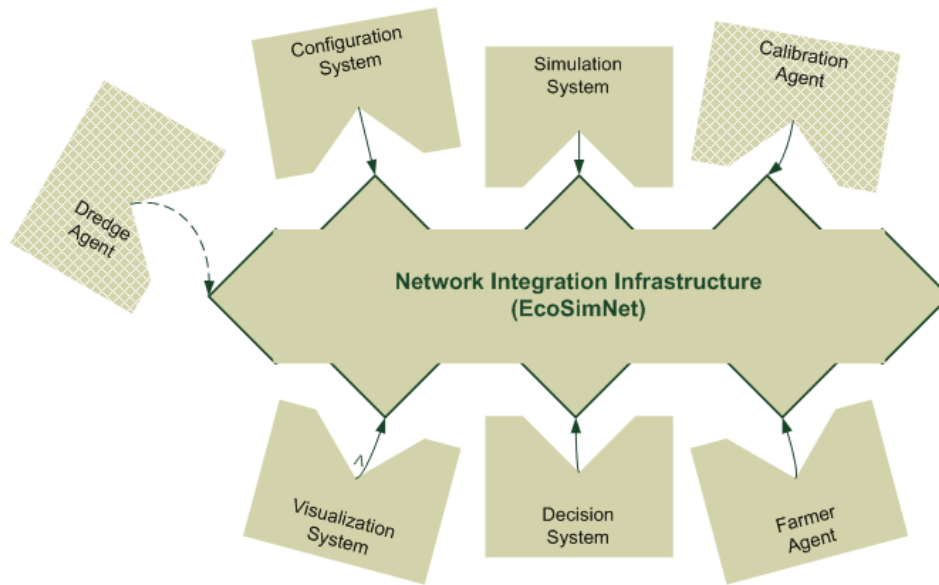


Figure 4-1: Generic network integration infrastructure

The framework was developed and applied in ecological models of coastal ecosystems, with emphasis to Sungo Bay (People's Republic of China) and Ria Formosa lagoon in Algarve, and can be used for aquaculture optimization, system carrying capacity calculation and other ecological management intents.

4.3 Simulation System

The simulation application used and enhanced in this work is an essential part of the system and is particularly focused in modelling of coastal ecosystems. These ecosystems are subject to strong anthropogenic pressures due to tourism and shellfish farming or fish aquaculture, factors that are responsible for important ecosystem changes characterized by eutrophic conditions, algal blooms, oxygen depletion and hydrogen sulphide production.

This section will describe the models used in the simulation system, and the simulator architecture specially developed for this application – the main/core application and the dynamic libraries that contain the simulation classes.

4.3.1 Model description

Models of aquatic ecosystems must include biogeochemical processes, such as photosynthesis, nutrient cycling and grazing, and transport processes. The biogeochemical processes are responsible for local changes of state variables, such as the concentration of chemical constituents and biomass of different species or groups of species. As the name indicates, the transport processes, or hydrodynamics, are responsible for the transport of pelagic variables across model domain and across model boundaries.

Increasingly sophisticated numerical models have been developed to promote reliable, real-time management of coastal ecosystems. Each model pretends to account realistically for the processes that drive the dynamic behaviour in coastal lagoons so that their status may be predicted and the effects of mitigation actions be properly evaluated, resulting in a series of good management practices that increase the sustainability of these fragile ecosystems (Chapelle et al., 2005b).

Different approaches are followed to simulate each set of processes. The mathematical equations used to model the biogeochemical processes are based on simplifications, implicit or ambiguous assumptions and, in consequence, there is a lack of universally accepted theories. In contrast, the hydrodynamic processes are very well established, based on simplified forms of the Navier Stokes' equations, known as the Boussinesq equations (cf. 2.4.2).

The model implemented in this work follows the object-oriented modelling approach - different objects are used to simulate hydrodynamic, thermodynamic and biogeochemical processes and variables. Each object is implemented within independent classes that contain its own way to simulate the object processes. If the mathematical equations have parameters that can be adjusted to each site, they are fixed during the calibration, verification and validation phases of the model.

The object-oriented approach allow the researchers to separate all the independent functional groups, modelling them individually, and establishing communication among them to exchange results and important data for their behaviours. Processes that extend the behaviour of already developed ones do not need to be newly created, inheriting all the equations already developed and just adding what distinguishes it from the parent.

Hydrodynamic Modelling

Hydrodynamic models solve advection and diffusion phenomena of momentum, buoyancy and turbulent kinetic energy (Chapelle et al., 2005b), and must setup the time and space scales that will be used, considering the dimensions in the physical space.

Physical dynamics in coastal ecosystems is forced by tidal height at the sea boundaries derived from the flood-ebb tides - very relevant in the case of Ria Formosa due to its large shallow intertidal areas - river discharges, wind drag forces, land drainage and waste water treatment plants at land boundaries.

The hydrodynamic process simulates from 1D to 3D solutions. For the case studies analysed in the present work, it is simulated as a two-dimensional simplified solution of the Navier-Stocks equation, adapted from Neves (1985). It is assumed that vertical mixing is strong enough to prevent vertical gradients in water properties – a frequent assumption in shallow water ecosystems. This 2D formulation is based on a finite difference staggered grid (Vreugdenhil, 1989). Flows are solved at the sides of the grid cells, whereas surface elevations and concentrations are calculated at the centre of the cells. Advection terms (cf. 2.3.2) are calculated using an upwind scheme, whereas diffusion terms are based on a central differences scheme. The resolution is semi-implicit. At the first semi-time step the u (west-east) component is solved implicitly and the v (north-south) component solved explicitly. At the second semi-time step it is the other way around. After the calculation of both velocity components, surface elevation is calculated by continuity, as well as the concentration of conservative and non-conservative substances, after solving the transport equation (Knauss, 1997) for all pelagic state variables:

$$\frac{dS}{dt} + \frac{\partial(uS)}{\partial x} + \frac{\partial(vS)}{\partial y} = A_x \frac{\partial^2 S}{\partial x^2} + A_y \frac{\partial^2 S}{\partial y^2} + \text{Sources} - \text{Sinks} \quad (15)$$

Where:

u and v are the current speeds in x (west-east) and y (north-south) (m s^{-1});

A is the coefficient of eddy diffusivity ($\text{m}^2 \text{s}^{-1}$);

S is a conservative or a non conservative variable in the respective concentration units;

Sources and *Sinks* are values calculated by the biogeochemical processes at each grid cell of the model.

The model includes a wet-drying scheme that prevents any grid cell from running completely dry, avoiding numerical errors. The general approach is to stop using the advection term when

water level drops below a threshold value (configured in the model database) to avoid numerical instabilities. When this limit is reached, computations do not take place in a given cell until a neighbour cell has a higher water level, allowing the water to be driven into the “dry” cell by the pressure gradient force (Chapelle et al., 2005b).

Detailed information about the system equations can be found in Duarte et al. (2003) for the Sungu Bay model and in Duarte et al. (2005a) for the Ria Formosa site.

Biogeochemical Modelling

The term “biogeochemical modelling” refers to the mathematical modelling of ecosystems that include biochemistry (nutrient cycles, respiration and feeding, mortality and recruitment, predation and competition). Biogeochemical models consist of compartments represented by state variables and described by differential equations, governing the transfer of material between them (Taylor, 1993). The choice of the state variables is guided by the addressed problem that the model was created for. Different modellers can view distinct variables and processes for the same problem, derived by the self subjectivity of the modeller and the priority each one assign to the modelled processes.

Biogeochemical cycles are driven by radiant energy and the Energy Circuit Language (“Energese” or Energy Systems Language) designed by Odum (1973, 1983) is frequently used to represent the energy flows within ecological and biological systems.

The biogeochemical model integrated in this work follows the guidelines pointed out in section 2.4.3 . The biogeochemical model implemented for both sites in study is a “coupled physical-biogeochemical” and specific classes were developed for each location derived from the benthic species that are characteristic from each site.

Water column biogeochemistry is simulated for nitrogen, phosphorous and oxygen. Processes such as mineralization of organic matter, nitrification and denitrification were considered for nitrogen. Particulate organic matter (POM) is mineralized to ammonium nitrogen, and oxygen is consumed in mineralization and nitrification and exchanged across the air–water interface. Total and organic particulate matter concentrations (TPM and POM) are simulated following equations described in Duarte et al. (2003). More details on the ecological model and a complete listing of equations and parameters can be found in Duarte et al. (2005a) and Chapelle et al. (2005a; 2005b). For macroalgae, the work of Serpa (2004) was used and for the sea grass *Zostera noltii*, the work of Plus (2003) was followed – vide Duarte et al. (2008).

Details about the differential equations used by the physical and biogeochemical processes modelled, the list of the parameters and their values, can be found in Duarte et al. (2003; 2008) and Chapelle et al. (2005b).

4.3.2 EcoDynamo

The simulator EcoDynamo - **E**cological **D**ynamics **M**odel (Pereira and Duarte, 2005) - is the central element of the general architecture proposed for the intelligent simulation system. It is a software application to simulate physical, biological, geochemical and anthropogenic processes in aquatic ecosystems, partially developed under the European project DITTY [<http://www.dittyproject.org>] (EC, 2003) and, in this work, was further enhanced to include communications, commands and multi-thread modules:

- The **multi-thread module** provides multi-threaded execution of the simulation, distributing by several threads the load of the different classes simulated and, when in the presence of a computer with multiprocessing capabilities, scheduling the charge of the system by the existing CPUs;
- The **communications module** provides the simulator with components that support network communications through TCP/IP sockets to interact with agents and other applications;
- The **commands module** processes the agents' actions and responds to them with the corresponding perceptions translated in ECOLANG messages (implementation of EcoDynamo protocol).

EcoDynamo was initially inspired in EcoWin (Ferreira, 1995) and they are still similar in several aspects – the object-oriented approach, the way how objects communicate with each other and the simulation step sequence. However they have many distinct features. The most important difference is that EcoDynamo was designed for coupled hydrodynamic-biogeochemical models, while EcoWin was designed for box models. The relatively small number of boxes of any type and shape, in typical EcoWin applications, enables each box to have many connections to a number of other boxes and these connections must be defined by the user one by one. In EcoDynamo, the type of grids used encapsulates the possible connections between cells, defined by the matrix grid structure, without the need for the user to define cell connections and allowing the use of very large grids (Pereira et al., 2006).

EcoDynamo application is a C++ object-oriented software and its internal structure is divided in seven modules (depicted in Figure 4-2). Each module exchanges data and information with others and is responsible for a delimited functionality.

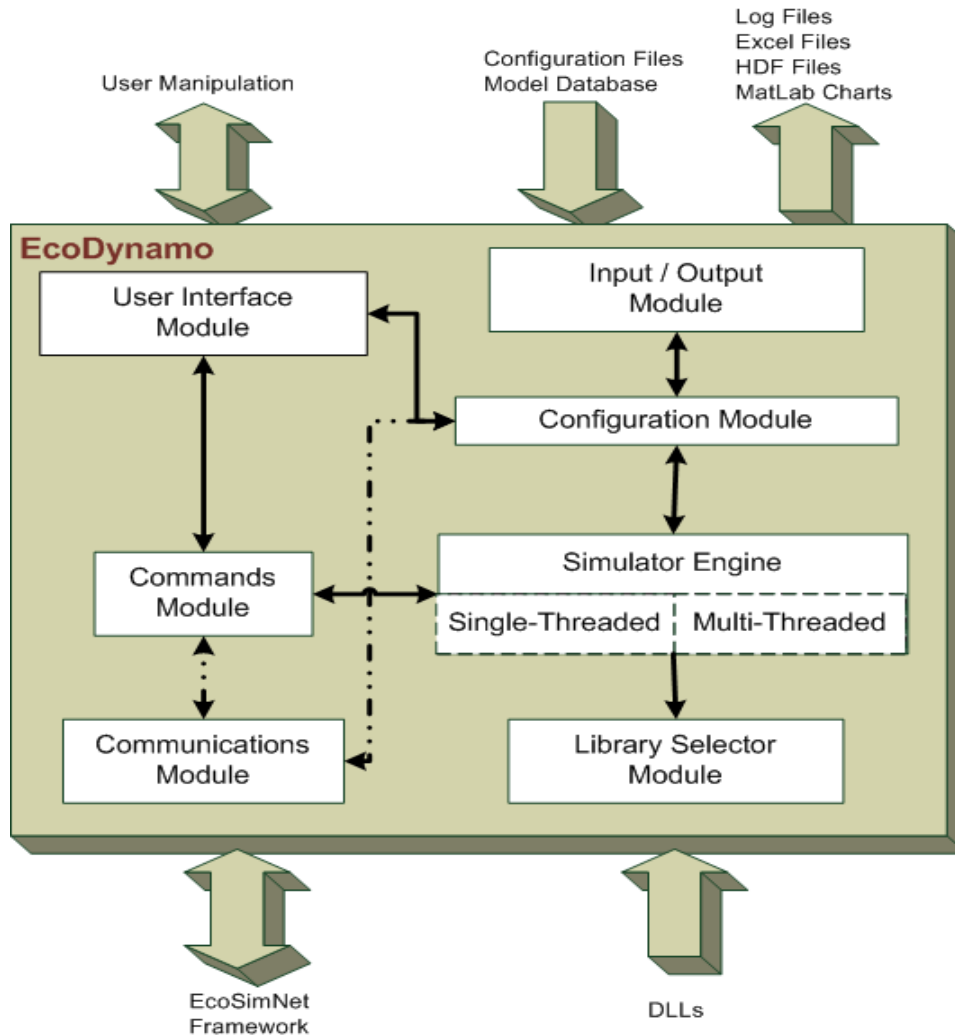


Figure 4-2: EcoDynamo internal structure

User Interface Module

As its name indicates, this module is responsible for the interface with the user – select the model to simulate, display all the information about the model configuration, select the classes/objects to simulate, the simulation time step, the variables to output, the output format, the type of support and the output frequency, the trace over the communications, as well as start and stop the simulations. Simulation activity can also be monitored in runtime, with options to see the messages exchanged by the objects and threads' activities. This analysis can be done offline with the help of log files, activated by the user for specified steps, before the simulation run.

The main window is divided in two panels – the simulation panel that controls the simulations with very intuitive buttons, and the output panel that controls where the variables selected for output will be recorded (Figure 4-3).

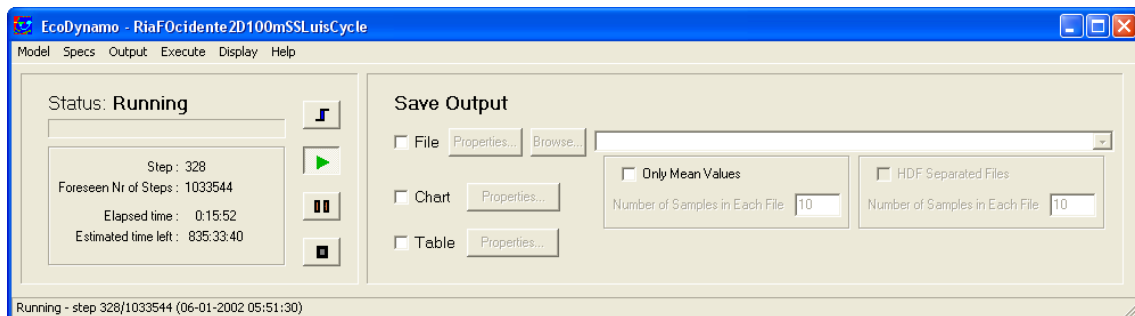


Figure 4-3: EcoDynamo main window

A detailed description of the user's manual can be found in Annex 1, where all the possible actions made by the user are presented.

Input / Output Module

The input/output module is responsible for reading the configuration files of the model chosen, for persistently record the results generated by the simulations, and for logging any trace activated before the simulation starts.

The model database is distributed by several text files formatted with tab-separated or comma-separated values, that can be accessed and modified by any text editor or commercial application (Notepad, Wordpad, Excel, Calc, etc.). The mandatory files are:

- Morphology file - describes the morphology of the ecosystem, including the model type, the number of cells (lines, columns and layers), geometric representation and boundary types;
- Classes file – lists all the classes that can be used in the simulation. For each class there are two files:
 - Variables file – lists the available variables and initial values;
 - Parameters file – lists the inner parameters and values.

Besides these files, there are other important but not mandatory:

- Benthic species file – describes the benthic species existent in the ecosystem and the places where they are located;
- Loads file – describes the locations and the amounts of any kind of loads into the ecosystem, such as Waste Water Treatment Plants discharges;

- River loads file – indicates the river locations and daily river flows to the ecosystem;
- Losses file – describes the losses locations (important for river models);
- Sediments file – defines sediment types and locations within the model grid;
- Points file – define sub-domains within the ecosystem.

The results can be saved in files or plotted in graphics. The General Output Files have, by default, three formats: **xls**, **txt** or **hdf**. The “xls” and “txt” formats are text files saved with tab-separated values (the “xls” extension is used for Excel or Calc applications quick view). The first seven columns are fixed for all registers: time in UTC (number of seconds from January 1, 1970 00:00), time (in day hours), model time step, column (west-east grid coordinate), line (north-south grid coordinate), layer (bottom-up grid coordinate) and cell numbers. The “hdf” format stands for High Density Format and handles large volumes of data, following the HDF specifications - see [<http://hdf.ncsa.uiuc.edu/>] in University of Illinois at Urbana-Champaign.

The output part of the module has two interface layers to the MatLab® application – one accommodates the HDF format for files, and the other provides a graphical output easily customizable by the user.

There is one option to save Mean Values Files. This option enables the user to run the model with a minor time step (normally only the hydrodynamic part of the model) and save the mean flow and velocity values in files, integrated over a time period under the choice of the user. The mean values files are formatted as “xls” files and their names start with “HydroTimeSeriesValues” and follow a sequential order. The first four columns are: date and time (DD-MM-YYYY hh:mm:ss format), time (UTC), register step and cell number. These first columns are followed by columns with the time-averaged results of the variables “Mean U Flow”, “Mean U Velocity”, “Mean V Flow” and “Mean V Velocity”.

The log files can be saved in text, excel or XML formats. These files contain the messages exchanged by the objects during specified steps of the simulation.

Extended detailed information about the format of each file is described in the EcoDynamo User’s Manual (Annex 1).

Configuration Module

This module guarantees that the configuration read from the model database is coherent and valid before the start of any simulation. It is the input interface to the simulator engine module and implements, among others, an important feature of the EcoDynamo – the sub-domain

concept. The sub-domain concept permits to run the simulation only in a subset of the model domain without loss of fidelity in the simulated processes. It is an important improvement and it can reduce the simulation runtime, discarding the irrelevant cells of the model domain and concentrating the computational resources only in the area of study. These sub-domains may be defined in various ways, e.g. with a Geographical Information System (GIS) application and transferred to one points file.

The configuration module also controls an option that enables the interruption of a simulation and the storage of a snapshot of the system's state when the simulation is interrupted (time step, database configuration and the variables' values). The simulation can be resumed later from the time step where it was interrupted, e.g. when the load on the computer's CPU is lower.

Commands Module

The commands module is responsible for controlling the simulator engine (run, pause, step and stop simulations) and it acts as a filter to commands received from users or from external applications that belong to the simulation network. If simultaneous commands are received from different sources, the responsibility of this module is to put them in a coherent way, sequencing them or simply discard some.

Another important role in this module is the processing of the agents' actions, into commands to the simulator, and the conversion of the simulator results to the corresponding perceptions translated into ECOLANG messages and sent to the agents (implementation of EcoDynamo protocol explained in section 4.4.1).

Simulator Engine Module

This module is responsible for controlling the simulation and send the results generated to the input/output module. It can be considered the core of EcoDynamo due its responsibilities in maintaining updated information about the simulation processes in all other modules. The simulation is performed as a cyclic loop that embraces three phases, involving all the active objects (selected by the configuration module):

- Phase 1 – hydrodynamic object calculates velocity fields and water elevations; other objects calculate local changes of their variables at each grid cell.
- Phase 2 – integrate the local partial changes – the hydrodynamic object transports all “transportable” variables across the grid cells.

- Phase 3 – update velocities, flows and cells geometry to the next time step.

It is divided in two sub-modules that execute the simulation: the single-threaded and the multi-threaded modules. In the single-threaded version all the objects are simulated by the order of selection. In the multi-threaded version, different objects are controlled by different threads, taking advantage from the independence of the objects and the existence of multiple processors in the computer – distributing the charge of the system by the existing CPUs. Despite the independence of the objects, the inter-relationships between some of them (like the need for a variable value from other object) force the synchronization within each time step, in order to guarantee system stability.

The communication between the different simulated objects, representing different variables and processes, is facilitated by the core shell of the simulator but is made directly by individual players. This allows the construction of a logbook, activated before each simulation run, with the interactions between different objects, which can be an important database in the machine-learning process used by the agents.

Library Selector Module

The library selector module is responsible for the selection and activation of the classes involved in the simulation. As different processes are simulated by different classes, the simulator engine reads the model configuration and selects the desired classes. This module imports the code of each class from the library and activates only the appropriate objects. When the simulation ends the imported code is freed.

It is the interface module between the objects contained in the library with dynamic code and the simulator engine, promoting the independence between the simulator and the classes/objects simulated, as well as code reusability.

Communications Module

The communications module is responsible for the interface with the environment (send/receive messages and control the network interface) enabling EcoDynamo to interact with other applications through TCP/IP sockets.

It implements the EcoDynamo protocol, exchanging information with other applications via messages formatted according to the ECOLANG specification (cf. 4.4.1). The actions received from external sources are converted into internal directives passed to the commands module.

Any external configuration command received is forwarded to the configuration module, depending on the current state of the simulation. The information generated internally by EcoDynamo (spontaneously or as responses to messages received) is forwarded through this module and sent to the outside, converted into ECOLANG messages as perceptions to the agents.

While the other modules are more or less interdependent, this module is completely independent and may be inactive during the simulation process, for example, when the simulation is done in standalone mode, without connections with other applications.

4.3.3 Library of Dynamic Linkable Objects

Considering the objectives presented at section 1.2, and aiming at developing a flexible and realistic simulator, the development of EcoDynamo followed several general requirements initially listed:

- The core engine of the simulator should be as independent as possible of external conditions (hardware or software) and should allow easy integration of new input/output modules or new objects to simulate;
- Different processes must be simulated by different classes with different variables, proper parameters and process equations;
- Classes must be self-contained and independent;
- All public behaviours must be defined by a base class or interface, from which all process classes inherit. This enables a uniform interface invocation from the simulator engine and external applications, taking advantage from the polymorphism property of the programming language;
- Base class must provide transparent mechanisms for data exchange between different objects;
- Objects must be, easy and dynamically, included or excluded from each simulation;
- Objects must be developed in order to be used outside EcoDynamo environment, as external libraries, enabling researchers from other teams, with different tools, to embed EcoDynamo objects in their systems.

One of the reasons to choose the C++ language was the possibility of taking advantage from the dynamic link libraries (DLLs) – pieces of code (classes) that are not part of the application core code, are compiled individually and linked in packages, and that can be integrated in the application during runtime when they are needed. Hence the name dynamic link libraries.

This concept allows the aggregation of code according its functionality and the creation of software packages that are independent, self-contained, and could be used like “plug and play” hardware. All of this code is the library of dynamic linkable objects and each package is referred as one DLL.

Generic DLLs

All packages that contain generic code (not designed as simulation objects) are included in the self-called generic dynamic link libraries, and are listed in Table 4-1.

Table 4-1: Generic DLLs included in EcoDynamo

| Component/DLL name | Classes and functionalities |
|---------------------------|--|
| Io.dll | ReadWrite – manipulates <i>txt</i> and <i>xls</i> files Log – manipulates log files in <i>xls</i> and <i>XML</i> formats Properties – manipulates <i>Properties</i> files |
| MatLabChart.dll | MLChart – manages chart configuration MLChartFigure – saves chart variable MLCharBox – saves chart data point |
| MatLabHDF.dll | MLHDF – manipulates <i>HDF</i> files |
| Utilities.dll | Queue – implements soft queue BUF – Queue element Parser – parsing facilities |
| ECDP.dll | EcoDynProtocol –manages EcoDynamo protocol |

The Input/Output Module handles several types of input/output and, at present, there are DLLs to deal with text files, log files, MatLab® charts and HDF files.

Each protocol supported by the Communications Module must have one DLL with the syntax and rules of that protocol. At present, only the EcoDynamo protocol is available, allowing the connection to the EcoSimNet platform with ECOLANG messages.

Simulation Objects DLLs

The Library Selector Module is not only responsible for selecting the generic DLLs (for input, output and communications) but, also, for selecting the DLLs that contain the classes to simulate the desired ecological objects.

EcoDynClass is the base class that implements the basic simulation object in the library of dynamic linkable objects, and all the remaining simulation objects inherit from it. EcoDynClass

controls the model time step evolution and maintains a list of all the active objects in the simulation and their relationships (Pereira et al., 2006). It reads the model morphology, initialises relevant fields and defines the default behaviour of the public methods, common to all objects:

- **Go** – responsible for object processes’ calculations;
- **Integrate** – responsible for time integrations within each grid cell;
- **Reinitialize** – responsible to update velocities, flows and system geometry to the next time step;
- **Update** – update one internal variable value, requested by an external object;
- **Inquiry** – send to an external object the value of one internal variable.

The simulation runs as a cyclic loop, and the simulator engine follows partially the EcoWin methodology (Ferreira, 1995) invoking, for all the objects, one method in each phase of the cycle:

- “Go” is invoked in phase 1,
- “Integrate” is invoked in phase 2, and
- “Reinitialize” is invoked in phase 3, although only hydrodynamic objects do something with it.

In Figure 4-4 all direct descendents from EcoDynClass are represented - the prefix T is used in the implementation of all template classes and is appended to the name of the class.

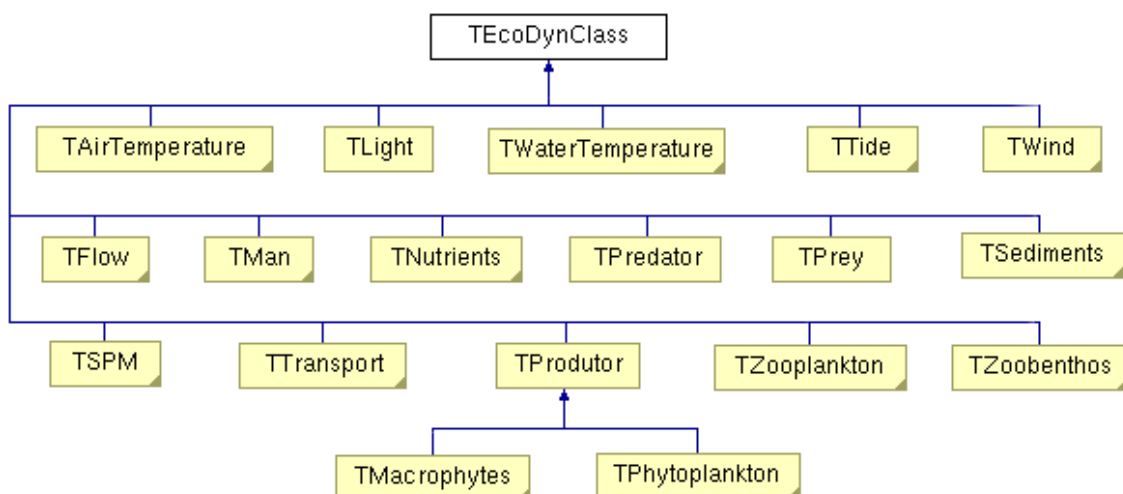


Figure 4-4: Inheritance diagram for EcoDynClass

The names of the descendant classes indicate the different processes that each one is responsible. Most of them serve as base classes for objects more suited to specific processes

of each ecosystem or species. The objects that compose the first line of Figure 4-4 represent the **forcing functions** present in all aquatic ecosystems, and Table 4-2 indicates the simulated forcing objects existent in the library of DLLs and the more relevant variables that each one calculates.

Table 4-2: Summary of EcoDynamo forcing classes and simulated variables

| DLL name | Class name | Class variables |
|----------------|------------------|---|
| Tair.dll | AirTemperature | Air temperature |
| Light.dll | Light | Total and photo synthetically active radiation (PAR) |
| Twaterobjt.dll | WaterTemperature | Radiative fluxes and balance between water and atmosphere and water temperature |
| Tides.dll | TideObject | Tidal height |
| Wind.dll | Wind | Wind speed |

The modelled **physical and biogeochemical processes** include:

- Hydrodynamics of aquatic systems: current speeds and directions;
- Thermodynamics: water and atmosphere temperature and energy balances between water and atmosphere;
- Biogeochemistry: nutrient biogeochemical cycles and biological species growth dynamics;
- Anthropogenic pressures, such as biomass harvesting.

Avoiding being too exhaustive, Table 4-3 summarizes the most important classes of biogeochemical processes and the simulated variables.

Table 4-3: Summary of EcoDynamo physical and biogeochemical classes and simulated variables

| DLL name | Class name | Class variables |
|-------------------------|--------------------------|--|
| Hydrobjt.dll | Hydrodynamics | Sea level, current speed and direction |
| Sediments.dll | Sediment biogeochemistry | Inorganic nitrogen, phosphorus and oxygen, dissolved in the interstitial water contained in the sediments, sediment adsorbed inorganic phosphorus, organic phosphorus, nitrogen and carbon |
| DissolvedSubstances.dll | Dissolved substances | Water column ammonia, nitrate and nitrite, dissolved inorganic nitrogen, inorganic phosphorus and oxygen |
| SuspendedMatter.dll | Suspended matter | Total particulate matter, particulate organic matter, carbon, nitrogen, phosphorous and water light extinction coefficient |
| Phytoplankton.dll | Phytoplankton | Phytoplankton biomass and production, chlorophyll concentrations and cell nutrient quotas |
| Zooplankton.dll | Zooplankton | Zooplankton biomass and production |

| | | |
|-----------------------|---|---|
| Macrophytes.dll | Kelp (<i>Laminaria japonica</i>), Macroalgae (<i>Enteromorpha sp.</i> and <i>Ulva sp.</i>) and Macrophyte (<i>Zostera noltii</i>) | Biomass and production, nutrient quotas and demographic fluxes |
| SuspensionFeeders.dll | Oyster (<i>Crassostrea gigas</i>), Scallops (<i>Chlamys farreri</i>) and Clams (<i>Ruditapes decussatus</i>) | Bivalves size, biomass, density, filtration, feeding, assimilation and scope for growth |

Detailed information about object's internal processes and state variables can be found in Duarte et al. (2003; 2008) and Chapelle et al. (2005b).

MAN OBJECT

This object simulates external actions for seeding and harvesting macroalgae and bivalves. The model may be initialised with or without any benthic species. The library has one Man object that reads, from the configuration files, fixed dates for seeding and harvesting, and has another object (Farmer) that translates external orders into those actions. Any of them, or both, can be selected in each simulation.

Annex 2 presents the hierarchy diagram for the most important classes implemented in the library of dynamic linkable objects.

Using DLLs objects outside EcoDynamo

One important feature included in the classes of the dynamic link library (DLLs) is that their code is ready to be linked with software written in other programming languages (e.g., Fortran or C).

Over the last decades, the community of researchers in the field of ecological modelling have built a huge number of modelling tools for the simulation of hydrodynamic and biogeochemical processes in aquatic systems, but different modelling research teams tend to adopt different modelling tools developed in different languages. This is an obstacle when teams want to exchange knowledge or reuse software modules. The approach followed in EcoDynamo enables other teams to share the code written for the simulated classes and objects, based on the object oriented programming potentiality (Stroustrup, 2000; Pereira et al., 2006). Each object represents different processes and variables and, as it is self-contained, can be used by programs written in different source code languages, following simple rules in the linkage phase.

During the project (EC, 2003), experiments were done with Coherens application (Luyten et al., 1999) (software built in Fortran code) in order to enable bidirectional code reutilization and results were reported in Pereira et al. (2006) indicating that the approach followed points in the right direction. In those experiments, the classes `Light` (`Light.dll`), `Phytoplankton` (`Phytoplankton.dll`) and `WaterTemperature` (`twaterobjt.dll`), from `EcoDynamo`, were successfully integrated with Coherens models.

The C++ DLL define and supply interface functions to be invoked by external applications to manipulate C++ objects (create, use and destroy objects, read their properties from permanent storage and call their methods).

When it is necessary to have an external application, e.g. written in Fortran, using and manipulating a C++ object, the main program, subroutines and functions in the Fortran code invoke the functions supplied by the interface. This is very challengeable because there is no pointer system in Fortran. The solution is to use the Fortran concept of “logical units”: the C++ interface will generate an integer reference number for all objects manipulated by Fortran. The Fortran code will have to keep a map associating those reference numbers to real objects. The solution proposed associates the address of the object in memory with the reference number in Fortran (the 32-bit integer in Fortran has the same size as a pointer in C or C++ when a 32-bit compiler is used). The compilation phase is straightforward – each source file is compiled with its native compiler. The linking process is a little bit more complex because to have objects instantiated when needed, and C++ special mechanisms activated, it is advisable to use the C++ linker, adding Fortran libraries as link options.

This is very system dependent and caution must be taken before rebuild the executable program in a new platform, but the time needed to make this adaptation is negligible compared to the time needed to implement the code of one new object. And here is one valid recipe for GNU compilers **gcc**, **g++** and **g77**, used with Coherens:

FORTRAN APPLICATION USING DLLS OBJECTS

Each DLL class must define a C-style singleton interface. The C-style interface provides a static method that returns the reference address of the object instantiated by the constructor when the first call to that object is performed by the Fortran code. Every time Fortran code wants to use methods from that object (or read/write data), the reference must be indicated by Fortran code or, in another way, must be supplied by the singleton interface method.

The singleton interface class must adhere to the following rules:

1. Definition of one public static method that returns the reference address of the class.
2. Definition of one block 'extern "C"' (C-style interface directive) with all the functions that can be called from Fortran:
 - a. The names of the functions must be lowercase with underscores appended;
 - b. All the parameters must be passed by reference.
3. Changes in the original source code (DLLs C++ sources) must be enclosed by the symbol `_PORT_FORTTRAN_` to enable conditional compilation in both projects (EcoDynamo DLLs and EcoDynamo/Coherens program).

The makefile that builds EcoDynamo/Coherens program (compile and link) must include the following rules:

1. The source directories of DLLs classes must be added to the compilation flags as include directories.
2. The symbol `_PORT_FORTTRAN_` must be defined in the compilation flags.
3. The Fortran libraries must be added to link command.

The Fortran code must follow the rules:

1. Definition of one global integer variable to save the reference address for each class.
2. During the initialization of the program, one first call to each DLL interface function must be made in order to create the class object.
3. Calls to interface functions always done with the reference variable.

For each class existent in the DLLs platform one singleton interface class must be defined. More information can be found in (Pereira et al., 2006) and in the MinGW project website (Minimalist GNU for Windows [<http://www.mingw.org>]).

4.4 Multi-Agent Simulation System

In the previous section, the modules for modelling and simulation were described. The simulation application has a simple interface with the users to facilitate the choice of the model to simulate, the configuration of the simulations, and the selection of the output variables and formats. The processes from the modelling module are used by the simulation module, in a completely transparent mode to the user.

This section will address the core of the network infrastructure – the communication language used by all the applications in the network and the rules to integrate each new application - the visualization system and agents already developed and included in the framework.

The complexity exhibited by coastal lagoons, hard to model and to simulate, is enlarged if the simulation model intends to include the intelligent entities that intervene in the system. In real ecological systems, man is heavily involved and his decisions are not based on fixed mathematical equations or simple rules but, contrarily, they are based on an in-depth analysis of the environment using his knowledge and experience.

Intelligent entities are modelled as agents that have perception of their environment, reason using their knowledge and are able to change the simulated scenarios by using a given set of configurable actions. Intelligent entities also communicate with each other – this functionality is also included in the agent model providing it with capacity to communicate with other agents or applications.

The general model of each agent developed in this work is very common (Iglesias et al., 1996) and its internal components are depicted in Figure 4-5. The **Communication module** is responsible for the interface with the environment (coding/decoding messages and network interface). The **Knowledge Database module** is where the agent stores the information about itself (self interests and services) and about what it knows from the environment (environmental restrictions, other agents and applications, their addresses, services provided by them and their interests). The **Control module** is responsible for the policies of the communications (order and selection of the messages to process and queue messages to send) and actions (implementation of the different criteria regarding the attention to the external requests).

For instance, the Farmer Agent seed actions are regulated by the environmental rules that forbid to seed bivalves in some areas. The “Actions Policy” element (in the Control module) must access the Knowledge Database module to know the allowed area to seed (“Environment Data” element) and the best quantity and type of bivalves to seed (“Self Data” element).

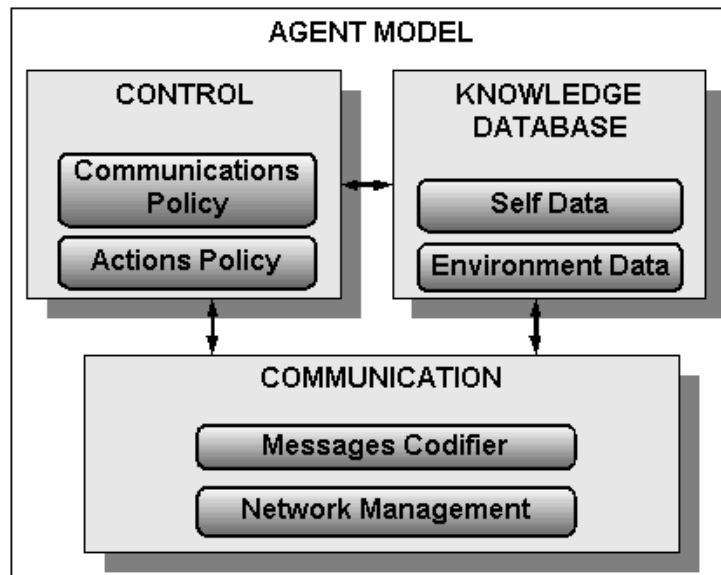


Figure 4-5: Generic agent model used in this work (Pereira et al., 2004b)

When multiple agents work together, their behaviours must be coordinated in order to reach a global result. Coordination is seen as the act of working together in a harmonious way to attain a common goal or agreement (Malone and Crowston, 1994; Reis, 2003). Sometimes, agents can cooperate to reach a common goal. However, often agents don't share one common objective and may have opposite interests. In these cases, there are conflicts to solve and a negotiation process must start to coordinate efficiently agents' actions. In the types of simulations used in this work, the existence of agents with different interests in the ecosystem use (industry, fishery, tourism...) will be an adequate way to treat the interactions between different activities, making necessary its coordination to provide an efficient shared use of the ecosystem, fulfilling the restrictions imposed legally.

4.4.1 ECOLANG

ECOLANG is a high level language especially developed to enable the interaction and the exchange of data between the components of the multi-agent simulation system – modules of the network integration infrastructure.

The first version of the language was presented in Pereira et al. (2005) and it follows a syntax based on the COACH UNILANG work (Reis and Lau, 2002). ECOLANG was designed for communications in TCP/IP networks and enables the applications belonging to the multi-agent simulation system to exchange data and other type of information related to the ecological domain. It fills the pre-requisites enounced before its development:

- High level language understood by software applications and humans;

- Simple syntax validation;
- Ontology oriented to aquatic systems;
- Easily adaptable to new actors added to the system;
- Independent from the objects' model and transport layer;
- Independent from any hardware or software platform, operating system or programming language.

The language describes ecological systems in terms of morphological and regional characteristics, and embodies the actions and perceptions of agents in messages. As ECOLANG messages are completely readable by humans and computers, it is very simple to debug and follow the flux of data exchanged in the network. The first formal set of instructions to use the ECOLANG messages was defined to transfer data between the EcoDynamo simulator and one agent, leading to call this agreement as the **EcoDynamo protocol**.

The specification of the ECOLANG messages uses the Backus-Naur Format (BNF) grammar definition (Backus et al., 1960; Ram et al., 1996), one of the best known meta-languages in the field of computer science. The notation was first referenced by John Backus and Peter Naur to describe the syntax of the programming language Algol 60 in an unambiguous manner. ECOLANG notation is an extension of the original BNF formalism, requiring only three primitive types (string, integer and real) and adding some new meta-symbols:

{ } – curly braces indicate usage of repetitive items (one or more times);

[] – square braces enclose types of values;

Terminal values use bold face letters.

The base syntax of each message is the following:

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

<ID> is the message identifier sent by the initiator – it is a sequential integer number controlled by each sender;

<SENDER> is the name of the message initiator application (source);

<RECEIVER> is the name of the message destination application;

<MSG_CONTENT> is the content of the message.

The ECOLANG messages can be divided in three parts:

- an **envelope** element – that delimits and enfolds the ECOLANG message;

- a **header** element – that identifies the message origin, sequence and destination;
- a **body** element – that defines the message contents.

The envelope (Figure 4-6) is composed by the word **message** and by the open and close curve brackets. The header is composed by the fields ID, SENDER and RECEIVER. The body of the message is the field MSG_CONTENT. The first word of MSG_CONTENT defines the message type and identifies the message semantic undoubtedly, in order to achieve independence from the implementation, allowing the construction of a generic protocol driver that is linked with each application – component ECDP.dll (cf. 4.3.3 – Generic DLLs).

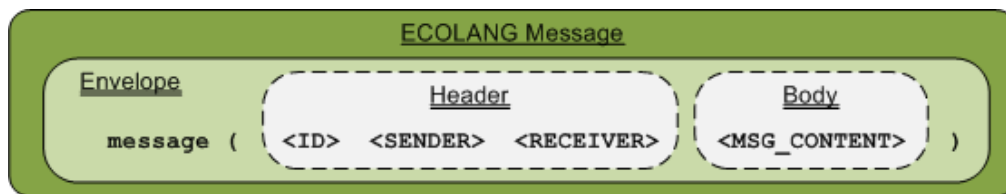


Figure 4-6: ECOLANG message structure

With ECOLANG messages it is possible to select the model to simulate, read the configuration of the model and change the scenario simulation, execute and pause the simulation runs, collect results from any region of the simulated area, define regions and sub-regions by name, aggregate regions into sub-domains, define events in the simulator to generate messages when some trigger conditions are achieved, and save the configuration of the database and the values of each variable in one specific simulation time step.

Messages exchanged by the applications cover several types: actions, perceptions, definitions, session and miscellaneous.

The **messages that indicate actions and perceptions** to/from agents are dedicated to each kind of agent present in the system. There are no restrictions to the messages that applications can use. Some care must be taken in the attempt to identify, as closely as possible, the type of action/perception with its name, avoiding duplication and ambiguous identification.

The **session messages** establish the sessions between agents or applications and identify the agent name, the computer where the agent is, and the port number in which agent “listens” the messages.

The **definition messages** include the definition of regions and information about the name and type of model loaded in the simulator, its dimensions, morphology and species of shellfish prevailing in it.

```
ECOLANG Messages

+++ Rx: +++
message(1 Develop_Agent EcoDynamo connect amcp11 172.22.128.30 49000)
--- Tx: ---
message(1 EcoDynamo Develop_Agent accept(1 ok))
+++ Rx: +++
message(2 Develop_Agent EcoDynamo model_name)
--- Tx: ---
message(2 EcoDynamo Develop_Agent model(2 RiaFOcidente2D100mSSLuisCycle))
+++ Rx: +++
message(3 Develop_Agent EcoDynamo model_dimensions)
--- Tx: ---
message(3 EcoDynamo Develop_Agent dimensions (3 182 300 1 2DH))
+++ Rx: +++
message(4 Develop_Agent EcoDynamo model_morphology)
--- Tx: ---
message(4 EcoDynamo Develop_Agent morphology (4 (0 56.173646) (1 56.173646) (2 56.173646) (3 56.173646) (4 56.173646) (5 56.173646) (6 56.173646) (7 56.173646) (8 56.173646) (9 56.173646) (10 56.173646) (11 56.173646) (12 56.173646) (13 56.173646) (14 56.173646) (15 56.173646) (16 56.173646) (17 56.173646) (18 56.173646) (19 56.173646) (20 56.173646) (21 56.173646) (22 56.173646) (23 56.173646) (24 56.173646) (25 56.173646) (26 56.173646) (27 56.173646) (28 56.173646) (29 56.173646) (30 56.173646) (31 56.173646) (32 56.173646) (33 56.173646) (34 56.173646) (35 56.173646) (36 56.173646) (37 56.173646) (38 56.173646) (39 56.173646) (40 56.173646) (41 56.173646) (42 56.173646) (43 56.173646) (44 56.173646) (45 56.173646) (46 56.173646) (47 56.173646) (48 56.173646) (49 56.173646) (50 56.173646) (51 56.173646) (52 56.173646) (53 56.173646) (54 56.173646) (55 56.173646) (56 56.173646) (57 56.173646) (58 56.173646) (59 56.173646) (60 56.173646) (61 56.173646) (62 56.173646) (63 56.173646) (64 56.173646) (65 56.173646) (66 56.173646) (67 56.173646) (68 56.173646) (69 56.173646) (70 56.173646) (71 56.173646) (72 56.173646) (73 56.173646) (74 56.173646) (75 56.173646) (76 56.173646) (77 56.173646) (78 56.173646) (79 56.173646) (80 56.173646) (81 56.173646) (82 56.173646) (83 56.173646) (84 56.173646) (85 56.173646) (86 56.173646) (87 56.173646) (88 56.173646) (89 56.173646) (90 56.173646) (91 56.173646) (92 56.173646) (93 56.173646) (94 56.173646) (95 56.173646) (96 56.173646) (97 56.173646) (98 56.173646) (99 56.173646) (100 56.173646) (101 56.173646) (102 56.173646) (103 56.173646) (104 56.173646) (105 56.173646) (106 56.173646) (107 56.173646) (108 56.173646) (109 56.173646) (110 56.173646) (111 56.173646) (112 56.173646) (113 56.173646) (114 56.173646) (115 56.173646) (116 56.173646) (117 56.173646) (118 56.173646) (119 56.173646) (120 56.173646) (121 56.173646) (122 56.173646) (123 56.173646) (124 56.173646) (125 56.173646) (126 56.173646) (127 56.173646) (128 56.173646) (129 56.173646) (130 56.173646) (131 56.173646) (132 56.173646) (133 56.173646) (134 56.173646) (135 56.173646) (136 56.173646) (137 56.173646) (138 56.173646) (139 56.173646) (140 56.173646) (141 56.173646) (142 56.173646) (143 56.173646) (144 56.173646) (145 56.173646) (146 56.173646) (147 56.173646) (148 56.173646) (149 56.173646) (150 56.173646) (151 56.173646) (152 56.173646) (153 56.173646) (154 56.173646) (155 56.173646) (156 56.173646) (157 56.173646) (158 56.173646) (159 56.173646) (160 56.173646) (161 56.173646) (162 56.173646) (163 56.173646) (164 56.173646) (165 56.173646) (166 56.173646) (167 56.173646) (168 56.173646) (169 56.173646) (170 56.173646) (171 56.173646) (172 56.173646) (173 56.173646) (174 56.173646) (175 56.173646) (176 56.173646) (177 56.173646) (178 56.173646) (179 56.173646) (180 56.173646) (181 56.173646) (182 56.173646) (183 56.173646) (184 56.173646) (185 56.173646) (186 56.173646) (187 56.173646) (188 56.173646) (189 56.173646) (190 56.173646) (191 56.173646) (192 56.173646) (193 56.173646) (194 56.173646) (195 56.173646) (196 56.173646) (197 56.173646) (198 56.173646) (199 56.173646) (200 56.173646) (201 56.173646) (202 56.173646) (203 56.173646) (204 56.173646) (205 56.173646) (206 56.173646) (207 56.173646) (208 56.173646) (209 56.173646) (210 56.173646) (211 56.173646) (212 56.173646) (213 56.173646) (214 56.173646) (215 56.173646) (216 56.173646) (217 56.173646) (218 56.173646) (219 56.173646) (220 56.173646) (221 56.173646) (222 56.173646) (223 56.173646) (224 56.173646) (225 56.173646) (226 56.173646) (227 56.173646) (228 56.173646) (229 56.173646) (230 56.173646) (231 56.173646) (232 56.173646) (233 56.173646) (234 56.173646) (235 56.173646) (236 56.173646) (237 56.173646) (238 56.173646) (239 56.173646) (240 56.173646) (241 56.173646) (242 56.173646) (243 56.173646) (244 56.173646) (245 56.173646) (246 56.173646) (247 56.173646) (248 56.173646) (249 56.173646) (250 56.173646) (251 56.173646) (252 56.173646) (253 56.173646) (254 56.173646) (255 56.173646) (256 56.173646) (257 56.173646) (258 56.173646) (259 56.173646) (260 56.173646) (261 56.173646) (262 56.173646) (263 56.173646) (264 56.173646
```

The first message received by the simulator is a connection request, made by “Develop_Agent” application to “EcoDynamo” application. Develop_Agent is in “amcp11” computer, IP address 172.22.128.30 and accepting connections in port 49000. After the acceptance of the connection by EcoDynamo (second message in the trace window, first message transmitted), Develop_Agent asks for model configured in simulator (messages “model_name”, “model_dimensions” and “model_morphology”) receiving the correspondent answers from EcoDynamo (messages “model”, “dimensions” and “morphology”).

4.4.2 EcoSimNet Framework

- 121 -

protocol, was named EcoSimNet (**E**co**l**ogical **S**imulator **N**etwork). Each application belonging to the EcoSimNet system must fulfil the following rules:

1. It must have a unique name within all the applications integrated in the network infrastructure – the name must be a string with no spaces;
2. It must reserve a free TCP/IP port to accept TCP/IP client socket connections and establish the communications channel for ECOLANG messages;
3. The first message sent to any member of the network must be “connect” – after receiving the “accept” message it is considered as one EcoSimNet member;
4. Before leaving the EcoSimNet system, it must send the “disconnect” message to all connected applications – after receiving the “accept” message it is considered out of the EcoSimNet system.

These simple rules constitute the EcoSimNet framework integration specification. The integration of each new agent or application into the EcoSimNet multi-agent simulation system must fulfil the EcoDynamo protocol specification. Basically, it has to translate all its actions over the environment to ECOLANG messages, and it has to interpret the received ECOLANG messages as perceptions from the environment or external commands – messages are translated to internal perceptions and processed as convenient.

ECDPForm Component

One special component (*ECDPForm*) was developed in C++ to provide the integration of the EcoSimNet internal service. It is responsible for the connection of any new agent or application to the network, facilitating its interoperability with other modules. This component is composed of four fundamental sub-components:

- One complete multi-threaded TCP server: uses a thread to listen/wait for client connections, and allocates a separate thread to handle each client connection to the server;
- One complete TCP client including sockets support: it is used as an ancestor class for ECOLANG specific protocol implementation;
- One cyclic timer, with 20 ms interval, controlling the communications flux between the above TCP components;
- One multiline edit control to display received and transmitted messages (Figure 4-7).

ECDPForm component is linked with the EcoDynamo protocol DLL (*ECDProtocol.dll*) and supplies an API to the developer that facilitates its attention to the processing of the ECOLANG messages – all the stuff for the connection to the system (required by the EcoSimNet framework and EcoDynamo protocol) is hidden by the component. All the applications belonging to EcoSimNet, and developed in C++, must include one component that inherits from *ECDPForm*.

The session established by two applications, with the *ECDPForm* component, maintains two connections between the peers in order to reduce the latency time in the process of send/receive messages, and to eliminate the connect/establish/disconnect time needed to manage a connectionless session.

Integration and Interoperability

The communication between all components integrating the EcoSimNet structure is peer-to-peer which avoids the necessity of a Domain Name Service for the agents and applications. Of course this implies that each agent, when arrives to the network, must know at least one element that is already in the framework. ECOLANG provides messages to query for known agents in the network and, as good neighbours, agents will enlarge its knowledge about the neighbourhood population in a collaborative fashion.

EcoSimNet platform is used for controlled experiments and, therefore, sophisticated mechanisms for integration of elements are dispensable. Comparing with FIPA specifications, used to promote the interoperation of heterogeneous agents and services:

- an “agent resource broker” (ARB) and a “wrapper service” (FIPA, 2001a) are not necessary since agents know exactly where are the information and services they need;
- an “ontology” is embedded in the ECOLANG definitions, so it is not necessary to have “ontology agents” to provide “ontology services” (FIPA, 2001b) to the network;
- the information contained in ECOLANG messages is simpler than those documented in FIPA ACL specification (FIPA, 2002) for communication between agents - although the format is not similar, first word of the field MSG_CONTENT acts as a kind of *performative* of the ACL message, identifying the type of message.

New applications are integrated in the network very easily, simply by sending the **connect** message to start collaboration, and sending **disconnect** message to leave the collaboration.

The rules to integrate the EcoSimNet framework do not restrict the number of application types in the network – it can support several simulators, multiple agents with the same objectives, and does not require that all applications deal with the same model. The infrastructure presented in this document acts as an integration platform for simulation and decision support, where the applications interoperate graciously with simple messages. Figure 4-8 represents one possible configuration of the system, with several agents representing the human interests, and multiple simulators to enable the parallel simulation of different scenarios.

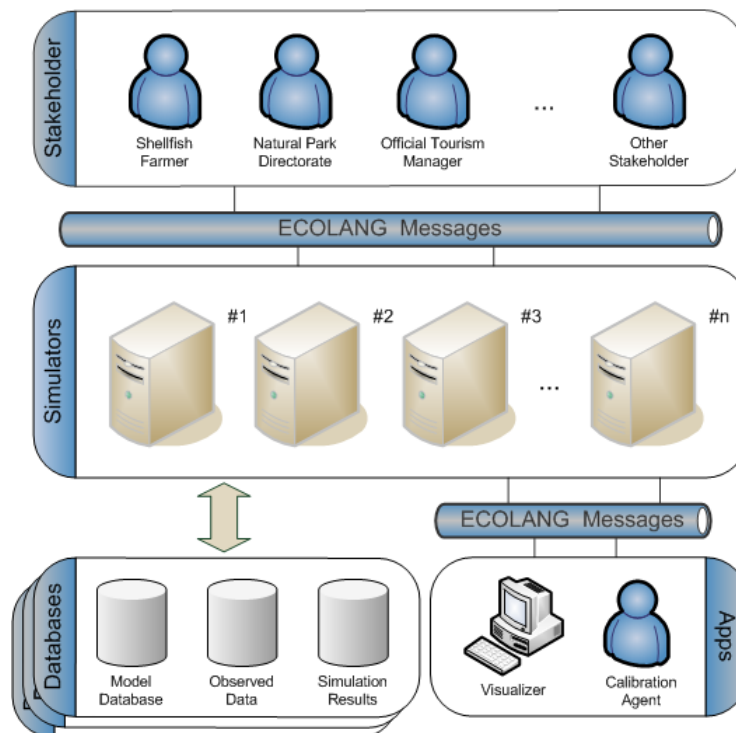


Figure 4-8: EcoSimNet architecture

In this configuration, it is distinctly visible that the only application that has access to the model database is the simulator. From the point of view of any other application in the network, the database of the system is in the simulator and there is no way to access directly the model configuration unless communicating that intention to the simulator through ECOLANG messages. This option makes sense, due to the nature of experiments this framework is used for, ensuring true independence between the simulated ecosystems and the agents. The knowledge about the ecosystem and how the model objects interact internally, working as a whole, is not passed to the agents that, as high level applications, are guided by specific goals to achieve their maximum performance or satisfaction, using the information contained in the simulators.

Another advantage of this approach is the guarantee of independence between the agents and their possible collaboration, allowing the exploitation of machine learning techniques (Michalski et al., 1984; Džeroski, 2001; Russel and Norvig, 2002).

Within the simulation domain, particularly when dealing with distributed simulations, there are no standard solutions for questions like integration and interoperability of heterogeneous simulators. EcoSimNet is a framework that enables the construction of a network with different simulators and sub-networks, integrated through the exchange of data, maintaining each sub-network with self-control but having ECOLANG as the common communication language.

Some multi-agent architectures comprise multi-domain networks and inter-domain coupling to assure interoperability between heterogeneous systems. Each single domain has its proper language of communication to exchange data between its members and, if other domains have different communication languages, inter-domain communication must be assured by a supra language of communication. KQML (Finin et al., 1993) is an excellent example of inter-domain language easily implemented in one special interpreter agent that translates outside messages (in KQML) to inside semantic, and vice-versa. None of these abstractions are used in the context of this work because the domain is well delimited, but it is perfectly feasible to extend this framework to fit in one multi-domain architecture, and exchange information with external applications or networks.

If we want to go further in the formalisation of the EcoSimNet, it fits perfectly in the abstract notion of the high-level architecture (HLA), a standard for interoperation between distributed heterogeneous simulators which are developed with different languages and platforms (Hong et al., 2007). The HLA specification was initially developed under the aegis of DMSO (Defense Modeling and Simulation Office)—renamed in 2007 M&S CO (Modeling and Simulation Coordination Office), and part of the Department of Defense of USA – afterward adopted by the OMG as the standard IEEE 1516 (IEEE, 2000c, 2000b, 2000a).

The HLA specification defines a runtime infrastructure (RTI) with interface rules to integrate simulators or applications (federates) that interoperate exchanging information. The notion of federation is also defined as an abstract organizational entity, that enfolds groups of federates acting as a whole organization or simulation island, that supplies information to other federations. The HLA does not provide a specific implementation or mandate to the use of any particular programming language, neither for RTI or federate applications. Figure 4-9 shows one configuration of EcoSimNet network that fits the HLA architecture paradigm.

4.4.3 Development Agent

In order to test the multi-agent simulation system, and the functionalities of the represented agents, a simple application was developed and integrated in the experimental system. This application (called Development Agent) implements all the possible ECOLANG messages and provides the system with a simple and easy-manipulated visual interface.

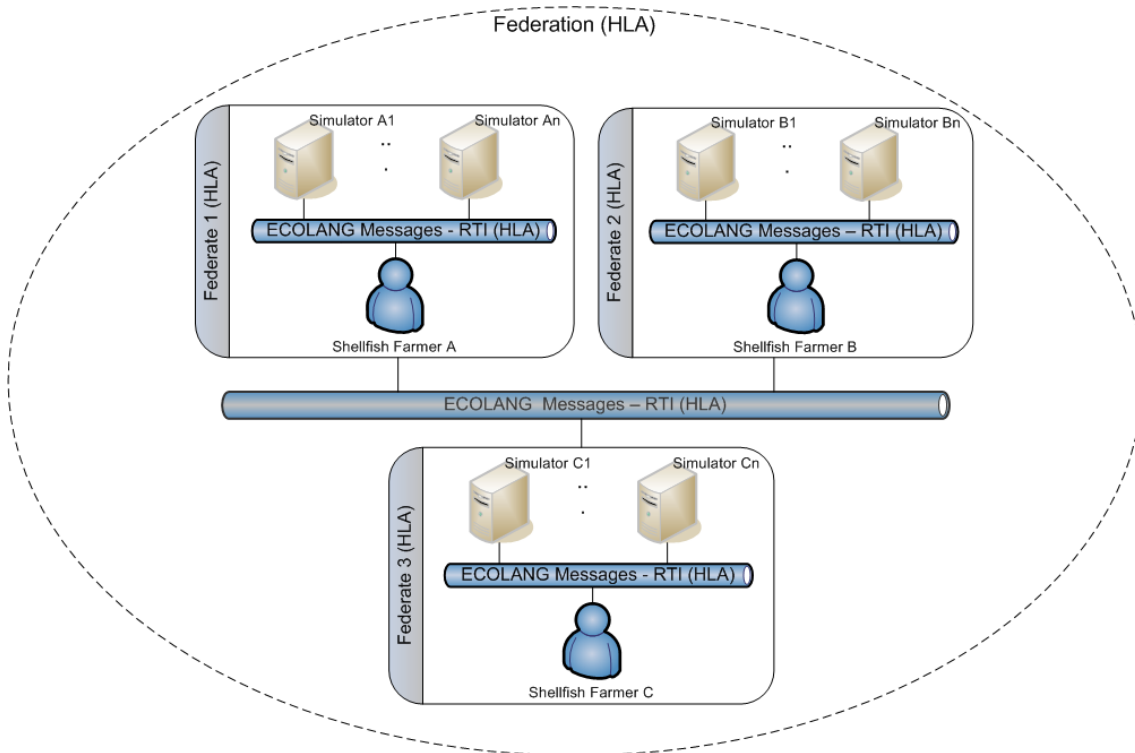


Figure 4-9: EcoSimNet as a logic HLA architecture

Visualization System

The visualization system is integrated in the Development Agent and explores one simple graphical interface to facilitate simulation testing and debugging processes. It was enhanced in some graphical aspects, from its initial implementation, supplying the user with exact notion of the two-dimensional space of the domain with distinction between land and water areas. The two-dimensional representations are like aerial maps of the regions with flat land and variable water depth. Figure 4-10 shows the morphology of western Ria Formosa, with benthic species areas, where the different colours (or the gray gradient in black and white documents) show the occupied areas and water depths.

The original colours were defined for visualization in a RGB monitor but, when exporting the image of the ecosystem to a document, the original colours can be hardly distinguishable, especially when the document is black and white and the gray scale did not reveal the relevant

features of the image. The visualization system provides the user with the possibility of changing colours, adjusting them to the sensibility of the user and the desired highlight.

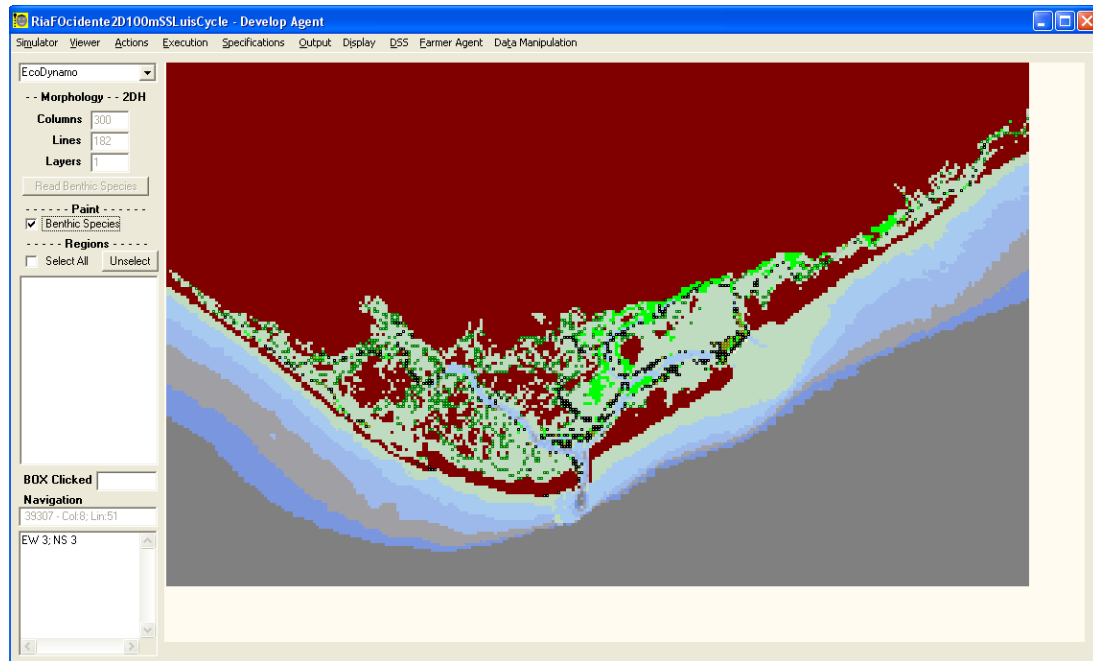


Figure 4-10: Ria Formosa lagoon with customized colours adapted to B/W documents

With the simple 2D visualization of the model domains depicted in Figure 4-10 it is easy to have some understanding of the site morphology, even without ever having seen the sites – next figures show two more models managed by the visualization system of the Development Agent and perfectly readable in B/W documents.

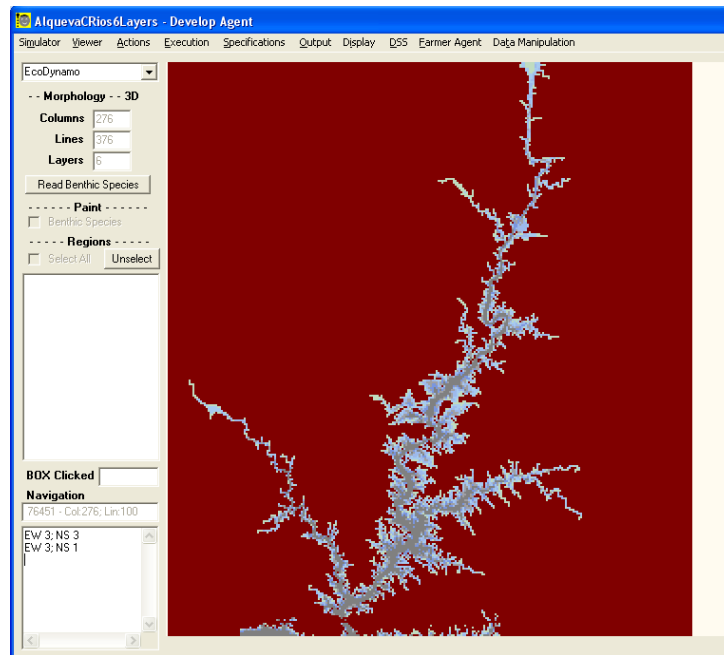


Figure 4-11: 2D representation of the Alqueva dam (Alentejo-Portugal) 3D model

The land area is originally coloured in maroon and is seen as the darkest gray in B/W pages.

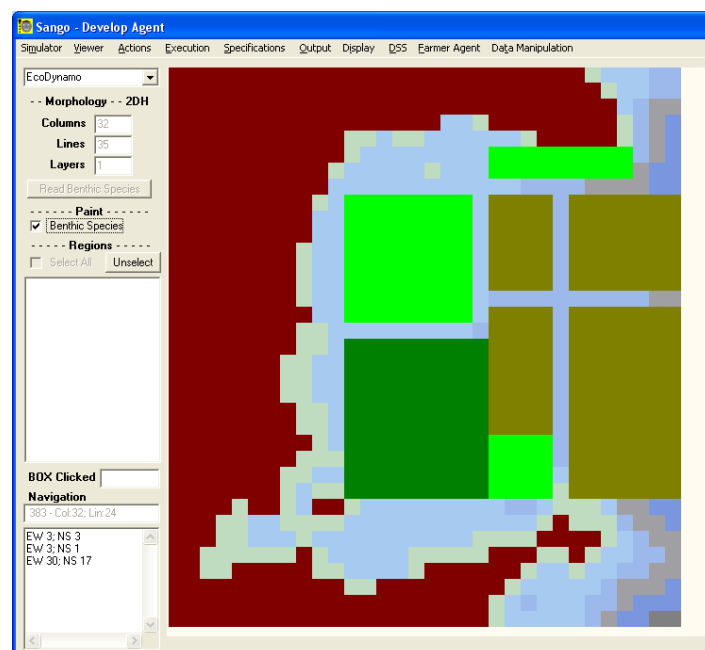


Figure 4-12: Sungo Bay 2D model with benthic species exploration areas

When the domain is extensive, like the cases of Ria Formosa (Figure 4-10 - 300 columns, 182 lines, 1 layer) or Alqueva dam (Figure 4-11 - 276 columns, 376 lines, 6 layers), it is very hard to distinguish perfectly the individual cells, especially if the monitor resolution is not very large. A zoom option is provided to clarify the visualization areas. This option can be executed repeatedly (Figure 4-13) until one clear image is shown to the user (Figure 4-14), namely the areas with benthic species clearly visible and distinguishable as well as navigation channels.

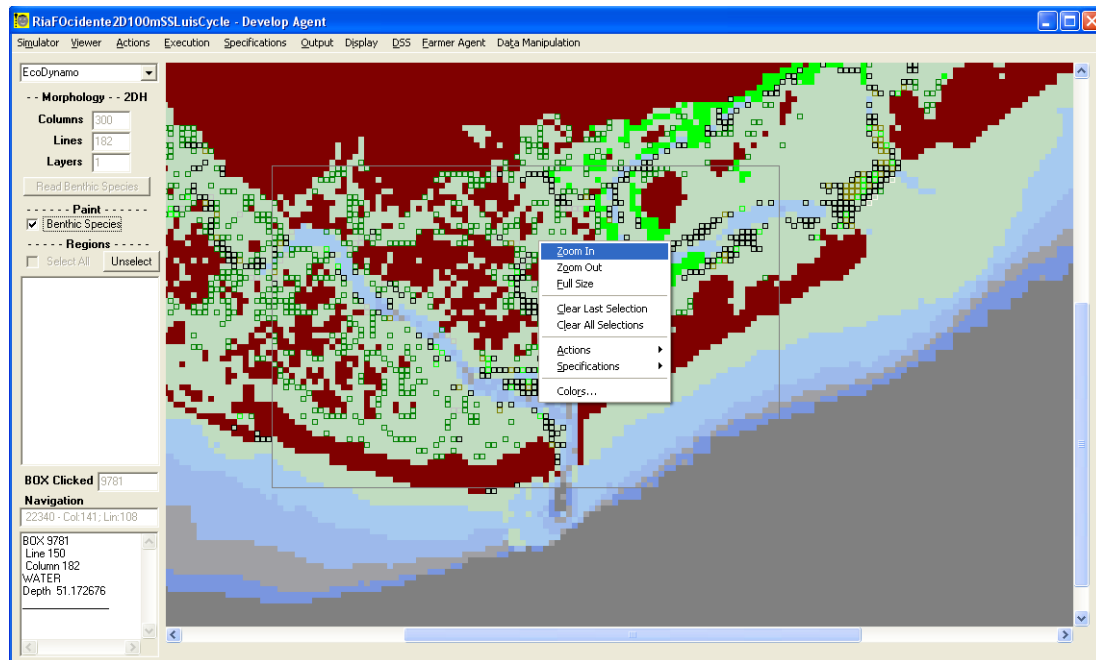


Figure 4-13: Zoomed area of the simulated model

This application enables the user to aggregate several cells into regions and, inside them, manipulate some characteristics of the model (for instance, bivalves' density, seeding and harvesting properties, depth of the cells). The regions can also be read from and saved in text files.

A 3D visualization autonomous tool is in development and it will allow adding distinct functionalities to the 3D simulation models.

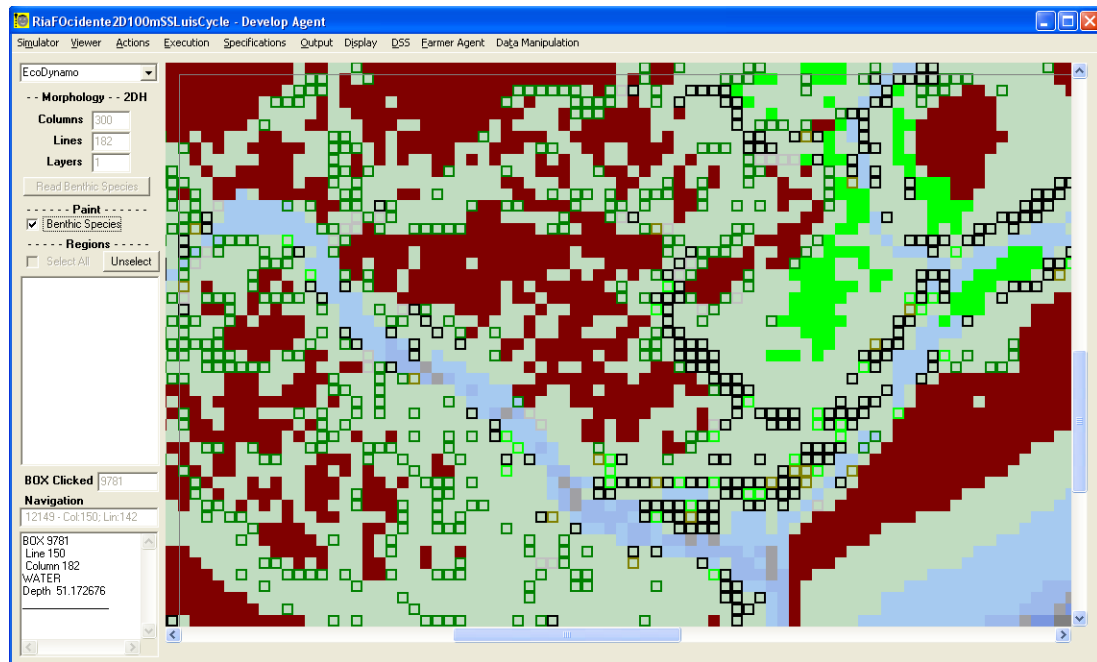


Figure 4-14: A deeper zoom of the simulated model area

ECOLANG Message System

The development agent (DA) is used to test the ECOLANG messages and EcoDynamo protocol dynamics, supplies to the user options to manipulate all the actions that could be performed over the simulator, and presents the answers received from the simulator, the perceptions that agents will sense.

The navigation in this application is very intuitive. When the application starts, the user selects the simulator he wants to connect. Automatically, the simulator send the model in simulation: the system morphology is received (name of the model, geometric representation, dimensions, number of columns, lines and layers, and box depth of each cell) - Figure 4-15 shows the messages exchanged between Development Agent and EcoDynamo (view from the EcoDynamo side), after the connection and during the initialisation of the Ria Formosa model represented in Figure 4-10. After the last morphology message received, the model area is represented with a 2D perspective in the visualization panel of the application.

are essential to the DSS module (cf. 4.5) to generate and store the values of relevant variables, used as data for analysis of the different scenarios simulated for ecosystem management.

4.4.4 Optimization and Machine Learning

The inclusion of advanced capacities of optimization and machine learning in the built agents is, perhaps, the most sophisticated and challengeable feature in the entire project. Some of the machine learning techniques are implemented and tested in the Farmer Agent. Negotiation will be integrated later when several different agents appear in the system.

The vast majority of problems related with management of coastal ecosystems deals with optimization techniques. As focused in section 3.3 , local search methods, exploring the neighbourhood, form a general class of heuristics to obtain a good (near-optimal) solution with a reasonable computational time. For a maximization/minimization problem, the local search ascent/descent method is the simplest neighbourhood search algorithm, also known as hill climbing. It starts from an initial solution (random or predefined), as its current solution, and then explores its vicinity using simple mechanisms to generate neighbouring solutions. Neighbours are then accepted to replace the current solution if they improve it. The search continues until no further improvement can be made and the algorithm terminates with a local optimum, which can be very far from optimality (Russel and Norvig, 2002). In order to avoid such disadvantages, while maintaining the simplicity and generality of the approach, most metaheuristics are based in the concepts (Osman and Laporte, 1996: 515):

- Start from good initial solutions, generated intelligently using GRASP (Feo and Resende, 1995) or space-search methods.
- Use learning strategies, like neural networks or tabu search (Glover, 1989, 1990), gathering information during the algorithm's execution in order to guide the search to find possibly better solutions.
- Employ non-monotonic search strategies that accept neighbours based on hybrid modifications of simulated annealing and tabu search, among others (Kirkpatrick et al., 1983; Eglese, 1990; Mishra et al., 2005).
- Introduce a more complex neighbourhood mechanism, such as ejection chains and compound moves with proper data structures.
- Employ a solution-generation mechanism that works on a set of solutions rather than a single solution, like genetic algorithms, evolutionary programming or scatter search (Holland, 1975; Cardon et al., 2000; Glover et al., 2003).

The methodology followed in this work, for Farmer Agent, uses some of the concepts listed above and mix them with the intention of achieving a near-optimal solution in a faster way, in order to simulate a lower number of scenarios.

4.4.5 Farmer Agent

The main goal of the Farmer Agent (FA) is to maximize profit from bivalve culture. The ecosystem has specific regions available to seed bivalves, a maximum available area to explore and the FA tries to optimize the bivalve growth in order to harvest them with the largest possible size.

The first prototype of the Farmer Agent tries to find the better combination of bivalve culture locations, minimizing the impact in the ecosystem water quality and getting as close as possible to the ecosystem carrying capacity. As bivalve's growth rate is influenced by the environment (water quality) and the entire neighbourhood, the agent's decision is not simple. In the context of aquaculture, carrying capacity can be defined as the ability of the system to support shellfish production (Raillard and Ménesguen, 1994), the maximum standing stock of a particular species that can supported by the ecosystem at which the annual production is maximized (Bacher et al., 1998) without negatively affecting growth rates (Ferreira et al., 1997; Duarte et al., 2003).

Almost all of the techniques used to search optimum or quasi-optimum solutions are based on iterative processes, where the choice of the next combination to test is based on the value of an evaluation function that determines the quality of the solution (Jones, 2005). The determination of this evaluation function will turn the search algorithm more or less intelligent. However, there is a close dependence between the algorithm to use and the problem to solve.

The FA tries to apply a distinct solution, enabling the free combination of the different techniques through the search time. It includes some known search algorithms based on Simulated Annealing and integrating Tabu Search (Siarry et al., 1997; Youssef et al., 2001; Mishra et al., 2005), Genetic Algorithms (Holland, 1975; Cardon et al., 2000; Amirjanov, 2006) and Reinforcement Learning (Sutton and Barto, 1998), and one intelligent program to select the solutions to test (different scenarios to the EcoDynamo simulator). The initial configuration is made by the user, indicating the sequencing of each algorithm and its action time. The FA chains the algorithms in order to enhance their advantages and to compensate their

weaknesses. The first version of this agent was presented in a scientific workshop (Cruz et al., 2007) with the simplified architecture shown in Figure 4-16.

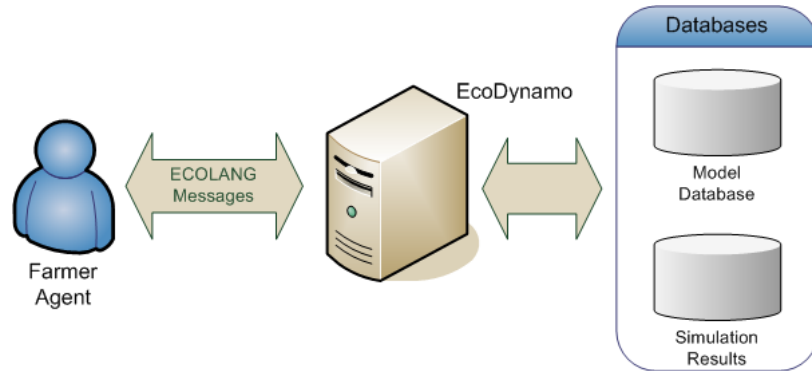


Figure 4-16: System architecture for experimental Farmer Agent

The sequence diagram for the optimization process lead by FA is pictured in the following figure:

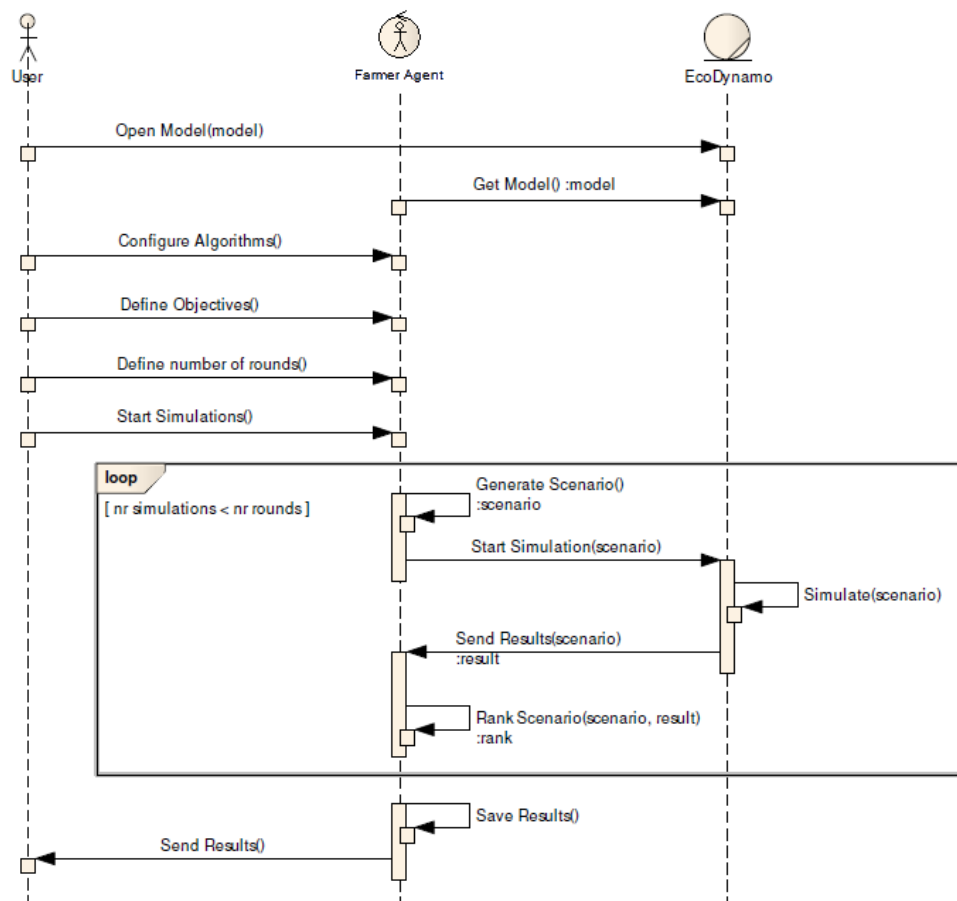


Figure 4-17: Farmer Agent optimization process - sequence diagram

One of the constraints pointed out in the conclusions of that study was the time consumed by each simulation run, and the large number of simulations needed to find the best solution. As

the time consumed by each simulation leaves the FA in idle state most of the time waiting for the simulation results, it was relatively simple to generalize the optimization process to support more than one simulator controlled by the same FA. The use of a distributed simulation network to speed up the decision process was the natural follow up. The architecture takes advantage of the capacity to perform parallel simulations of different exploration scenarios and their analysis in real time. Figure 4-18 shows the solution adopted.

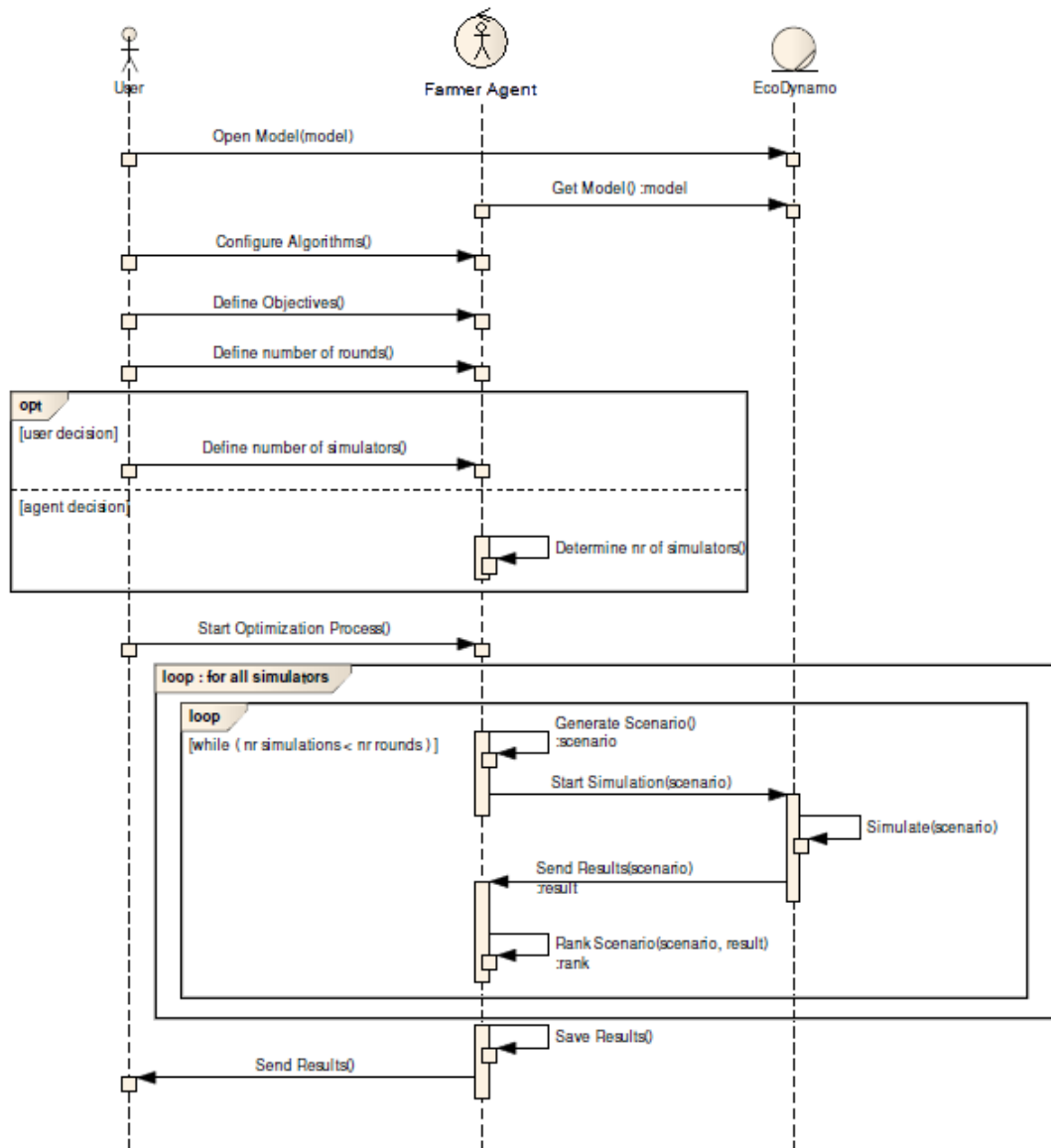


Figure 4-18: FA optimization process – general sequence diagram

The sequence diagram of the internal loop, with the ECOLANG messages exchanged between FA and a simulator, is depicted in Figure 4-19. The optimization cycle starts with the regions definition. The region names remain during the optimization cycle. For each iteration, the model is initialised and one simulation step is performed to populate the state variables of all

objects dependant from the forcing functions. The FA seeds the shellfish species in selected regions and the simulation runs N steps. When the simulation ends (reception of the “end_step” message), FA harvests the shellfish species from the regions where they were seeded and stop the simulation. Before the next iteration, FA saves the results, selects a new set of regions and repeats the procedure, until the number of iterations reach the number preconfigured.

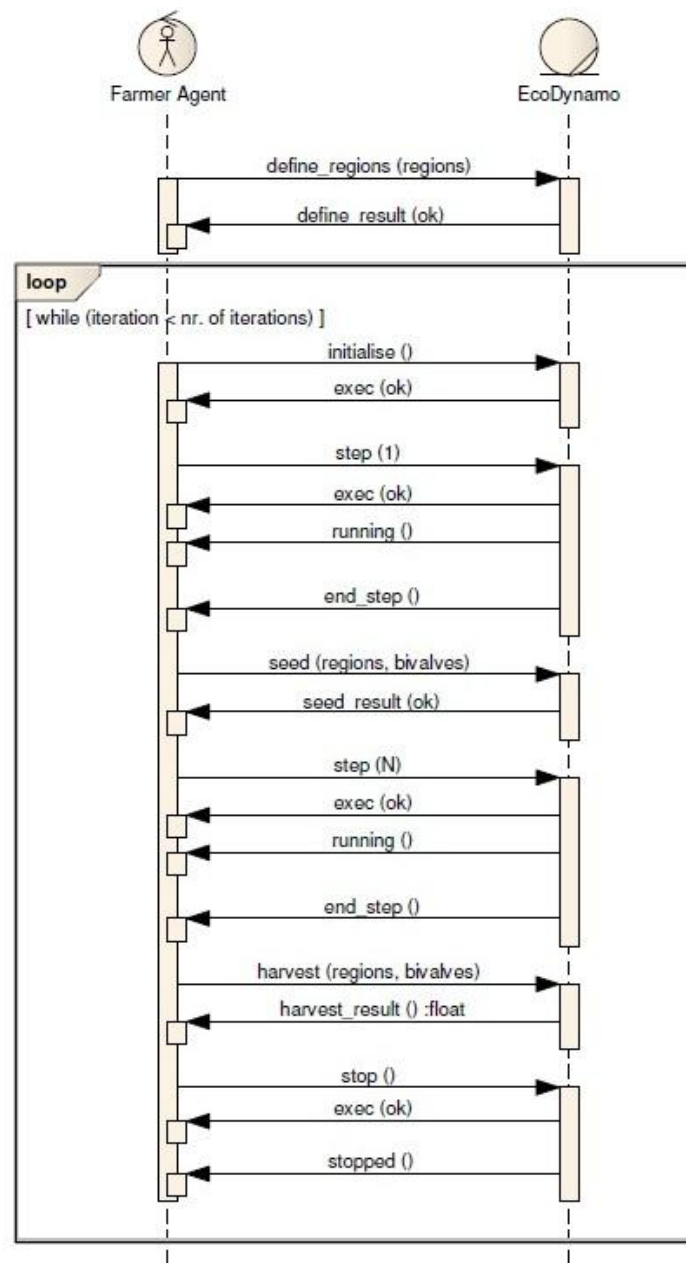


Figure 4-19: ECOLANG messages exchanged between FA and EcoDynamo during one iteration of the optimization cycle

Starting from the basic architecture of EcoSimNet, the concept of computing clusters and parallel computation is exploited, creating islands where an agent monitors a set of simulators

and simultaneously simulate different scenarios (Figure 4-20). One of the similar agents must take the lead role, encouraging other agents to collaborate in the achievement of the desired results as quickly as possible. The coordinator defines synchronization points during the course of the experiments, collecting the results obtained by the agents, and adopting the best result as the starting point for a new round of independent simulations. The stop order is given by the coordinator when a criterion of optimization is achieved, a maximum number of simulation rounds is reached or a time limit is exceeded

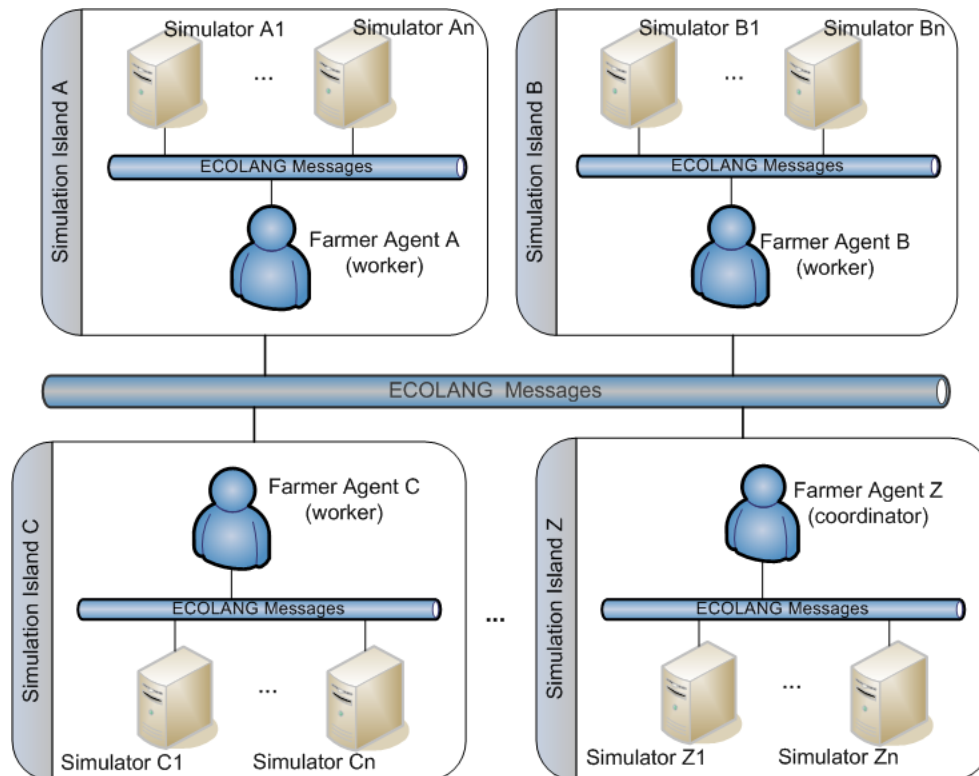


Figure 4-20: EcoSimNet with "simulation islands"

In this architecture it is assumed that each simulator has access to the model database, accessible by the network. Before the system starts, each agent is provided with a script file that defines its identification and its role.

Every agent in the system has the same code, with a system of customizable tactics that intends to simulate the reasoning of human shellfish farmer. The base algorithm developed is a simple hill-climbing optimization algorithm, based on simulated annealing with Monte Carlo probability (Kirkpatrick et al., 1983) and tabu search (Mishra et al., 2005) – the agent seeks iteratively a new solution and saves the one with higher quality as the best. There are a set of configurable optimization parameters that can be activated to influence the selection of the best algorithm to be applied. The generation of new solutions was facilitated and improved by the inclusion of known optimization algorithms like genetic algorithms (Holland, 1975) and

reinforcement learning (Kaelbling et al., 1996; Sutton and Barto, 1998) that can be triggered in any stage of the process. The system of customizable tactics can activate (automatically or configured by the user) any one of the implemented algorithms during the experiments, with the possibility of combining more than one algorithm involved in the selection of the best solution.

The software implemented in the FA is able to communicate with multiple simulators and has a decision-making process based on a parallel simulated annealing algorithm, that integrates, in real-time, the results generated by the different simulators.

Before the start of the optimizations, the user configures the number of simulation steps to be used in the simulations, the initial shell weight, meat weight and density of bivalves' species to study, and the solution domain within the model database. The user can choose between optimize farming areas or farming densities, with mono or polyculture.

Customizable Tactics: Implemented Algorithms

In the field of ecological systems it is very unusual to reach only one solution, even when starting with the same initial solution. It is enough that the system gives an idea about what could be a good solution, regardless of the initial starting solution. A system of customizable tactics was developed to implement the sense of intelligence, allowing different multi-objective optimum solution finder techniques to be applied simultaneously or at different times during the optimization process. The following algorithms were implemented in this agent:

FARMERSA - SIMULATED ANNEALING

The implemented Simulated Annealing (FarmerSA) follows the usual guidelines as documented in the literature (Kirkpatrick et al., 1983) and in section 3.3.2, allowing the configuration of the typical parameters – the initial temperature, the final temperature and its descent rate, as well as the maximum number of iterations desired (Figure 4-21). The Monte Carlo probability of accepting worse solutions as starting points for new iterations can be disabled, transforming FarmerSA in simple hill climbing where only solutions with higher quality are approved. When the problem is optimization of bivalve's farming areas, the notion of neighbourhood breadth is also parameterized.

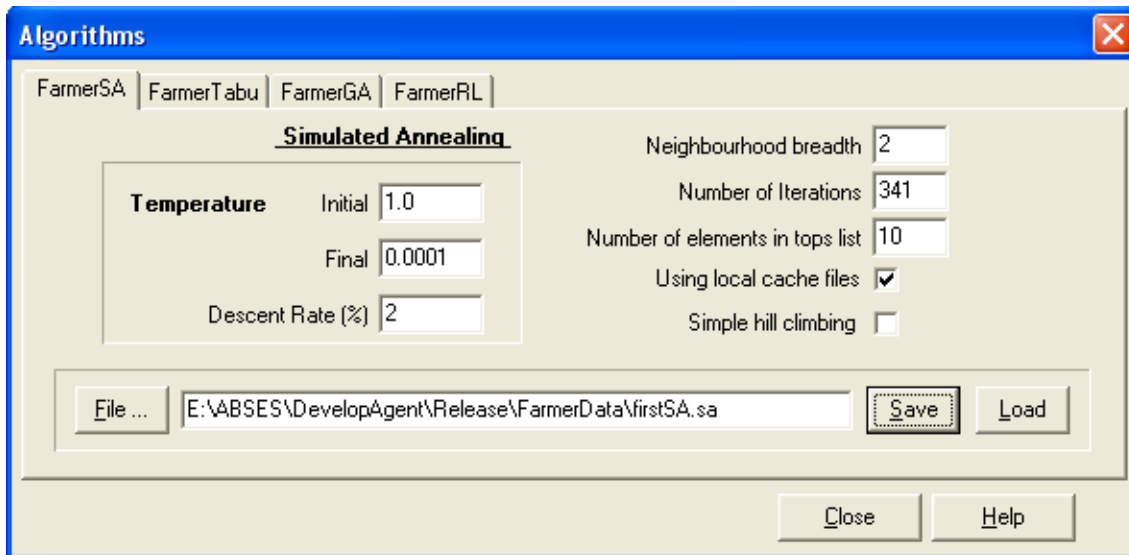


Figure 4-21: Initial configuration of FarmerSA

FARMERTABU - TABU SEARCH

The adaptation of Tabu Search implementation (FarmerTabu) is coupled with simulated annealing (Mishra et al., 2005), and is based on the maintenance of a hash list with all the previously tested solutions, so that when it is enabled, it simply prevents the simulation from choosing a previously tested solution as the next solution to test. It also keeps a counter of how many times the algorithm has taken to choose a previously tested solution from its choice of neighbours. It is possible to define a threshold value for the counter of refused solutions to activate a special mutation factor that serves to stir the process into other solution search paths, moving from paths apparently over explored without much success (Figure 4-22).

The FarmerTabu algorithm can be toggled on or off at any point in time of the process, and remains active or inactive from that point.

FARMERGA – GENETIC ALGORITHMS

The FarmerGA implementation maintains a list of best solutions found so far and crossbreeds them to form new combinations of regions. The number of solutions to take into account as parents and the number of new breeds are defined as parameters. Unlike common Genetic Algorithms (Holland, 1975), this implementation does not account for any mutation factor (Figure 4-23).

FarmerGA is used during optimizations related with bivalves' distribution within areas of occupation, and the way it crossbreeds the parents list is based on the quality hierarchy and one configurable parameter. In rough terms, it attempts to breed the best solution with all

others in the list of best solutions, the second best with half of all the others, the third best with a third of all the others and so on, selecting the best individual locations amongst both parents.

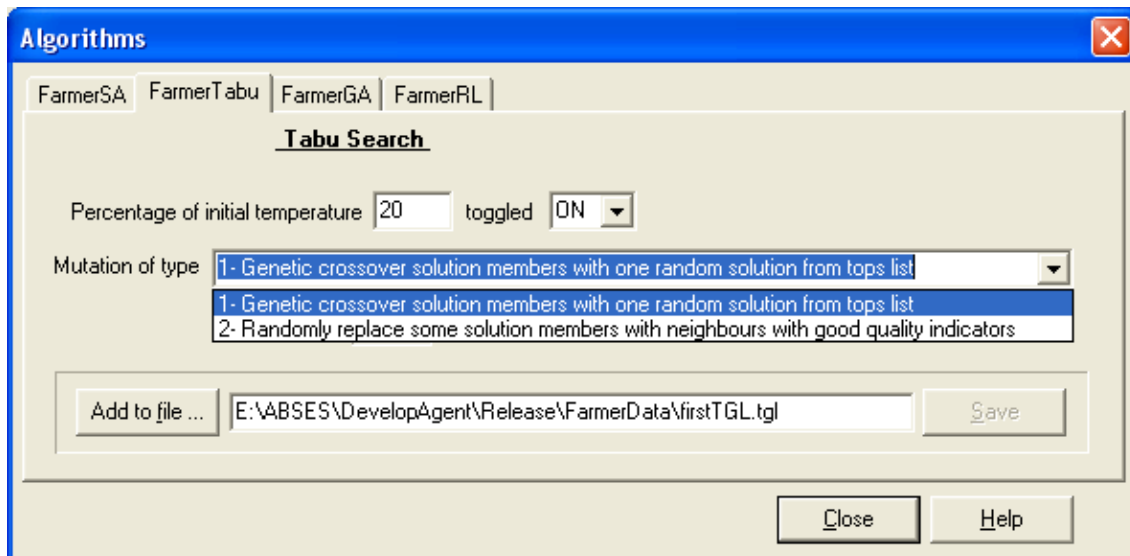


Figure 4-22: Configuration of FarmerTabu

When FarmerGA is toggled on, the process will trigger the running of genetic cross breeding among a list of good results. This operation must be done only after the agent had time to collect a list with sufficient good results in order to build a large and wide spread list of children solutions. Genetic crossing is applied during only a few steps (typically three or four).

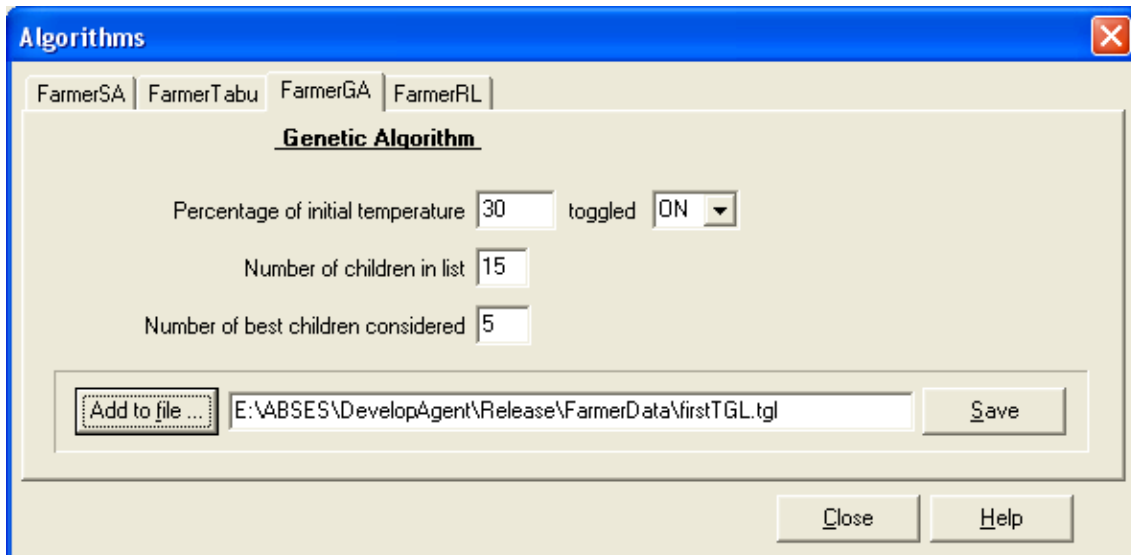


Figure 4-23: Configuration of FarmerGA

FARMERRL – REINFORCEMENT LEARNING

While RL intends to represent Reinforcement Learning, this implementation is somewhat far from the definition (Kaelbling et al., 1996; Sutton and Barto, 1998): a way of programming agents by reward and punishment without needing to specify how the task is to be achieved.

When the problem is the optimization of bivalves' farming areas, this implementation maintains a neighbours list for each possible region, containing information on its neighbourhood locations quality for farming. As the test iterations occur, calculated weights are added or subtracted to the quality of the areas selected in the tested solution, increasing or decreasing its value. The value of the weight depends on the geometric distance between the farming result of an individual area and the average amongst all the zones of the tested solution. The farming quality value of each region is taken into account when the next selection of other neighbouring solutions for testing is performed. As more iterations occur, the quality values of good seashell farming regions gets higher and the quality values of bad seashell farming regions gets lower, thus restraining the scope of search to better regions. The strength of the weight is a configurable parameter and can vary in real-time, as well as the way to choose the next solution. The definition of neighbour solutions also influences greatly on how this optimization will perform. FarmerRL has eight different ways to choose the next neighbour solution (Figure 4-24):

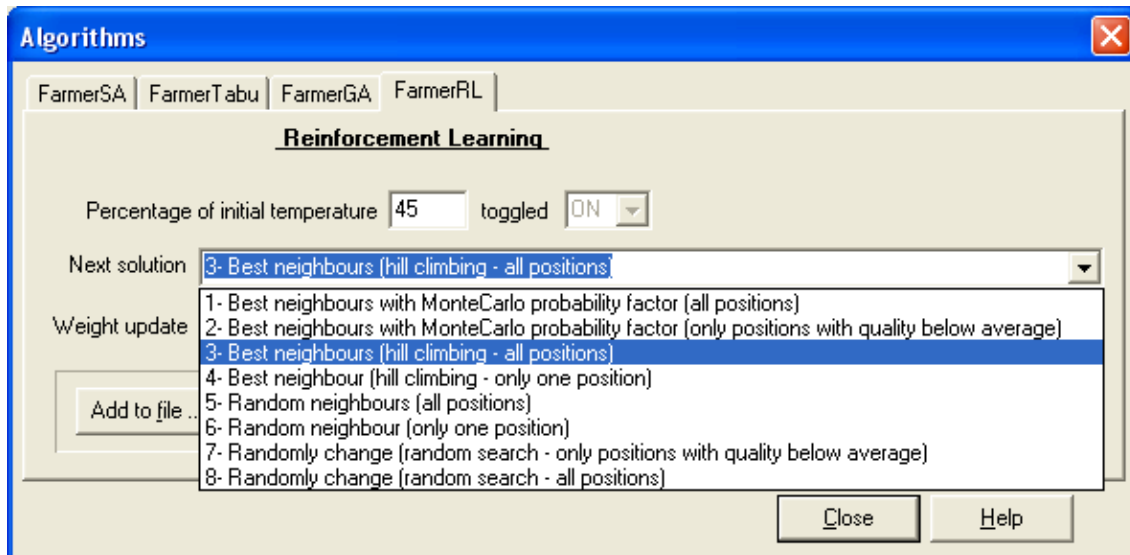


Figure 4-24: Configuration of FarmerRL algorithm with next solution options visible

- (i) Choose the best neighbour for each position, using Monte Carlo probability factor.
- (ii) Choose the best neighbour for each position, using Monte Carlo probability factor for the positions with quality below average.
- (iii) Choose the best neighbour for each position, without Monte Carlo probability factor.
- (iv) Randomly change only one position by the best neighbour, without Monte Carlo probability factor.
- (v) Change all positions by random neighbours.
- (vi) Randomly change only one position by one of its neighbours.
- (vii) Randomly change only positions with quality below average.
- (viii) Randomly change all positions.

Like FarmerTabu, the FarmerRL algorithm can be toggled on or off at any point in time of the optimization process.

Customizable Tactics: Tactics System

The implemented algorithms for optimization can be enabled throughout the iterative search process to influence the choice of the next solution to test - it is possible to start the process with a random search algorithm and switch to a hill climber algorithm a quarter or half way through the process, and so on. This choice of architecture was based on the fact that different algorithms produce different results depending on the properties of the problem itself that unavoidably affect the performance of the final result. A system of customizable tactics was developed to permit real-time reconfiguration to alternate between the different algorithms, customizing the algorithms' interventions and the values of their parameters to allow proper

fine tunes and grant even better possible results. This customization can be predefined by the user.

The base algorithm FarmerSA is always applied and its configuration is saved in a text file. For the example shown in Figure 4-21 the file *firstSA.sa* will contain the line:

```
sa 1.0 0.0001 0.980 2 341 10 1 0
```

where the values that follow *sa* represent the parameter values in the window fields.

The remaining algorithms can be defined more than once per optimization process, typically representing reconfigurations of the parameters at different moments of the process, and saved in a specific tactics file. This file contains text lines, referred as toggle lines, each representing one algorithm configuration at different steps of the process. The file starts with the keyword *init*, ends with the keyword *eof*, and each inner line begins with the keyword *tog* followed by the algorithm number (0 for FarmerTabu, 1 for FarmerGA and 2 for FarmrRL) and by three or four parameters, depending on the algorithm used. One example of the tactics file *firstTGL.tgl*, could be:

```
init
tog 0 0.0 1 0.8 1
tog 1 0.4 1 15 5
tog 2 0.5 3 0.05
eof
```

The complete configuration of the optimization process is grouped in a text file (*config.cfg*) that gathers the duration of the simulations (number of steps) and the type of optimization desired to the algorithms previously mentioned. Example of *config.cfg* file:

```
initSA 1.0 0.0001 0.980 2 341 10 1 0
simulation 86400 0
togglesfile firstTGL.tgl
species oyster 0.0005 0.00002 55.10 30 1.0 inner-outer.sol areas
species scallop 1.192300 0.152070 56.50 30 2.0 inner-outer.sol areas
```

Each line is headed by a keyword indicating the meaning of its fields. A complete description of the files' formats can be found in Annex 4.

Parallel Simulated Annealing

Simulated annealing is considered a good tool for complex nonlinear optimization problems (Kirkpatrick et al., 1983) but one of its major drawbacks is the slow convergence, more

apparent if it is desired an extensive exploitation of the search space. One way to improve its efficiency is the development of a parallel version of the algorithm.

Several parallel implementations of this algorithm exist and they are, inherently, problem dependent. In the work of Ram et al. (1996) the clustering algorithm and the genetic clustering algorithm were presented as distributed algorithms for simulated annealing, and the authors emphasize the evidence that a good initial solution results in a faster convergence.

The **clustering algorithm** technique starts n nodes of the network to run the simulated annealing algorithm: one of the nodes is the coordinator and distributes different initial solutions among the other nodes (worker nodes). After a fixed number of iterations, the nodes exchange their partial results with the coordinator to get the best one. The coordinator validates and saves the best solution, and all the nodes restart a new simulated annealing round with a solution based on the best. This process is repeated until a “near-optimal solution” is achieved or a predefined number of cycles is attained. The coordinator node is the responsible to distribute the initial solutions by the worker nodes, to filter the best among the best solutions so far and to stop the process.

The **genetic clustering algorithm** technique combines the simulated annealing with the genetic algorithms paradigm, adopting as motto that one good initial solution can result in a faster convergence – after the first iteration of the clustering algorithm technique, the coordinator chooses the best p solutions received, considers them as the initial population for the genetic algorithm, applies iteratively one operator on two parents chosen to generate two offspring that will replace the parents, and chooses n initial solutions to send to the worker nodes. Each worker node runs the simulated annealing algorithm. As in the prior technique, the cycle is repeated until the coordinator stops the process. The general idea of this algorithm is that genetic algorithm improves the whole population while simulated annealing aims to produce one best solution – if the genetic algorithm generates good initial solutions, then the process will converge faster.

In this work, as the coordinator and the worker agents have their own independent strategies and tactics to generate new solutions and achieve the optimal solution, the process is some kind of an improved clustering algorithm, taking advantages from the autonomy of each agent. The flexibility of the EcoSimNet framework allows each agent to control a different set of simulators, which can be seen as a collection of simulation islands or clusters (Figure 4-20). Simulators running in different computers have different simulation times, but there is no

need to synchronize them because the agent compares the results received against all results accumulated so far.

The basic algorithm for the coordination node is presented in Figure 4-25. The number of synchronization points determines the number of times worker nodes exchange partial results with the coordinator, and this number can improve the optimization process. The number of iterations between synchronization points is distributed uniformly by the worker agents; the coordinator node waits for agents synchronization, and each worker node controls the number of simulations with all simulators controlled by it, which means that if there is one very fast simulator and another very slow, it is expectable than the faster simulator perform more simulations than the slowest.

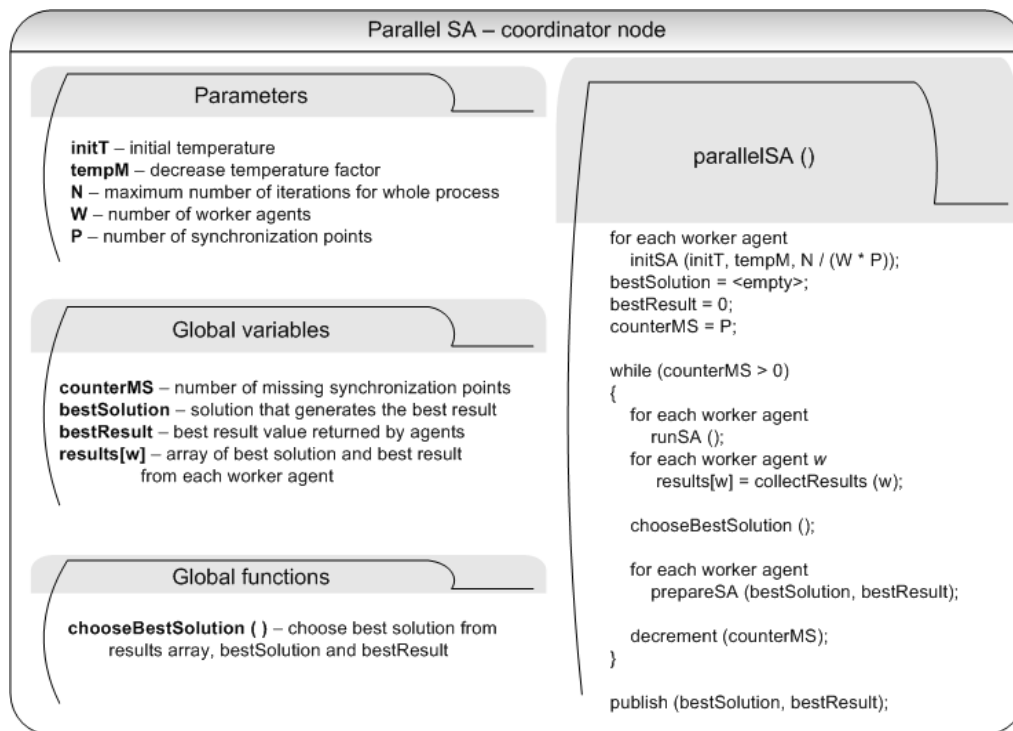


Figure 4-25: Parallel SA algorithm (coordinator mode)

Figure 4-26 presents the algorithm followed by the worker nodes.

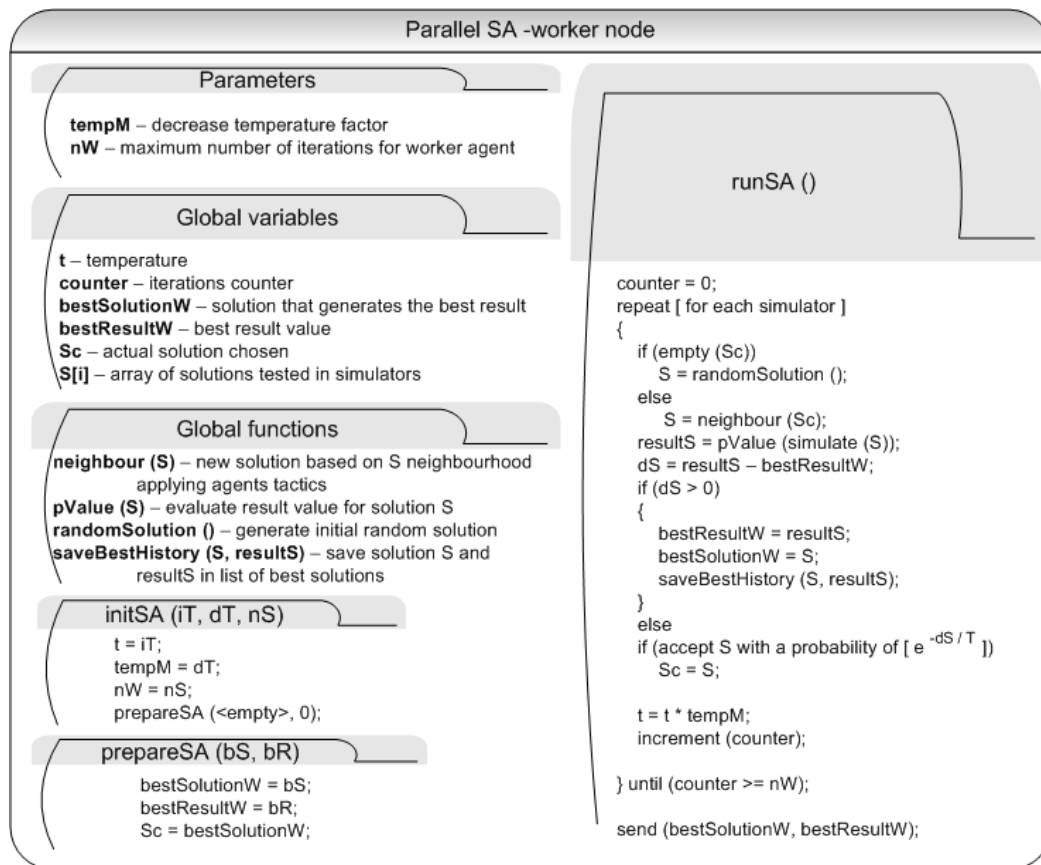


Figure 4-26: Paraller SA algorithm - worker node

Taking into account the time needed for each simulation and the time required to choose the best solution, often there are benefits if one agent accumulates the coordinator and worker roles.

4.5 Decision Support System

Ecological models are frequently developed for management purposes. The definition of a general structure to facilitate the construction of a decision support system (DSS) starts by the identification of the basic functionalities that it must answer. Every decision support system must be developed having in attention its final users. If the DSS application answers/explains questions like **how, why, for whom**, it will be surely a successful tool. Any DSS project must identify, since the beginning, the local actors and stakeholders and involve them in a formal basis in the project (Fedra, 2005).

As mentioned in the introduction of this chapter, the fundamental modules that a DSS must contain are: modelling and simulation, visualization and interaction with users, analysis and data management. The design of independent interacting modules is the proper way to follow, because it makes the system more scalable, flexible and robust, and facilitates the integration

of new modules or the replacement of some of them by others with more dedicated processing.

One important aspect of this approach is that it allows the construction of applications dedicated to the interface with the final users, leaving the hard work to the “inner” specialized applications not directly manipulated by the users, but communicating in a proper way with simulation and analysis tools. This grants management specialists to interact with an ecological system, configuring only high level objectives (management objectives) and waiting for the answer from the system, that translates the high level objectives to deeper scenarios configuration in model databases and model simulations, and returns the results, converted in a user-friendly format to the user by the interface application.

The previous sections of this document have described the modules of modelling and simulation, visualization and interaction with users, optimization with agents and extended interaction between applications. It is now time to focus on the analysis and data management modules.

Serment (2007) points out three essential contexts involving one environmental decision support systems (EDSS):

- **Regulation:** regular activity that leads to decisions whose consequences are immediate, short or very short term. Their scope is limited and includes a very specific area of an ecosystem.
- **Piloting:** general activity leading to medium-term planning and to manage the ecosystem in order to maintain its stability; several parts of the ecosystem may be covered by the decisions. The range is intermediate between those of strategic planning and regulation.
- **Strategic planning:** activity leading to major collective decisions whose scope is very broad and the consequences are long term. These decisions may, for example, involve the future of an entire ecosystem.

The EDSS may intervene in any of these levels of decision. Let's take an example related with the Natural Park of Peneda-Gerês (northeast of Braga district – north of Portugal):

- **Decision related to the regulation:** there is a peak of pollution in the area of Mata de Albergaria, and a quick decision must be taken. The decision could be to close access to motorized vehicles. The scope of these decisions is limited since they relate directly to the park entrances. The time allotted for decision making is very short.

- **Decision related to the piloting:** in the summer, peaks of pollution have been regularly observed in the mentioned subarea of the natural park. A decision that could be taken to anticipate this problem is to develop a network of shuttle buses and block access of private motorized vehicles to the park during critical periods, without preventing sightseeing or interfere with other activities. These decisions do not affect the whole ecosystem, and the consequences are not immediate. The balance of the consequences of this decision can be made only at the end of the season.
- **Decision related to strategic planning:** analysis of the air quality in recent years shows that quality is declining steadily. The decision is to reduce emissions of CO₂ within the park, by developing eco-tourism. This decision will lead to changes throughout the park with the development of walking trails, cycling, riding and the acquisition of a fleet of electric shuttles to facilitate the transport inside the park. Benefits or impairment of that decision can only be assessed over the long term.

The modules existent in an EDSS must be capable of interchange information and use data and functions from other modules. The decision maker must be able to configure one exploitation scenario and order the system to start the analysis – the simulator uses the model to produce results that can be visualized in the visualization application and saved in one persistent structure – comparing the data with other results from previous simulations or more complex analysis.

Although the system developed in this work intends to integrate all the facilities pointed to the fundamental modules of an EDSS, the persistent storage of the results enables the decision maker to use other applications to make the analysis of the generated data.

4.5.1 Construction and Evaluation

The methodology implemented for the EDSS is the Analytic Hierarchy Process (AHP) (Saaty, 1980). The AHP methodology was conceived to be used when multiple and conflicting objectives/criteria are present, and both qualitative and quantitative aspects of a decision need to be considered. In ecological systems, some of the usual objectives are contrasting (e.g. socio-economic interests versus environmental preservation). This leads to a field where, in general, the best management option is not the one that maximizes each single criterion, rather than one which achieves the most suitable trade off among the different criteria.

The software package used is built in MatLab® and integrated in a C++ DLL (Pereira et al., 2007). The MatLab® routine installed in the application was adapted from one developed at Siena University, Italy (Siena, 2006).

4.5.2 EDSS Manipulation – AHP Methodology

Setting up of a model scenario (the preparation of the configuration files, the definition of the ecosystem characteristics and the initial values of the variables) may be a complicated task. Therefore the usage of the models by non-modellers, involved directly on management issues, may be discouraged by the complexity mentioned. Even with the results obtained by the simulator for the different scenarios, it may be difficult to order them according to some quality criteria. That's why an EDSS can be a valuable tool to the decision makers.

The functionalities required by the EDSS were first included in the Development Agent (Figure 4-27), before their implementation as an autonomous application. The module was conceived to receive, as inputs, the results captured by the development agent in each scenario (that define the criteria used by the decision maker), to define the relative importance between each criterion and to rank the several scenarios using the relative importance criteria comparison definition.

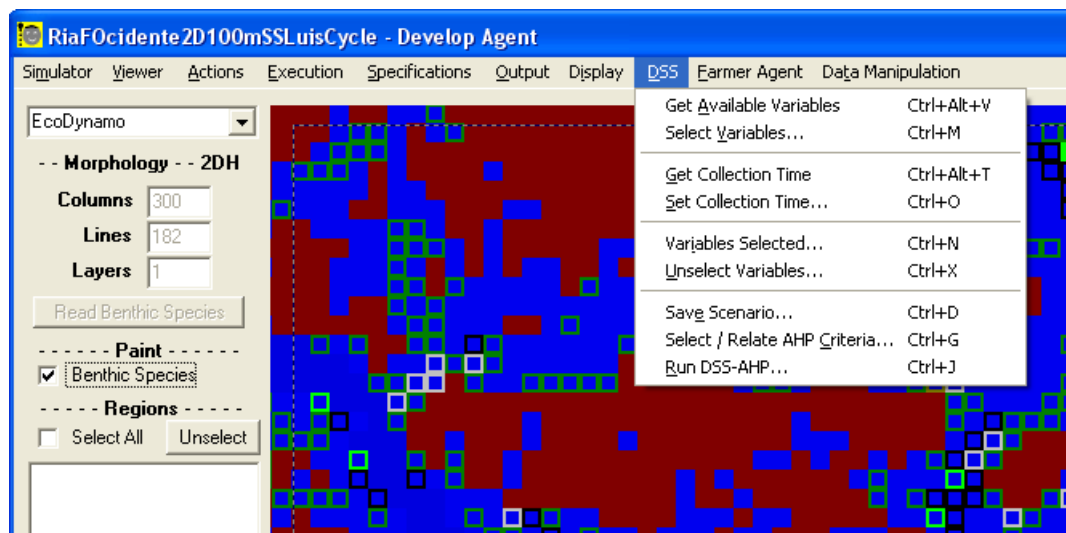


Figure 4-27: DSS options available in Development Agent

The manipulation of the EDSS is intuitive. The user can generate several scenarios to analyze ecosystem behaviour, adjust the initial values of the desired variables and save the configuration as an exploration scenario with the Development Agent. These operations can be made by any stakeholder or authority agent, generating the number of scenarios that correspond to hypothetical management realities or options. Before the simulation of the first

scenario, the user selects the variables that are directly related with the relevant criteria that indicates the contrasting objectives of the study. For each variable selected, the user defines what value will be considered in the analysis – total value, mean value, inverse of the total value or inverse of the mean value (see Figure 4-28). Each variable represents one criterion in analysis and corresponds to one column of the matrix of indicators (cf. 3.5.4). Each line of that matrix corresponds to one scenario, and will be filled with the values obtained during the simulation for each variable, also called indicator value.

During the simulations, and before the EDSS analysis, the decision maker must define the importance relationships between the variables (criteria). This phase will originate a file with the pair-wise comparison matrix values (PCM) for the selected criteria. The criteria comparison is introduced clicking small numerical toggle buttons that indicate the relative importance between each criteria pair (greater the value, greater the importance from the left column criterion to the right column criterion – Figure 4-29).

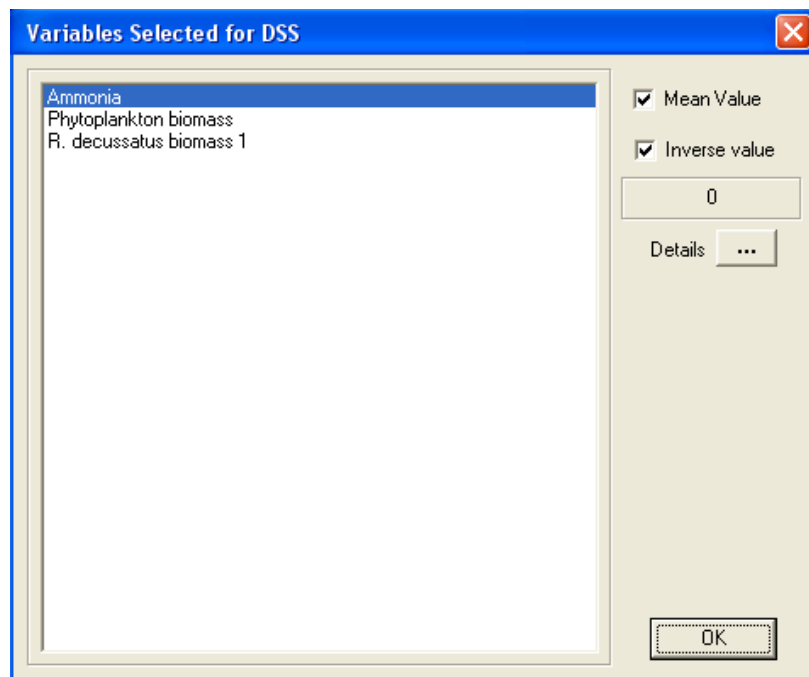


Figure 4-28: Window to choose the type of value considered for each variable analysis

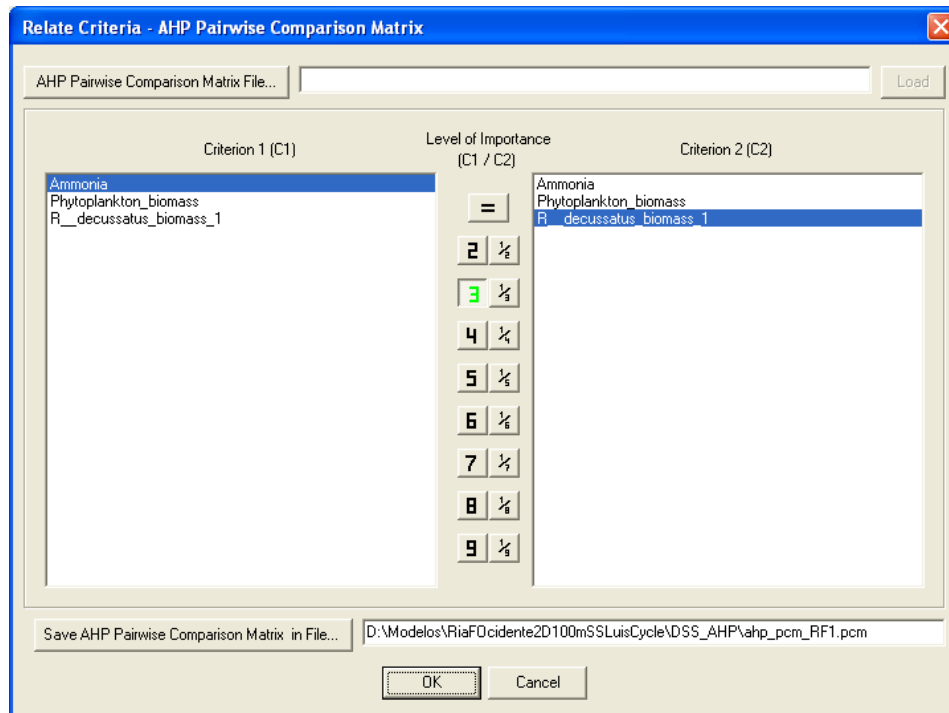


Figure 4-29: Pair-wise Comparison Matrix definition

For a good decision, the user must generate the largest possible number of scenarios (one scenario corresponds to a specific configuration of the simulated system) and save all the results in the same file, identifying each scenario with a consistent name (Figure 4-30).

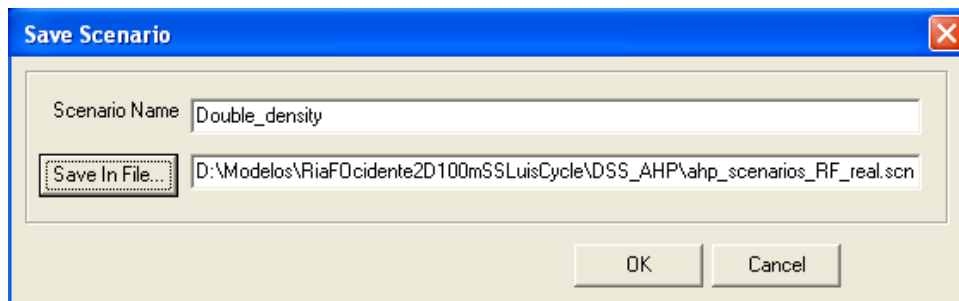


Figure 4-30: Saving the results with one scenario name

With the results of all the relevant scenarios simulated, saved in a file, and with the pair-wise comparison matrix saved in another file, the user can run the decision support application (Figure 4-31) and save the results in another file. The DSS application integrates the AHP software developed in MatLab®, in accordance with the steps set out in section 3.5.4, and shows the result in a MatLab® graphical window. All the files used by the DSS have alphanumeric characters with values separated by tabulations (TSV) to facilitate their integration with other management applications.

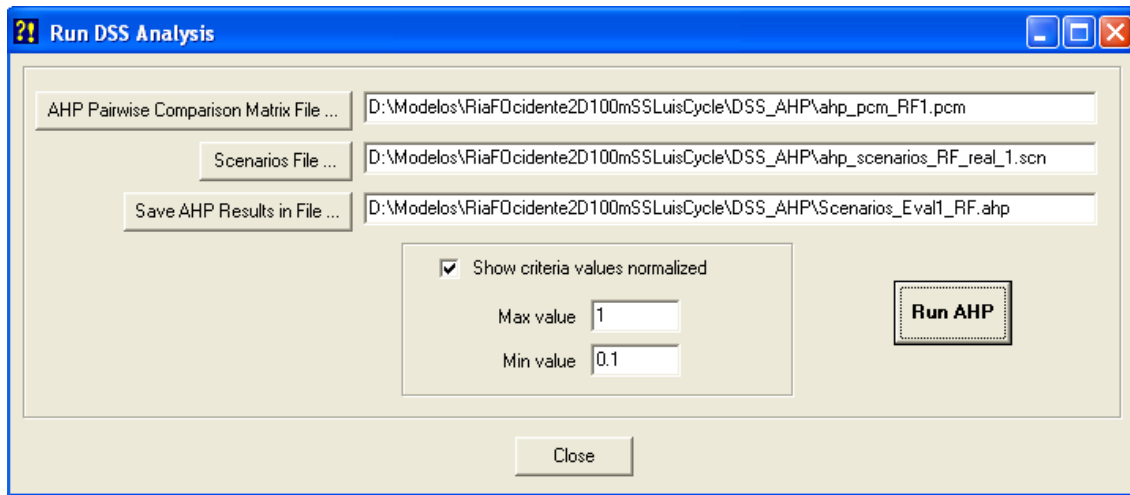


Figure 4-31: Select files for DSS analysis

The default filename extensions (“pcm” for Pair-wise Comparison Matrices files; “scn” for scenarios files; “ahp” for DSS results files) are used to quickly distinguish the type of data each file contains, but are not mandatory.

4.6 Simulations

After having gained confidence in the model outputs/predictions and in the final results generated by the complete framework, it was necessary to plan a scenario analysis to generate data to be used as support for decision making. The scenario analysis should be done in terms that the final outcome of the system could be a guide for coastal lagoon modellers and help the implementation of similar systems in other coastal ecosystems.

4.6.1 Sites Description

Ria Formosa

Ria Formosa is a shallow mesotidal lagoon, located along the eastern part of the south coast of Portugal, with an extension of 55km, a maximum width of 6 km and with a wet area of 105km² (Figure 4-32). The lagoon is protected from the ocean by a sandy barrier island interrupted by six inlets (S.Luís, Faro-Olhão, Armona, Fuzeta, Tavira and Cacela). Tidal movement and water exchanges between the lagoon and the ocean (through the inlets) determine the system evolution (Falcão et al., 2003).

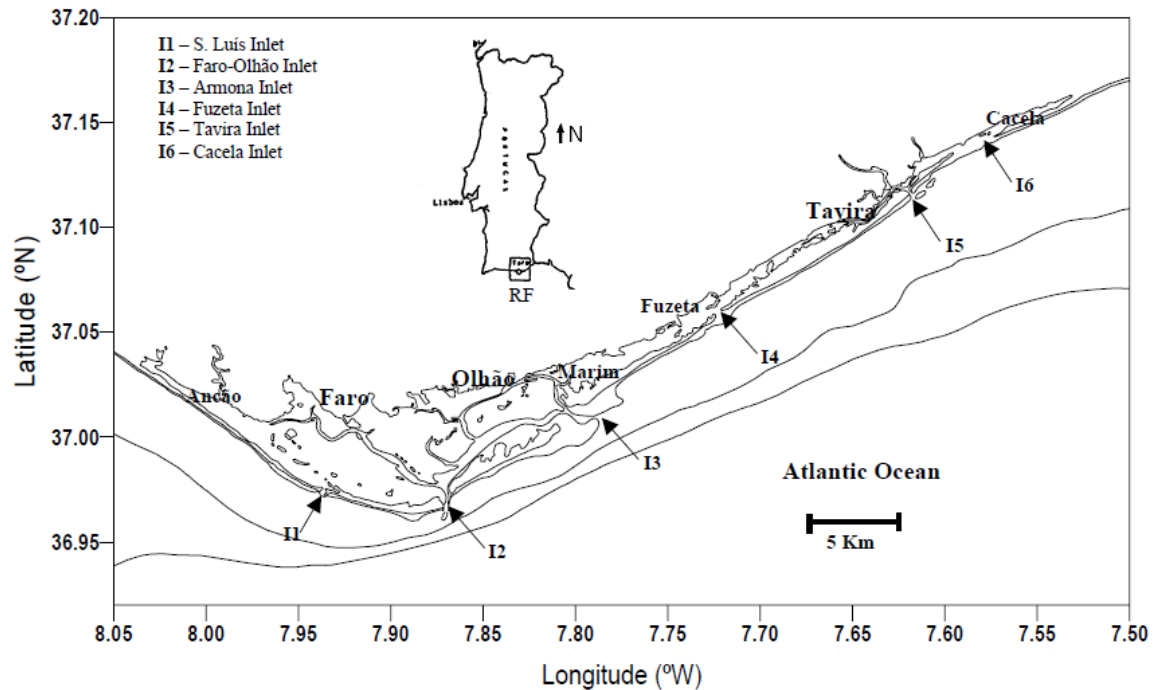


Figure 4-32: Geographic location of Ria Formosa and its inlets (Falcão et al., 2003)

The limit of Ria Formosa's watershed is the hydrographic basin of river Guadiana. Its rivers are originated mostly in the Caldeirão Mountain range and watercourses drain perpendicular to the South in the direction of the Atlantic Ocean. Most of the rivers are ephemeral with no or very little runoff between June and October. The most important watercourses in the Ria Formosa basin are river Gilão and streams Alportel, S. Lourenço, Zambujosa, Seco and Cacela (Figure 4-33). The Ria Formosa basin has an area of 864.26km², with a maximum altitude of 522m, and an average altitude of 112m (MAOT, 2000), draining flows into the coastal lagoon mostly through agricultural neighbouring land, where there has been some intensive use of fertilizers.

Ria Formosa is one of the largest Portuguese lagoons, and in 1987 some of its hinterland has been included in the Ria Formosa Natural Park (184km²) and accepted as a Natura 2000 network site. It is also one of the 28 Portuguese Ramsar sites (designated as Wetlands of International Importance: <http://www.ramsar.org/>), due to the recognition of its environmental relevance. Ria Formosa has large intertidal areas (corresponding roughly to 50% of the total area) where many conflicting uses coexist (fisheries, aquaculture, harbour activities, tourism and nature conservation). There are about 100 different land use classes, divided among six major groups: urban, agricultural, forest, rangeland and pastures, wetlands and water bodies. It is an ecosystem of great importance for it serves as a breeding site and development of a large number of marine species. Its shallow zones of sprawling tidal serve areas of cultivation of shellfish and fish, with very large touristic activities.



Figure 4-33: Ria Formosa watershed (Falcão et al., 2003)

The intertidal area, mostly covered by sand, muddy sand-flats and salt marshes, is exposed to the atmosphere for several hours, over each semi-diurnal tidal period. Tidal amplitude varies from 1 to 3.5m and the mean water depth is 3.5m (Falcão et al., 2003). The natural biogeochemical cycles are essentially regulated by tidal exchanges with the sea water and by the exchanges with the sediment interface. Management of this coastal ecosystem involves several institutions such as the Natural Park Authority, several municipalities and the Portuguese Navy. Within the scope of the already mentioned DITTY project (EC, 2003), several possible management scenarios were defined by the Natural Park Authority, and were evaluated from the environmental and economic point of view, by using an hydrologic model for the watershed and a coupled hydrodynamic-ecological model for the lagoon (Duarte et al., 2008).

The Soil and Water Assessment Tool (SWAT model) (Neitsch et al., 2002a; Arnold and Fohrer, 2005) has been applied to the watershed in order to simulate water discharges to Ria Formosa, providing forcing to a two-dimensional vertically integrated coupled physical-biogeochemical model. The model includes water column and sediment processes as well as their interactions and several biological sub-models (e.g. phytoplankton dynamics and bivalve growth) implemented with the object oriented modelling software EcoDynamo.

Aquaculture represents an important contribution to the Algarve's economy (approximately 10000 individuals are directly or indirectly involved in the shellfish production activity).

Shellfish culture exploit species with high economical value, like the clam (*Ruditapes decussatus*) and the oyster (*Crassostrea angulata*), that are cultivated in the intertidal areas of the lagoon. According to the operational plan for coastal planning, dated 1997, the annual production of clams reaches 5000 tons/year and the oysters production reaches 2000 tons/year (Falcão et al., 2003).

The model developed for Ria Formosa is 2D vertically integrated, coupled hydrodynamic–biogeochemical model. The wet-drying scheme referred in 4.3.1 requires a relatively high spatial and temporal resolution. In the present case, the cell resolution is 100m x 100m and the time step is 3s. A lower temporal resolution leads to numerical errors, in spite of the semi-implicit numerical scheme of the hydrodynamic model.

The original model grid had 470 columns by 282 lines (132540 cells). The analysis of general ebb and flood currents in Ria Formosa reported in Duarte et al. (2005a) and represented in Figure 4-34, showed that there was hardly any direct flow between its western and eastern sides, separated by a vertical line in the figure. Therefore, it was decided to split the model domain in two – a western and an eastern domain – using a higher resolution (50 m) in the latter.

This splitting procedure implied important gains in computing speed, by reducing grid size from the original 470 columns by 282 lines to 300 columns by 182 lines (54600 cells) for the western sub-domain, and 300 columns by 69 lines (20700 cells) for the eastern sub-domain. This also allowed the extension of the spatial resolution in the eastern side, where more detail is needed to simulate the narrow channels and the interface with River Gilão. It was on the western domain that the simulations have mainly focused.

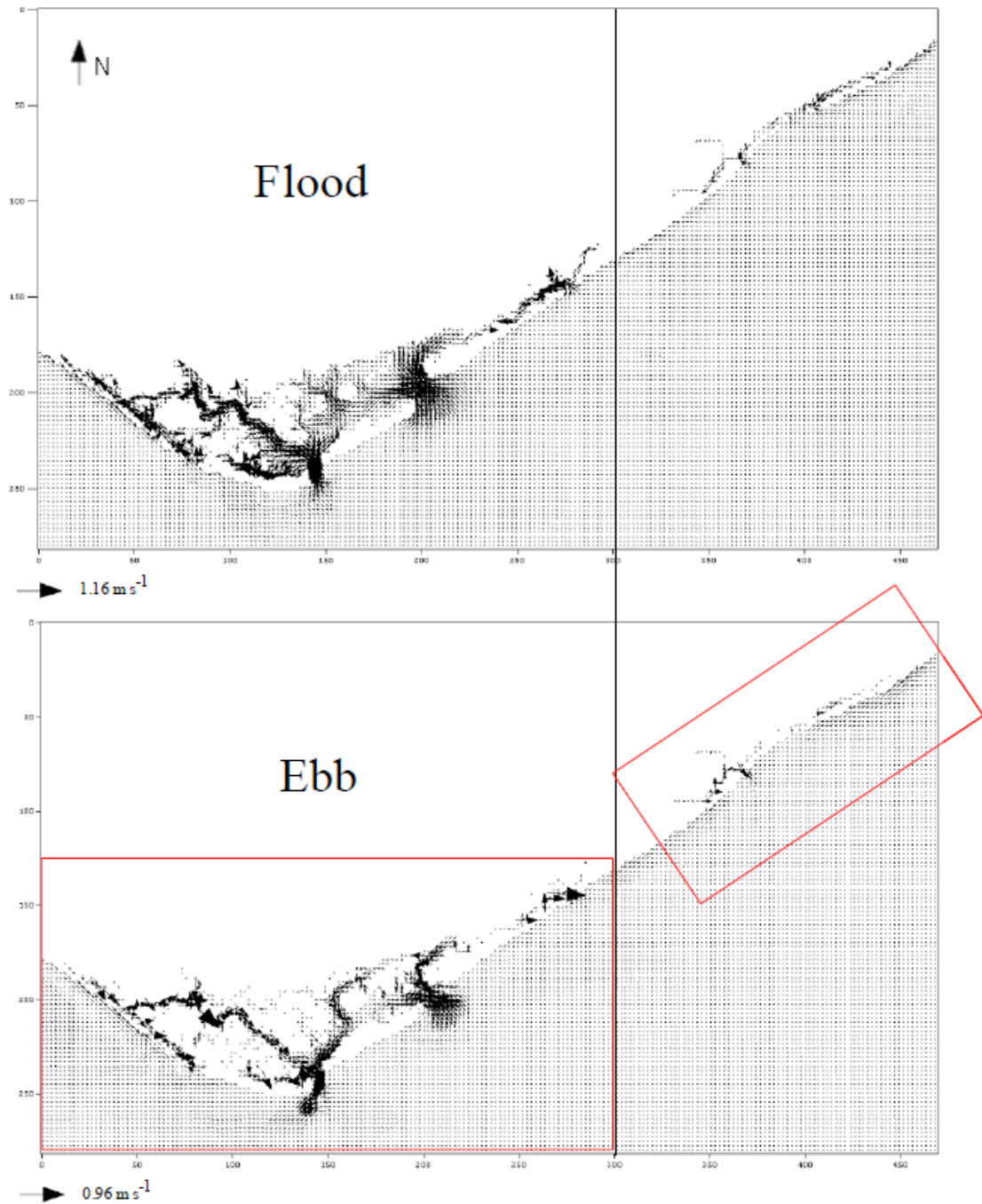


Figure 4-34: General circulation patterns during the flood and the ebb cycles. The vertical line separates the Western Ria from the Eastern Ria and the rectangles in red represent the two sub-domains considered (Duarte et al., 2005a)

Sungo Bay

Sungo Bay is located in Shandong Province of People's Republic of China. With an area of 180km² and depths varying gradually until approximately 20m at the sea boundary (Figure 4-35), it has been used for aquaculture for more than 20 years. The main cultivated species include kelp (*Laminaria japonica*), oyster (*Crassostrea gigas*) and scallop (*Chlamys farreri*).

Scallops and oysters are mostly contained in lantern nets and kelps are tied to ropes. A model was previously implemented, calibrated and validated with independent data sets, based on data collected in site over a period of 17 years (1983–2000), for use in estimating the environmental carrying capacity for the benthic species (Duarte et al., 2003). The whole bay has been used for mariculture and one of the most limiting factors for bivalve culture in Sungo Bay is scallop mortality. High summer mortalities at the beginning of XXI's century have led to changed aquaculture practices, including shifting the rearing periods, reorder of exploitation areas and mariculture scenarios, including areas of polyculture or monoculture.

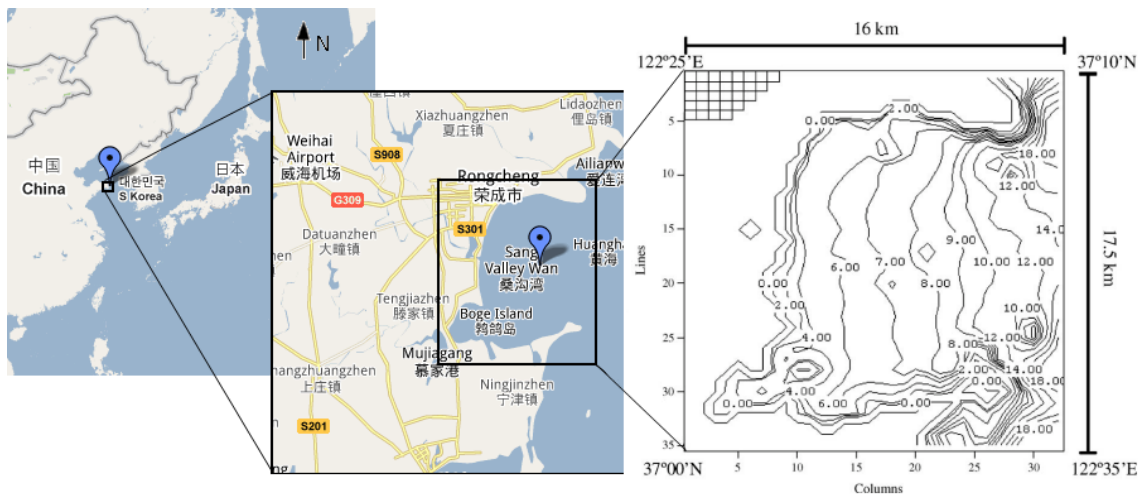


Figure 4-35: Sungo Bay location, with model domain, bathymetry and part of the model grid - adapted from (Duarte et al., 2003)

The model developed for Sungo Bay is a 2D vertically integrated, coupled hydrodynamic–biogeochemical model with 1120 cells (32 columns × 35 lines) and a spatial resolution of 500m. The model time step is 36s. However, due to the semi-implicit method used for time integrations, each time step is divided in two semi-time steps of 18s. At every semi-time step, one of the speed components is calculated semi-implicitly and the other explicitly, on an alternating sequence. The model has a land and an ocean boundary. It is forced by tidal height at the sea boundary, light intensity, air temperature, wind speed, cloud cover and boundary conditions for some of the simulated state variables. In (Duarte et al., 2003) are described all the equations used for the hydrodynamic and biogeochemical processes.

4.6.2 Scenarios Generation

Due to the relative speed of the Sungo Bay model, the experiments taken and the scenarios generation were focused on analysing management scenarios related to the distribution of a fixed number of cells with oyster or scallop species around the bay for aquaculture production

optimization. Oysters were seeded with a density of 55 individuals m^{-2} , 0.5g of shell weight and 0.02g of meat weight (corresponds to 4mm shell length, approximately). Scallops were seeded with a density of 56 individuals m^{-2} , 1.19g of shell weight and 0.15g of meat weight (corresponds, roughly, to 2.7cm shell length). These values try to approximate the values used in the real exploration of the bay as reported by Duarte et al. (2003).

In the Ria Formosa lagoon the scenarios generated intended to launch a decision process integrating three different/opposing criteria: aquaculture revenue, water quality and ecological sustainability.

4.7 Summary

The definition of a generic framework, to facilitate the construction of an environmental decision support system (EDSS) to manage ecosystems with anthropogenic exploitation, was focused in this chapter.

The fundamental modules that must exist in an EDSS (modelling and simulation, visualization and interaction with users, analysis and data management) were presented such as an implementation of each one of them. One concern maintained throughout the project, and referred to herein, was the independence of the modules, to make the system more scalable, flexible and robust. Of course, the interaction with users and the data management are transversal functionalities that are present, in a greater or lesser degree, in all the modules.

The developed system that originates this work was completely described, with emphasis on the EcoSimNet framework, infrastructure that incorporates the simulator application (EcoDynamo), the simulated classes, the agents developed and the environmental decision support system (with the AHP methodology), and the ECOLANG language that is used by all the components to exchange information between each other.

It was explained how the intelligent optimization algorithms were incorporated in the farmer agent and how the performance of the system can be increased by doing parallel simulations of different scenarios.

The results and indicators obtained with this integrated system can greatly help in decision making and support the management of aquatic ecosystems, since the decision-maker can introduce multiple criteria and conflicting objectives for analysis, can change the relative importance between criteria and can see, in a very short time, the graphical results of its options.

It is important to notice that the simulation system is being continuously improved in order to add more simulated classes and to include new ECOLANG messages, necessary to satisfy new requirements for the integration of new agents, until it reflects/integrates the interests of humans, with decision capacities, intervening over the ecological simulated system.

5 Results and Discussion

“It’s so much easier to suggest solutions when you don’t know too much about the problem.”

Malcolm Forbes

5.1 Introduction

In the precedent chapter, the architecture of the EcoSimNet system and the inter-relationships between the components were exposed, focusing on the most relevant aspects of the simulation system, a set of simulators surrounded by one multi-agent system and one environmental decision support system that was built on the top of the system. The implementation lines and options followed for each one of the different components were indicated.

In this chapter some results are presented to prove how the objectives of this work (cf. 1.2) were achieved for scenarios pointed out before (cf. 4.6.2), in order to validate the approach and the methodology followed.

5.2 Realistic Simulation of Ecosystems

The model of Ria Formosa lagoon (cf. 4.6.1) was implemented using EcoDynamo (cf. 4.3.2) and a set of objects from the library of the dynamic linkable objects (cf. 4.3.3), simulating the hydrodynamic, thermodynamic and biogeochemical processes. A detailed analysis of the

calibration and validation stages was reported in previous publications from Duarte et al. (2005a; 2007; 2008), and a subset of the results is presented here. Figure 5-1 shows the locations of four tide-gauge stations surveyed by the Portuguese Hydrographic Institute (Hidrográfico, 2001), with current velocity data collected over periods of several days, between January and March 2001 and used for model calibration of physical processes.

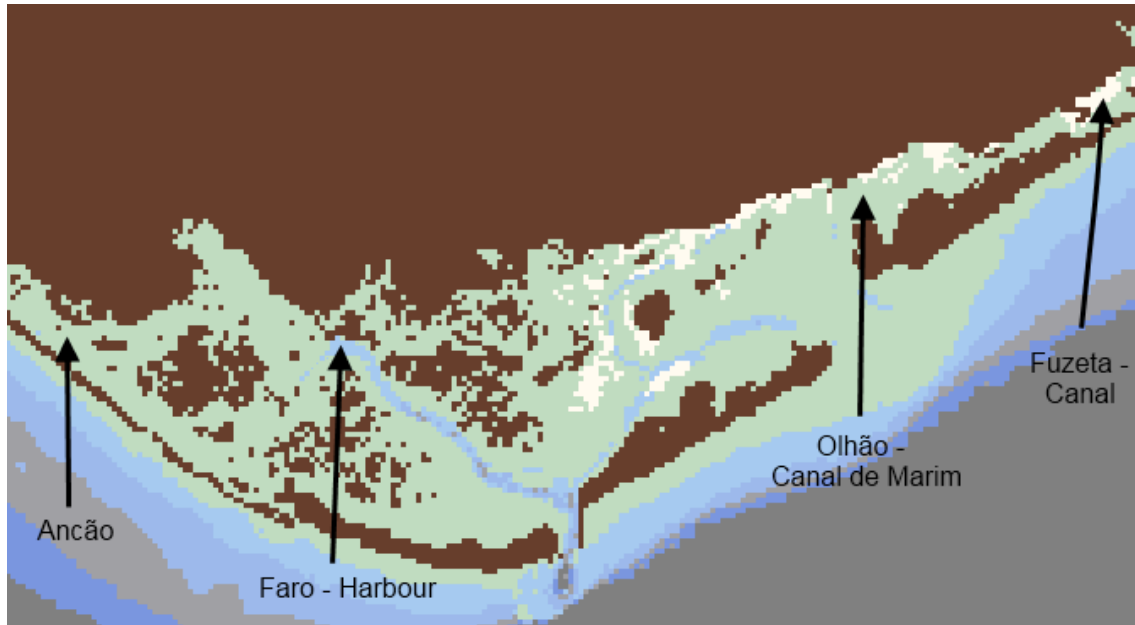


Figure 5-1: Ria Formosa lagoon with locations of tide-gauge stations (triangles) used for hydrodynamic calibration

Figures 5-2 and 5-3 show the measured and predicted current velocities in Faro-Harbour and Fuzeta-Canal (monitoring locations depicted in Figure 5-1). From the observation of the results, the visual fit between measurements and predictions is generally good.

Measured velocities range varied from nearly zero till values around 100 cm s^{-1} , with peak velocities occurring at the middle of the ebb and the middle of the flood.

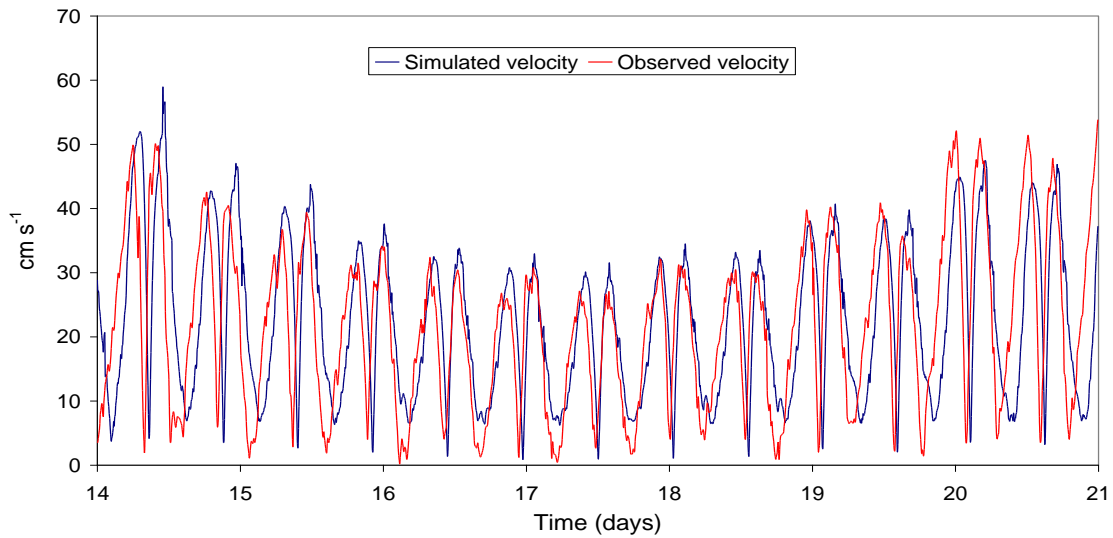


Figure 5-2: Predicted and measured velocities at Faro - Harbour

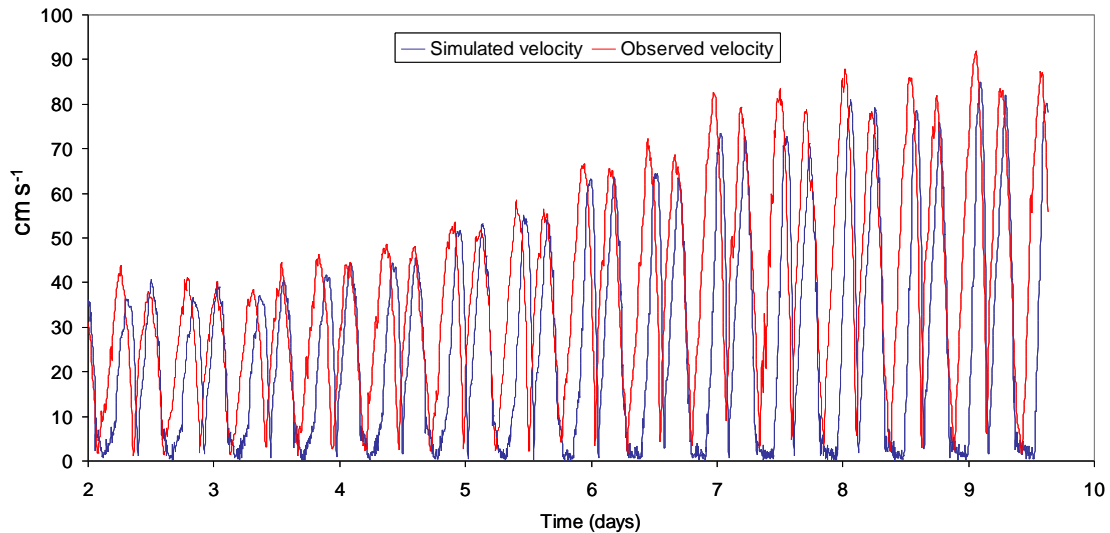


Figure 5-3: Predicted and measured velocities at Fuzeta - Canal

To validate the biogeochemical processes, and due to the slowness of simulations in Ria Formosa, the hydrodynamic part of the model was run first, the results saved in data series sequences with time-averaged results for flow and velocity variables and, later, the model was run with a larger time step, using the saved results to provide the hydrodynamic forcing for the biogeochemical simulations.

Figure 5-4 shows the locations of the water quality stations inside the lagoon, where nutrient data was measured (WQA, WQB and WQC). The data available, and used to test the model, was collected in 1992 (Falcão, 1996). Due to the fact that lagoon bathymetry used in the model is more recent (Hidrográfico, 2001), the comparison between observed and predicted data should be carried out with caution.

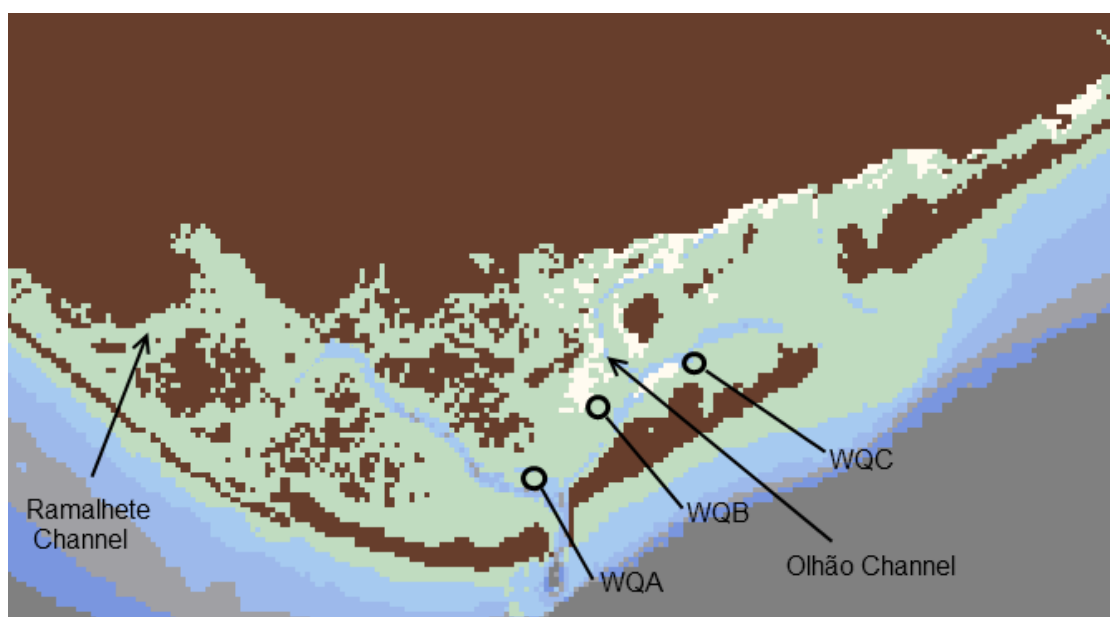


Figure 5-4: Ria Formosa lagoon with locations of water quality stations (circles) - WQA, WQB, WQC - used for biogeochemical calibration

Figures 5-5 and 5-8 show simulated values for ammonia, nitrate, phosphate and water temperature at Ramalhete and Olhão channels and the correspondent observed values at station WQB during ebb and flood. Also the assumed calibrated values are superimposed.

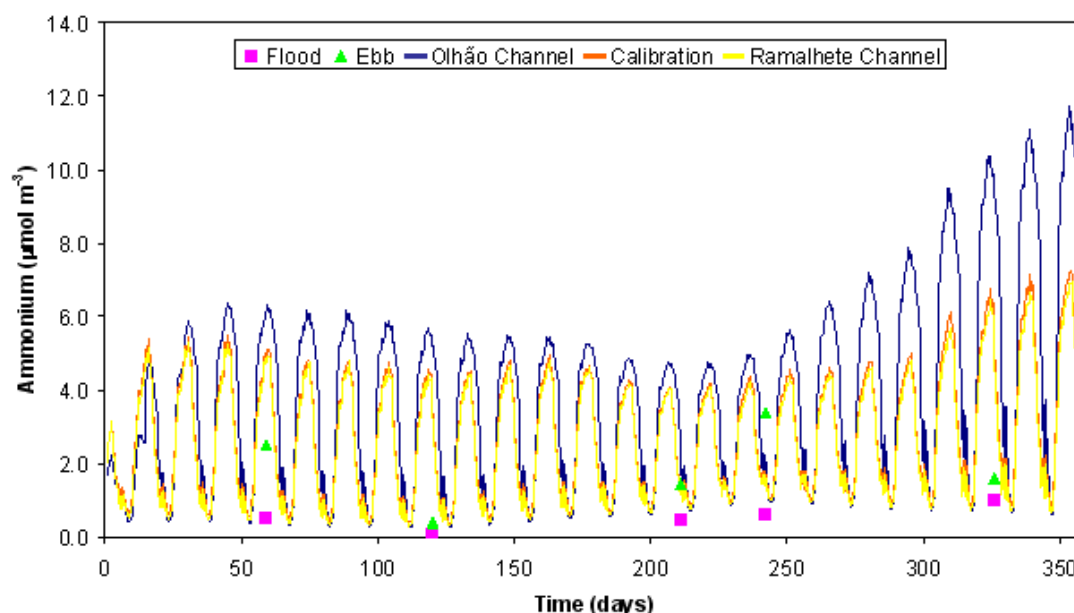


Figure 5-5: Simulated and observed ammonia at station WQB

Observations were made during the ebb and during the flood for each sampling instant. The small number of observations prevents any powerful statistical test to quantify the model performance however, in most situations, the ranges predicted by the model are within those

observed, with a poorer estimation for ammonia and nitrate, slightly overestimated by the model.

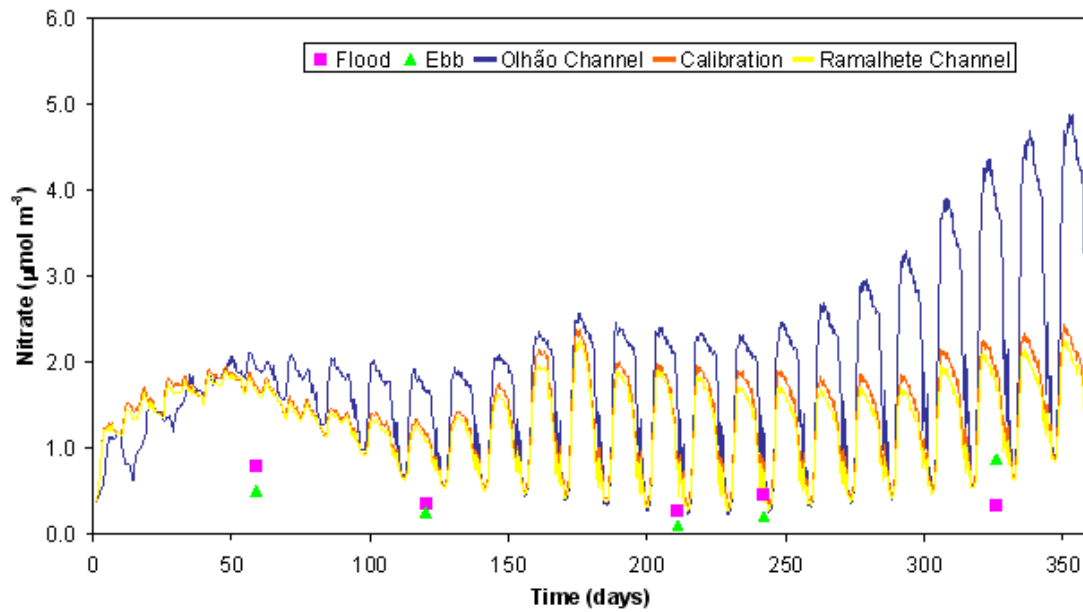


Figure 5-6: Simulated and observed nitrate at station WQB

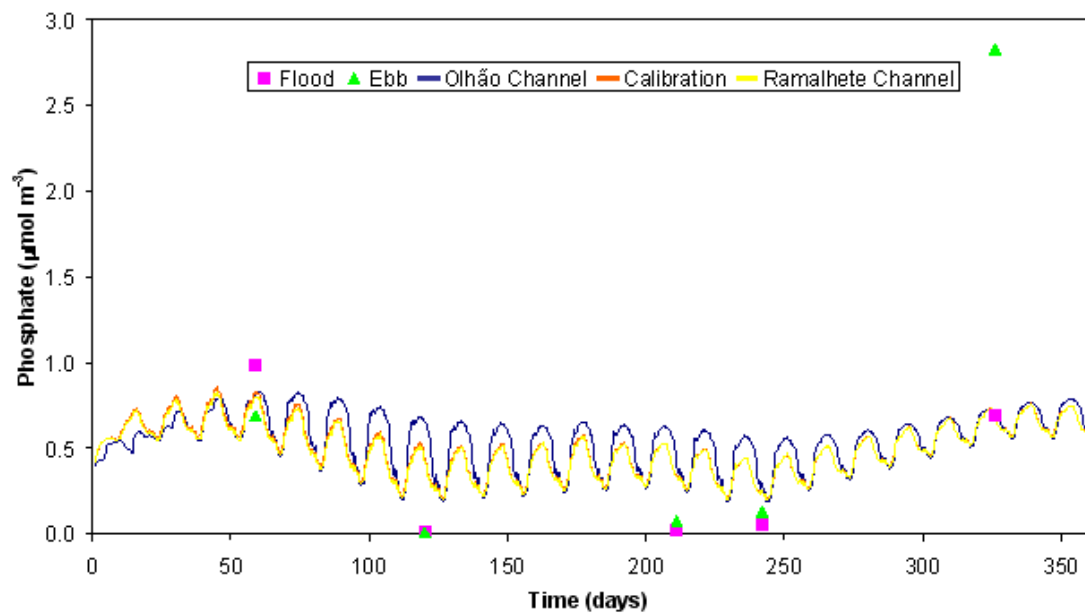


Figure 5-7: Simulated and observed phosphate at station WQB

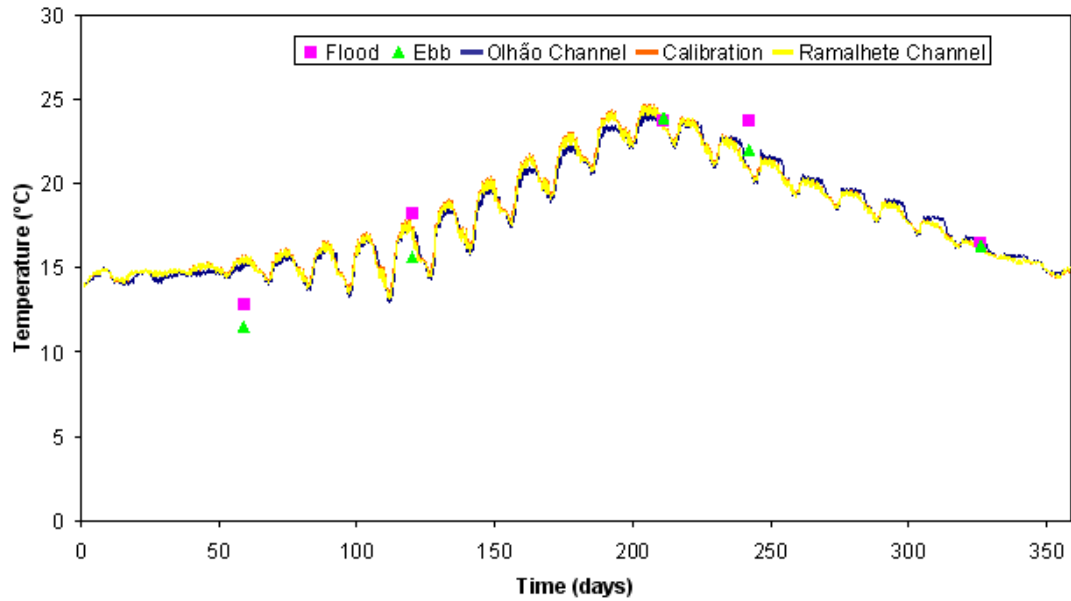


Figure 5-8: Simulated and observed water temperature at station WQB

The model is also able to predict clams individual weight within the range of values observed in Ria Formosa (Figure 5-9). Cultivated clams grow from an initial weight of c.a. 0.12 g to a weight ranging between 0.2 and 0.4 g of meat dry weight, over a period of c.a. half a year. The observed data was taken from Falcão et al. (2000).

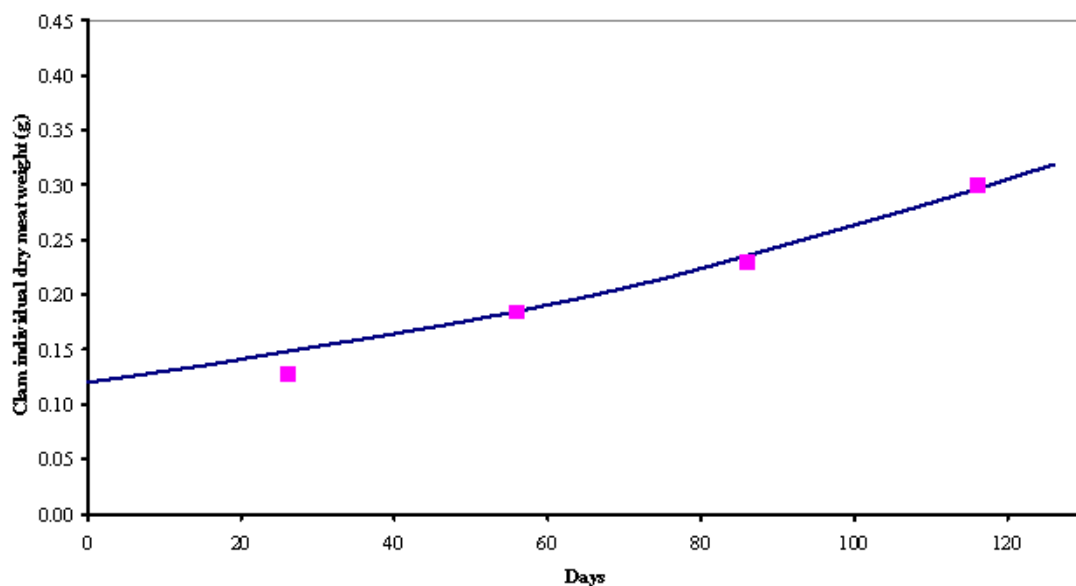


Figure 5-9: Predicted (line) and mean observed (squares) clams weight at two different points

The model of Sungo Bay (cf. 4.6.1) was implemented based on a previous model built with the EcoWin (cf. 2.4.2) tool by Duarte et al. (2003). The migration to EcoDynamo was confirmed

and validated comparing the results presented in the former study with the detailed results published by Duarte et al. (2005b).

5.3 Flexible Simulation

5.3.1 Integrating Objects in Simulations

One of the requirements pointed out on the development of the DLLs code (cf. 4.3.3) was the possibility to dynamically include or exclude objects from simulations in order to enable flexible ecological simulations. Next figure shows the overlapped results of oysters growth in Sungo Bay from two simulations with phytoplankton object included and excluded. Graph represents the oyster growth during 112 days. It is evident that oysters grow only in the presence of the main food source.

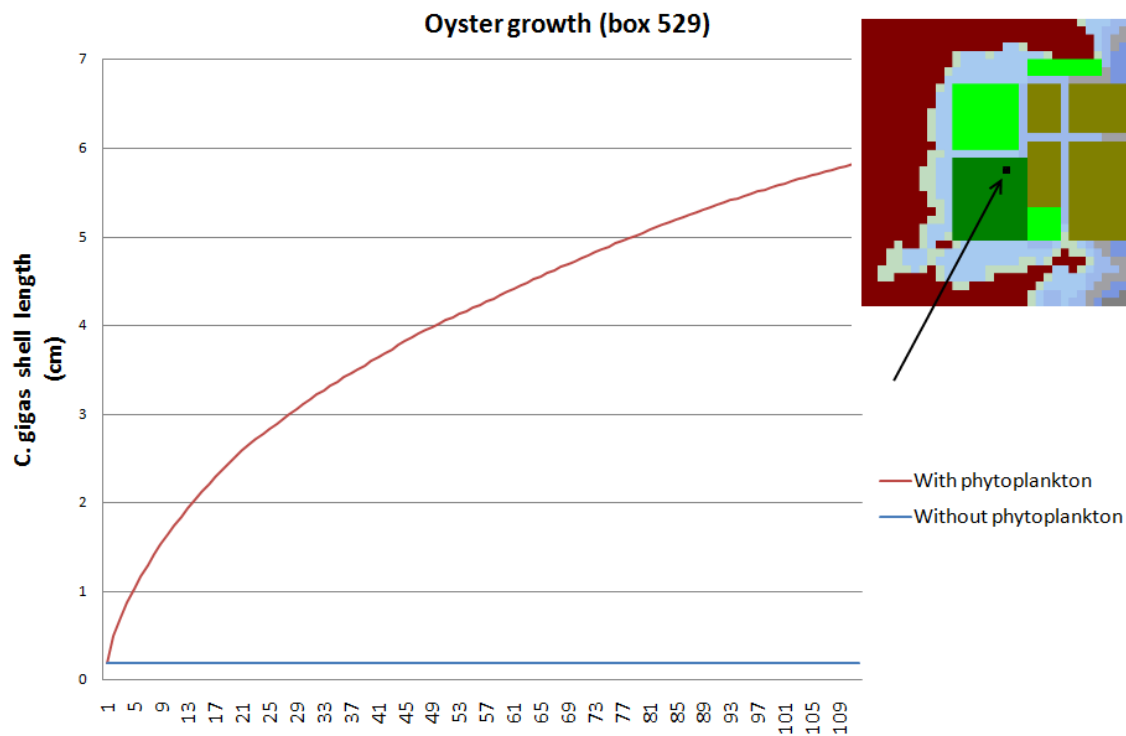


Figure 5-10: Oysters growth in one location of Sungo Bay with (red line) and without (blue line) phytoplankton object included

Several advantages exist from the possibility of select / unselect objects for simulation. One is related with the evaluation of the relative importance of each object in the model predictions. By selecting / unselecting objects, it can be seen whether the model responds as expected and whether it makes sense, compared to what is known, a priori, from the relationships between different variables and processes. This feature is crucial during the model calibration, to adjust

the parameters of each class, by selecting only the relevant objects that influence the class under calibration.

5.3.2 Integrating DLLs with legated platforms

Another requirement pointed out on the development of the DLLs code (cf. 4.3.3) was the possibility of the code be linked with applications made in different programming languages, in order to reutilize developed processes and embed EcoDynamo objects in legated systems. One of the tests made linked the Coherens code (Luyten et al., 1999) with the object that calculates the luminous intensity in any geographic location, day and hour, having as parameters the latitude and the cloudiness (class Light from Light.dll). The idea is Coherens use the values of light intensity generated by Light object, for its internal calculations of photosynthesis.

The following figures show the values sent by Coherens to the Light object, in Ria Formosa lagoon, at different times (10, 12 and 18 hours) of the cloudless Julian day 182 – July 1, and the answers returned.

```

C:\Windows\system32\cmd.exe
Latitude?
37
Day of the year?
182
Time of the day (hours)?
10
Cloud Cover?
0
Light Extinction Coefficient?
0.5
Depth?
1.5
Total surface irradiance (W/m2): 877.038452
PAR surface irradiance (W/m2): 368.35614
Daylight hours (h): 14.5023966
Depth integrated irradiance (W/m2): 617.006409
Depth integrated PAR irradiance (W/m2): 259.1427
Noon surface PAR (W/m2): 416.291901
Photic depth (m): 1.5
Sub-surface irradiance (W/m2): 414.28363
Sub-surface PAR irradiance (W/m2): 173.999115

```

Figure 5-11: Ria Formosa - value of light variables at 10 a.m. of July 1

Light object calculates daylight hours and noon surface photosynthetically active radiation (PAR) for each day, and total surface irradiance, PAR surface irradiance, depth integrated irradiance, depth integrated PAR irradiance, sub-surface irradiance and sub-surface PAR irradiance for each time of the day and water depth.

In addition to the explained in section 4.3.3 , detailed information about the integration of objects developed in the Library of Dynamic Linkable Objects into applications written in other

programming languages can be found in Pereira et al. (2006), where special emphasis is given to integration with Coherens (Luyten et al., 1999).

```

C:\Windows\system32\cmd.exe
Latitude?
37
Day of the year?
182
Time of the day (hours)?
12
Cloud Cover?
0
Light Extinction Coefficient?
0.5
Depth?
1.5
Total surface irradiance (W/m2): 991.171204
PAR surface irradiance (W/m2): 416.291901
Daylight hours (h): 14.5023966
Depth integrated irradiance (W/m2): 697.30011
Depth integrated PAR irradiance (W/m2): 292.866028
Noon surface PAR (W/m2): 416.291901
Photoc depth (m): 1.5
Sub-surface irradiance (W/m2): 468.196106
Sub-surface PAR irradiance (W/m2): 196.642365

```

Figure 5-12: Ria Formosa - value of light variables at noon of July 1

```

C:\Windows\system32\cmd.exe
Latitude?
37
Day of the year?
182
Time of the day (hours)?
18
Cloud Cover?
0
Light Extinction Coefficient?
0.5
Depth?
1.5
Total surface irradiance (W/m2): 174.663544
PAR surface irradiance (W/m2): 73.3586884
Daylight hours (h): 14.5023966
Depth integrated irradiance (W/m2): 122.877769
Depth integrated PAR irradiance (W/m2): 51.6086655
Noon surface PAR (W/m2): 416.291901
Photoc depth (m): 1.5
Sub-surface irradiance (W/m2): 82.5052185
Sub-surface PAR irradiance (W/m2): 34.6521912

```

Figure 5-13: Ria Formosa - value of light variables at 6 p.m. of July 1

5.4 Culture Optimization

The experiments related with the optimization processes were controlled by the Farmer Agent (FA) described in section 4.4.5, working with the different configurations admissible by the EcoSimNet framework. FA processing follows the sequence diagrams presented in Figure 4-18 and Figure 4-19.

5.4.1 Spatial Optimization

In the spatial optimization of aquaculture leases the FA has to discover the best combination of locations to seed and harvest bivalve species. Given a lagoon model loaded in the simulator, FA interacts with EcoDynamo in order to run a series of simulations with different scenarios seeking to find the one with the maximum bivalve production (weight harvested) constrained by the fixed number of areas to seed in a larger admissible area. Due to the inability to test the entire solution space, the optimization process will test a considered number of solutions in order to give more reliable results. The time needed to simulate the complete cycle of the bivalve's grow (about a year and a half) is another constraint that FA must surpass.

The experiments were carried out using the Sungo Bay model (described in section 4.6.1) derived from its simplicity, the velocity of the model simulations and the existence of real data that can be used to compare the results.

First experimental set

The first study intended to find the five cells (square areas of 500m by 500m) that together had the best production of oysters, within a wider rectangular area of 88 admissible cells with the same individual area – corresponding to more than 39 millions of possible combinations in an area of 4000 m by 5500 m. The rectangular area over study and one possible solution (combination of five areas within the mentioned region) are represented in Figure 5-14.

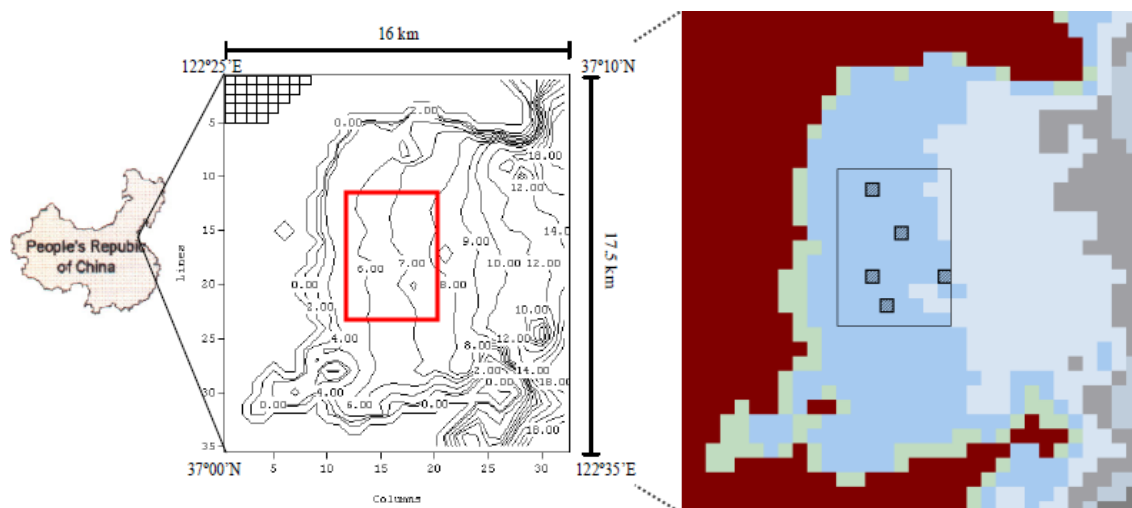


Figure 5-14: Experimental area and one admissible solution

Simulated oysters feed on suspended particles (detritus and phytoplankton cells) and cause significant local depletion of those food items. Due to the realistic characteristics of the ecological simulation, the existence of oysters in one location will affect the growth of oysters

in the neighbourhood, and placing many regions together could negatively affect the potential yield for them. Factors like tidal flux, quantity and quality of suspended food particles and water quality, influence oysters' feeding and substantially increase the complexity of the problem.

Simulating a year and a half with the Sungo Bay model, to perform a complete bivalve cultivation cycle realistic simulation, corresponds to run more than 1 576 800 simulation steps of 30 seconds (about 547 days). In a computer with a relative powerful performance (Intel® Core™2 Quad CPU Q9300 @2.50GHz and 8.00GB RAM, with a 64-bit Operating System) the complete simulation cycle takes about 8 hours to run, which turns virtually impossible to support a management decision based on the results of all possible combinations - exploitation scenarios. Considering the value of temperature for simulated annealing algorithm varying from 1.0 to 0.001, with a descent rate of 2% between iterations, each experimental round contains, at least, 341 different scenarios, which takes about 2728 hours of simulations, running more than 113 days.

The time taken and the heavy processor power required, implied trying to find out if shorter simulations could suggest clues to what are the best scenarios to test. The first experiments used only simulations of 1000 steps, which corresponds approximately to 8 hours and 20 minutes of real-time – each simulation took about 20 seconds to complete and one experimental round was completed in less than 2 hours. Reported results of these experiments showed that the approach had many strengths, namely identifying different tactics that could lead more quickly to good solutions (Cruz et al., 2007) and verifying that although the initial solution is always random all the tactics converge to similar solutions, creating an idea of a family of good locations as the most profitable zone for oysters farming, even with a simulation time very far from reality. The results obtained (Table 5-1 and Figure 5-16) served to test the hybridization of the algorithms explained in section 4.4.5 with the concern of avoiding the simulation of repeated solutions. Maintaining the FarmerTabu algorithm active during all the process can accelerate the convergence to a good solution - the repeated insistence on a solution can be a sign that it is a good solution and a slight mutation on it can lead to a new better combination.

Using the FarmerGA algorithm, during the course of the optimization process, improves the construction of good solutions because it crosses over the best solutions obtained so far to derive new combinations. Figure 5-15 shows the graphical evolution of the production indicators through iterations in one experiment, being clearly visible the improvements in the

solutions when FarmerGA actuates. The red line (MaxValue) shows the best value found so far and the blue line (IterValue) indicates the value obtained by the current solution.

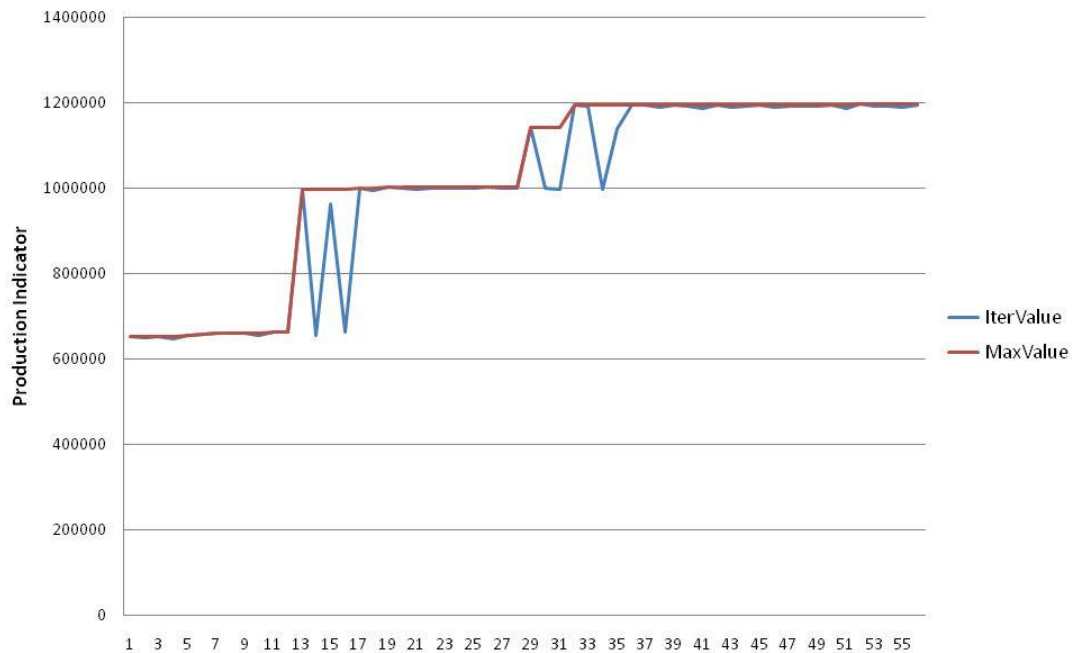


Figure 5-15: Optimization results evolution during an experimental round, with visible gains when FarmerGa actuates – experiment DM::IO.60_sca (cf. 5.4.1 – Fourth experimental set))

The appliance of the FarmerRL algorithm can introduce some good turbulence in the generation of the solutions, essentially in the early stages of the process, and spread the combinations all over the solutions domain. The great number of variants that could be applied in FarmerRL makes this algorithm very appealing to be used in different configurations throughout the optimization process.

The values obtained in each simulation are directly related with the harvested yields for oysters, but must be viewed as an indication and not as the real weights harvested for commercialization, because the FA sees the production as the whole quantity existent in the seeded areas while, in real harvest, only individuals with a shell length greater than a minimum size have commercial value and are captured, making the actual amount less than that provided by FA.

For the first experimental set, several rounds generated 1876 different solutions taking advantage of the speed for simulating only 1000 steps. As expected, the values obtained do not diverge greatly, as can be viewed in Table 5-1, but the best solutions reveal a trend to occupy the east side of the delimited area to explore, near the open sea (Figure 5-16). The

diversity required for this type of algorithms is ensured by the randomness of the initial solutions.

Table 5-1: Relevant results (oysters: 5 cells – 1000 steps)

| Rank | Production Indicator | Cells |
|------|----------------------|--------------------|
| 1 | 181.864 | 63, 71, 79, 86, 87 |
| 2 | 181.863 | 55, 71, 79, 86, 87 |
| 3 | 181.861 | 71, 78, 79, 86, 87 |
| 4 | 181.860 | 70, 71, 79, 86, 87 |
| ... | ... | ... |
| 1876 | 181.761 | 4, 16, 17, 34, 74 |

The cells are numbered from 0 to 87 – starting from the bottom left of the available area and increasing from left to right and from bottom to upper bounds.

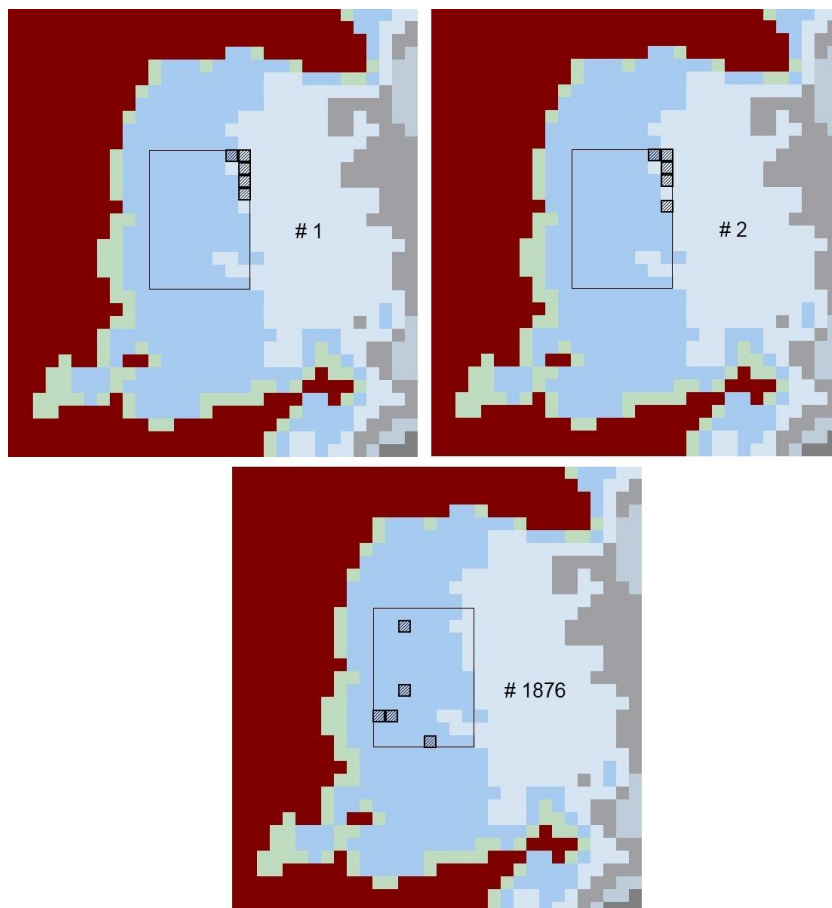


Figure 5-16: Graphical representation of the 2 best (top) and the worst (bottom) solutions (oysters: 5 cells, 1000 steps – 8h20m)

Second experimental set

The existence of a trend for the best results increased the expectation of achieving good results with larger simulations and the second experimental set took advantage from the existence of four simulators in parallel - the simulation steps were increased to cover one month of real time oysters growth (from 1000 to 86 400 steps). The number of areas to seed and the exploitation area were maintained to verify if the extrapolation assumed in the first experiments as the good “farming areas” still remains.

Each simulation for the second experimental set took a little more than 20 minutes, which gives about one day and six hours for each round with 341 iterations and four simulators. 1039 different combinations were tested in several rounds within this configuration and several interesting observations were triggered by the results obtained (extreme results are presented in Table 5-2, Figure 5-18 and Figure 5-18):

- The production indicator grew more than 86.4 times (growth of the simulation time) indicating the non-linear growth of the oysters within the simulation period;
- The difference between the maximum and the minimum production indicators is 15.2% of the minimum;
- It is not evident a trend for the best solutions, that are spread over the domain.

Table 5-2: Relevant results (oysters: 5 cells – 86 400 steps)

| Rank | Production Indicator | Initial P.I. | Growth Rate (%) | Cells |
|------|----------------------|--------------|-----------------|--------------------|
| 1 | 19 923 | 48.2 | 413.3 | 31, 45, 47, 70, 78 |
| 2 | 19 909 | | 413.0 | 37, 38, 45, 69, 78 |
| 3 | 19 880 | | 412.4 | 38, 53, 54, 61, 87 |
| 4 | 19 877 | | 412.4 | 30, 46, 53, 63, 87 |
| ... | ... | | | ... |
| 1038 | 17 386 | | 360.7 | 83, 84, 85, 86, 87 |
| 1039 | 17 299 | | 358.9 | 2, 7, 16, 41, 74 |

A possible explanation for the inexistence of a trend for the best solutions can derive from the fact that oysters growth and production are determined by the abundance of organic substances and phytoplankton. The ecosystem in question is subjected to tides that influence how the main food sources are transported along the bay and eight hours (time of the first experiment) is not enough to simulate properly the mixing processes for the food variability that affects oysters growth - when oyster biomass increases, also increases the demand for

food and local depletion of food – and, another compelling reason, eight hours of simulation is insignificant when compared to the water residence time in the bay: between one and 19 days (Duarte et al., 2003).

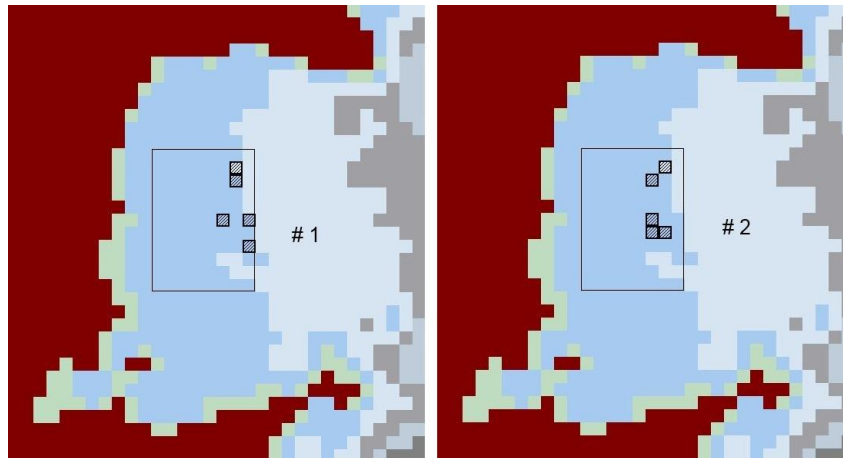


Figure 5-17: Graphical representation of the best solutions (oysters: 5 cells, 86 400 steps – 1 month)

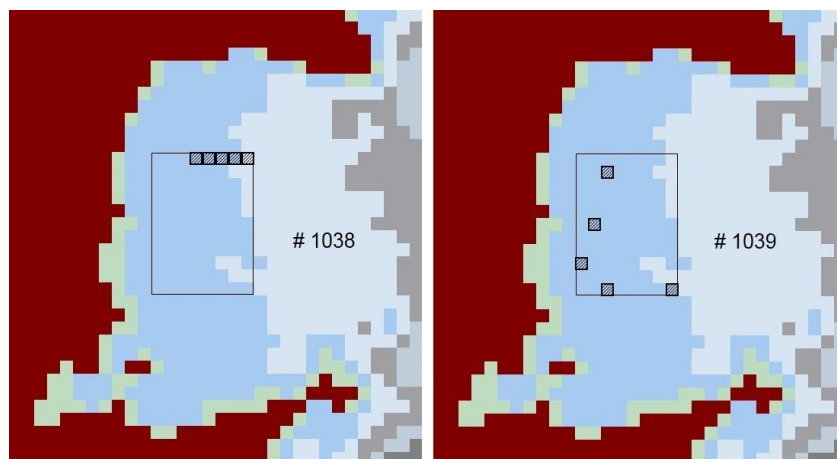


Figure 5-18: Graphical representation of the worst solutions (oysters: 5 cells, 86 400 steps – 1 month)

Third experimental set

The results of the second experimental set reinforced the idea that only more realistic scenarios and longer simulations could point directions to good locations with feasible results. The third experimental set was designed to seed 30 areas within the same admissible area for exploitation of oyster culture, to verify if a trend is more obvious when the ratio between occupied area versus exploitation area increases, maintaining a one month simulations of real time.

The increase of the occupied area, from 5.7% to 34.1% of the exploitation area, didn't affect the duration of the simulations for the third experimental set, since they are dependant from the number of steps and the number of classes being simulated, and those values remain the same. The results obtained show that the growth rate of the production indicator for the best solution is less than one half of the obtained for the best solution of the second experimental set, which reveals that the neighbouring of the seeded areas influence the local food depletion leading to slower oysters growth as the oysters concentration increases. Experiments done, generated 1031 different combinations with the relevant results presented in Table 5-3, in Figure 5-19 and in Figure 4-8.

Table 5-3: Relevant results (oysters: 30 cells – 86 400 steps)

| Rank | Production Indicator | Initial P.I. | Growth Rate (%) | Cells |
|------|----------------------|--------------|-----------------|----------------------------|
| 1 | 57 757 | 289 | 199.9 | Different sets of 30 cells |
| 2 | 57 721 | | 199.7 | |
| 3 | 57 707 | | 199.7 | |
| 4 | 57 660 | | 199.5 | |
| ... | ... | | | |
| 1030 | 55 027 | | 190.4 | |
| 1031 | 54 974 | | 190.2 | |

The best combinations show a tendency to concentrate the oysters in the northeast side of the exploitation area, which disagrees with the previous experiments, perhaps because those areas are near the open sea, where the water exchanges are more intense and food tends to be more abundant.

As a curiosity, the difference between the maximum and the minimum values for the production indicator is about 5.0% of the minimum. The two best combinations share 18 cells and the five best solutions share 12 cells, all located in the north-eastern border of the exploration area, while the best and the worst result share 7 cells. The initial production indicator, at the moment of seeding the bivalves, was 289, which indicates a gain of about 200 times the initial value.

To generate results with more management possibilities new experimental sets were designed: expand areas of exploitation and diversify crops of bivalves with, at least, one more species.

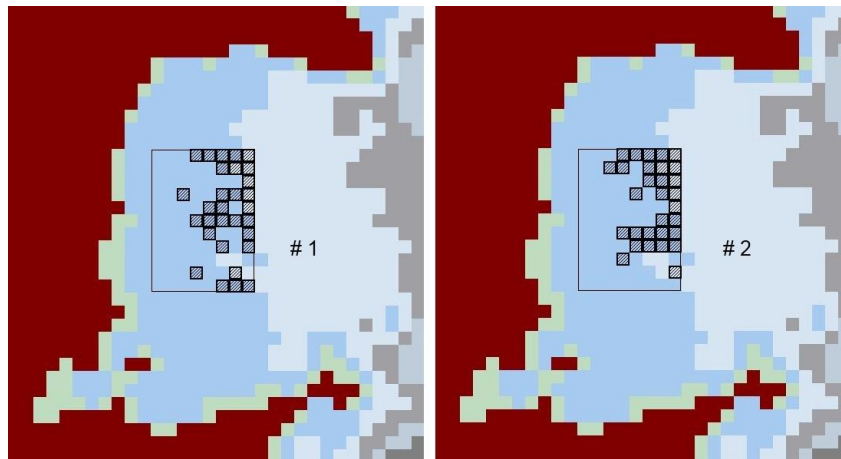


Figure 5-19: Graphical representation of the best solutions (oysters: 30 cells, 86 400 steps – 1 month)

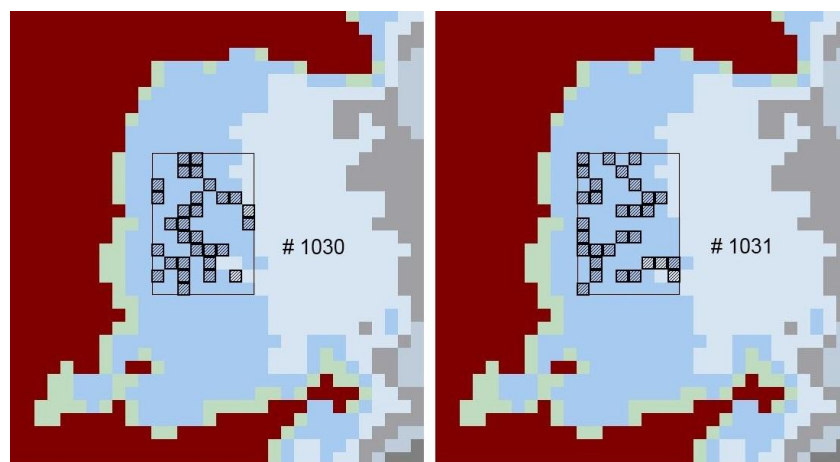


Figure 5-20: Graphical representation of the worst solutions (oysters: 30 cells, 86 400 steps – 1 month)

Fourth experimental set

For the new experimental sets, the allowed regions for exploitation were expanded and are represented in Figure 5-21. Also scallops were included to diversify bivalve culture, and the seeded areas were expanded and fixed in 60 cells, trying to simulate more realistic scenarios where the bivalve individuals are spread over the allowed area of exploration.

Three criteria were analysed in this experimental set:

- Which is more profitable - monoculture of oysters or scallops, or polyculture?
- What gives best results - concentrated or spread seed areas?
- Are there preferred exploitation areas for oysters or scallops?

The new experiences forced to change the way how results were saved and registered in the computers. Each farmer agent registers its own activity in different folders and files, named according to the configurations to test (see detailed information in Annex 4).

Sungo Bay was divided in five regions, with the same area of 88 cells (8 columns by 11 lines) used in the first experiments. The idea was to occupy 60 cells with bivalves (oyster or scallops). When monoculture is studied, all the 60 cells contain the same bivalve species. When polyculture is studied, each bivalve species occupies 30 cells. For the concentrated experiments all the 60 seed cells belong to the initial area A. For the distributed experiments the initial area A was expanded, in the latitude axis, creating regions B and C – Figure 5-21(a) – and in the longitude axis, area A was duplicated to the new “outer” area– Figure 5-21(b). Region A was renamed as “inner” region.

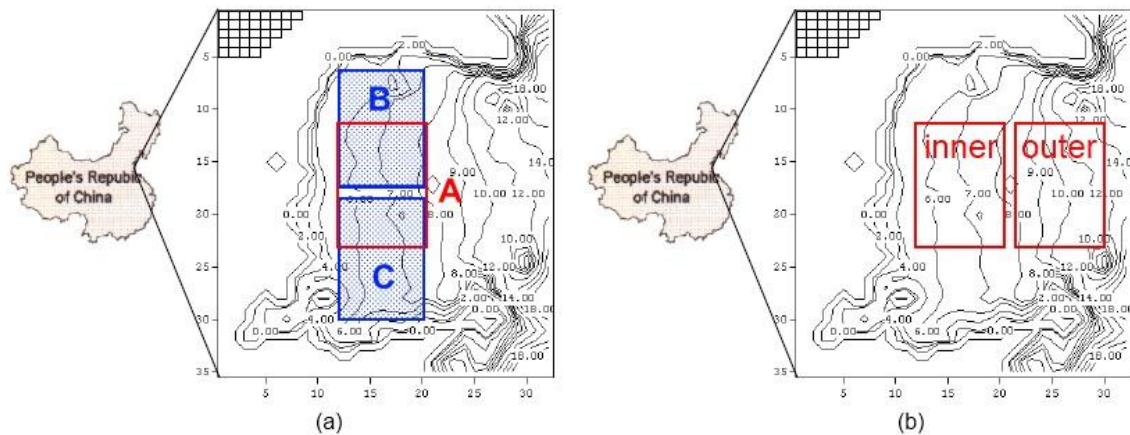


Figure 5-21: Exploitation areas for the new experiments.

The monoculture experiments with oysters and scallops are summarized in Table 5-4.

Table 5-4: Summary of bivalves monoculture experiments

| Oysters reference | Scallops reference | Description |
|-----------------------|-----------------------|--|
| CM::A.60_oys | CM::A.60_sca | Concentrate 60 cells in area A |
| DM::BC.60_oys | DM::BC.60_sca | Spread 60 cells between areas B and C |
| DM::IO.60_oys | DM::IO.60_sca | Spread 60 cells between inner and outer areas |
| DM::B.30_oys+C.30_oys | DM::B.30_sca+C.30_sca | Distribute 30 cells in area B and 30 cells in area C |
| DM::I.30_oys+O.30_oys | DM::I.30_sca+O.30_sca | Distribute 30 cells in inner area and 30 cells in outer area |

The first letter of each experiment reference stands for concentrated (C) or distributed (D) and the second (M) stands for monoculture. The remainder of the reference is easily understood.

Also experiments with polyculture were designed and are resumed in Table 5-5. In these experiments, 30 cells are seeded with oysters and another 30 cells with scallops. In some cases cells overlap. The first letter of each experiment reference maintains its significance and the second letter (P) stands for polyculture.

Table 5-5: Summary of polyculture experiments

| Experiment | Description |
|-------------------------|--|
| CP::A.30_oys+A.30_sca | Oysters and scallops spread over area A |
| DP::BC.30_oys+BC.30_sca | Oysters and scallops spread over areas B and C |
| DP::IO.30_oys+IO.30_sca | Oysters and scallops spread over inner and outer areas |
| DP::B.30_sca+C.30_oys | Scallops in area B and oysters in area C |
| DP::B.30_oys+C.30_sca | Oysters in area B and scallops in area C |
| DP::I.30_sca+O.30_oys | Scallops in inner area and oysters in outer area |
| DP::O.30_oys+I.30_sca | Oysters in inner area and scallops in outer area |

The experiments simulate one month of bivalves growth and the results are shown and discussed in the following sections. The objective of the mentioned experiments was to determine if there are privileged locations for oysters or scallops, if the coexistence of both species is beneficial or adverse. As the duration of the simulations was maintained (86 400 steps), the experiments took approximately the same time as the previous (about 20 minutes for one simulation and one day and six hours for each experimental round). Before starting the experimental sets, the initial values for the production indicators were registered in Table 5-6.

Table 5-6: Initial values for production indicators (P.I.) of oysters and scallops

| Species | P.I. (30 cells) | P.I. (60 cells) |
|----------|-----------------|-----------------|
| Oysters | 289 | 578 |
| Scallops | 531 039 | 1 062 078 |

MONOCULTURE EXPERIMENTS - OYSTERS

Table 5-7 makes a summary of the best results obtained (related to the initial values of the production indicators - Table 5-6). A first analysis to the numbers, after one month of simulation, suggests that the inner area seems to be better for oyster culture than the remaining tested areas (experiments CM::A.60_oys and DM::IO.60_oys).

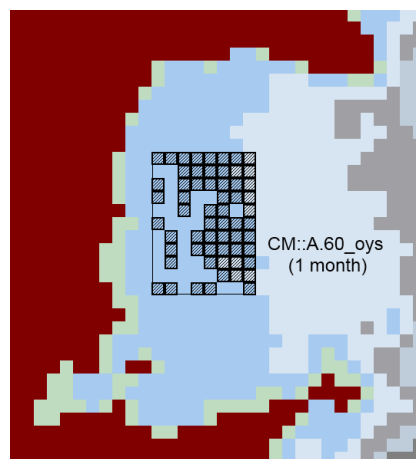
Table 5-7: Results from monoculture experiments – oysters (60 cells, 86 400 steps – 1 month)

| Task | Best value (P.I.) | Partial total B/I (#cells) | Partial total C/O (#cells) | Absolute Gain (P.I.) | Gain Ratio | Worst Value (P.I.) |
|-----------------------|----------------------|-------------------------------|-------------------------------|-------------------------|---------------|-----------------------|
| CM::A.60_oys | 80 876 | - | - | 80 288 | 138.92 | 78 648 |
| DM::BC.60_oys | 79 843 | 33 369 (25) | 46 474 (35) | 79 265 | 137.14 | 74 543 |
| DM::IO.60_oys | 80 668 | 65 646 (47) | 15 022 (13) | 80 090 | 138.56 | 71 146 |
| DM::B.30_oys+C.30_oys | 77 596 | 41 941 (30) | 35 655 (30) | 77 018 | 133.25 | 77 443 |
| DM::I.30_oys+O.30_oys | 77 012 | 49 864 (30) | 27 148 (30) | 76 434 | 132.24 | 72 317 |

The difference between maximum and minimum values for the production indicator (related with the minimum) differs according the experiment:

- CM::A.60_oys – 2.8%
- DM::BC.60_oys – 7.1%
- DM::IO.60_oys – 13.4%
- DM::B.30_oys+C.30_oys – 7.1%
- DM::I.30_oys+O.30_oys – 6.6%

and reveals that the more homogeneous results are obtained in the inner area. Figure 5-22 and Figure 5-23 show the distribution of cells with oysters from experiments CM::A.60_oys, DM::IO.60_oys and DM::BC.60_oys. It is more or less obvious that the central area of the bay is the best candidate to seed oysters.

**Figure 5-22: Graphical representation of the best solution for experiment CM::A.60_oys (1 month)**

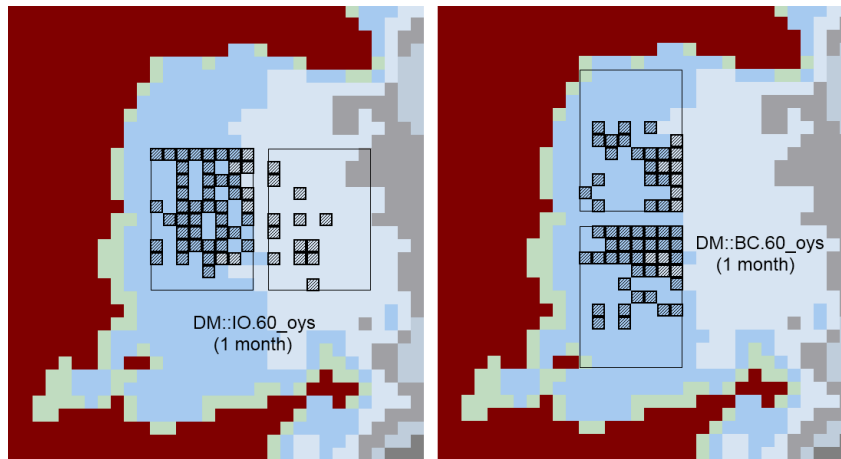


Figure 5-23: Graphical representation of the best solutions for experiments DM::IO.60_oys and DM::BC.60_oys (1 month)

MONOCULTURE EXPERIMENTS - SCALLOPS

Table 5-8 makes a summary of the results obtained (related to the initial values of the production indicators - Table 5-6). A first analysis to the numbers, after one month of simulation reveals, with no doubt, that the outer area seems to be much better for scallop culture than the remaining areas (experiment DM::IO.60_sca seeds 57 cells in the outer area).

The difference between maximum and minimum values for the production indicator (related with the minimum) differs according the experiment:

- CM::A.60_sca – 0.5%
- DM::BC.60_sca – 1.7%
- DM::IO.60_sca – 3.8%
- DM::B.30_sca+C.30_sca – 0.9%
- DM::I.30_sca+O.30_sca – 2.1%

and reveals that when outer area can be explored, the results show a significant improve.

Table 5-8: Results from monoculture experiments – scallops (60 cells, 86400 steps – 1 month)

| Task | Best value (P.I.) | Partial total B/I (#cells) | Partial total C/O (#cells) | Absolute Gain (P.I.) | Gain Ratio | Worst value (P.I.) |
|-----------------------|----------------------|-------------------------------|-------------------------------|-------------------------|---------------|-----------------------|
| CM::A.60_sca | 1 100 894 | - | - | 38 816 | 0.04 | 1 095 618 |
| DM::BC.60_sca | 1 102 847 | 555 738 (30) | 547 109 (30) | 40 769 | 0.04 | 1 084 033 |
| DM::IO.60_sca | 1 189 978 | 57 558 (3) | 1 132 420 (57) | 127 900 | 0.12 | 1 146 944 |
| DM::B.30_sca+C.30_sca | 1 095 374 | 547 364 (30) | 548 010 (30) | 33 296 | 0.03 | 1 085 848 |
| DM::I.30_sca+O.30_sca | 1 172 087 | 547 863 (30) | 624 224 (30) | 110 009 | 0.10 | 1 147 607 |

Figure 5-24 and Figure 5-25 show the distribution of cells with scallops from experiments DM::IO.60_sca, CM::A.60_sca and DM::BC.60_sca. Scallops seem to grow better in the outside area of the bay.

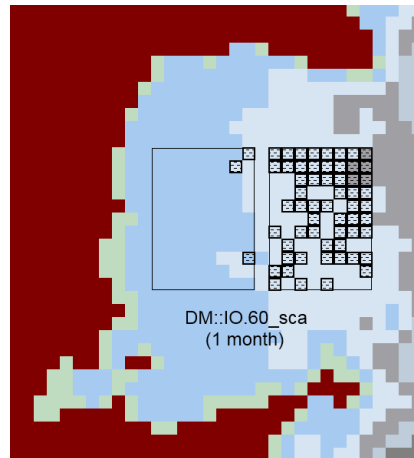


Figure 5-24: Graphical representation of the best solution for experiment DM::IO.60_sca (1 month)

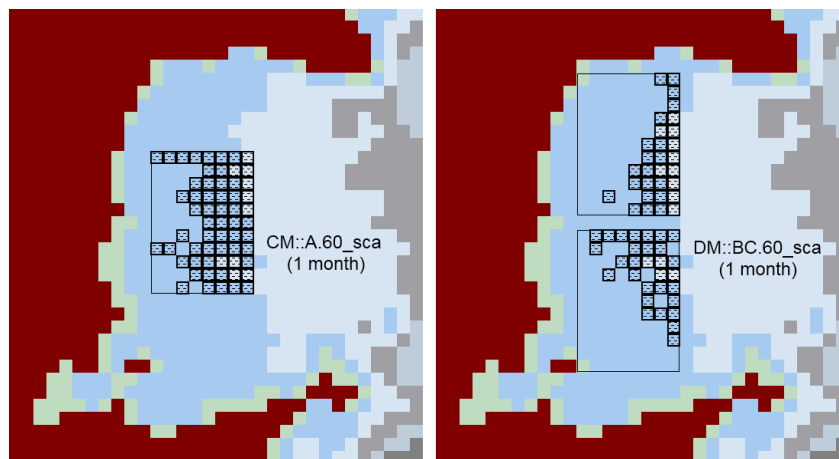


Figure 5-25: Graphical representation of the best solutions for experiments CM::A.60_sca and DM::BC.60_sca (1 month)

POLYCUULTURE EXPERIMENTS – SCALLOPS AND OYSTERS

The polyculture experiments referred in Table 5-5 are divided in two main groups:

- One group with specific areas for oysters and scallops.
- Another group where the bivalve species can be mixed:
 - In one concentrated area
 - In two areas.

All the results obtained were cross-compared and related to the initial values of the production indicators. Table 5-9 is a summary of the best results. A first analysis after one month of simulation confirms the idea that scallops seems to grow better if seeded in the outer region and oysters in the inner region.

In DP::BC.30_oys+BC.30_sca experiment, 14 cells share oysters and scallops culture, in DP::IO.30_oys+IO.30_sca, the mixed culture is shared in 12 cells, while in CP::A.30_oys+A.30_sca experiment, 16 cells are common. Oysters seems to grow better in the inner area and scallops in the outer area (DP::I.30_oys+O.30_sca), although when cultures are mixed the outer region reveals good results for both species (DP::IO.30_oys+IO.30_sca).

Table 5-9: Results from polyculture experiments – oysters (30 cells) and scallops (30 cells) [86400 steps, 1 month]

| Task | Best value (P. I.) | Partial total B/I (#cells) | Partial total C/O (#cells) | Absolute Gain Oyster/Scallop (P.I.) | Gain Ratio Oyster/Scallop | Worst value (P.I.) |
|-------------------------|-----------------------|---|---|---|------------------------------|--------------------------|
| CP::A.30_oys+A.30_sca | 595 002 | - | - | 30 027 / 33 647 | 103.9 / 0.06 | 583 328 |
| DP::BC.30_oys+BC.30_sca | 588 272 | oys: 18 452 (22) sca :444 752 (24) | oys: 9770 (8) sca :115 297 (6) | 27 934 / 29 010 | 96.66 / 0.05 | 578 125 |
| DP::IO.30_oys+IO.30_sca | 647 110 | oys: 18 516 (13) sca: 76 859 (4) | Oys :12 123 (17) sca: 539 612 (26) | 30 349 / 85 432 | 105.0 / 0.16 | 609 326 |
| DP::B.30_sca+C.30_oys | 587 645 | - | - | 33 330 / 22 987 | 115.3 / 0.04 | 582 907 |
| DP::B.30_oys+C.30_sca | 582 098 | - | - | 28 833 / 21 938 | 99.8 / 0.04 | 573 040 |
| DP::I.30_sca+O.30_oys | 586 314 | - | - | 21 101 / 33 884 | 73.0 / 0.06 | 578 427 |
| DP::I.30_oys+O.30_sca | 670 479 | - | - | 39 158 / 99 993 | 135.4 / 0.19 | 648 567 |

The difference between maximum and minimum values for the production indicator (related with the minimum) differs according the experiment:

- CP::A.60_oys+A.30_sca – 2.0%

- DP::BC.30_oys+BC.30_sca – 1.8%
- DP::IO.30_oys+IO.30_sca – 6.2%
- DP::B.30_sca+C.30_oys – 0.8%
- DP::B.30_oys+C.30_sca – 1.6%
- DP::I.30_sca+O.30_oys – 1.4%
- DP::I.30_oys+O.30_sca – 3.4%

Figure 5-26 to Figure 5-29 show the distribution of cells with scallops and oysters from experiments DP::I.30_oys+O.30_sca, DP::IO.30_oys+IO.30_sca, CP::A.30_oys+A.30_sca and DP::BC.30_oys+BC.30_sca.

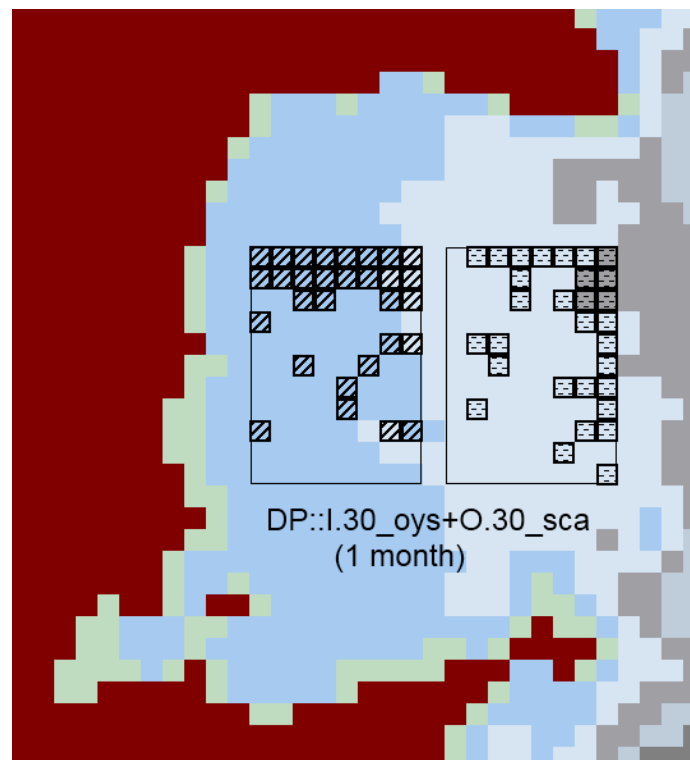


Figure 5-26: Graphical representation of the best solution for experiment DP::I.30_oys+O.30_sca (1 month)

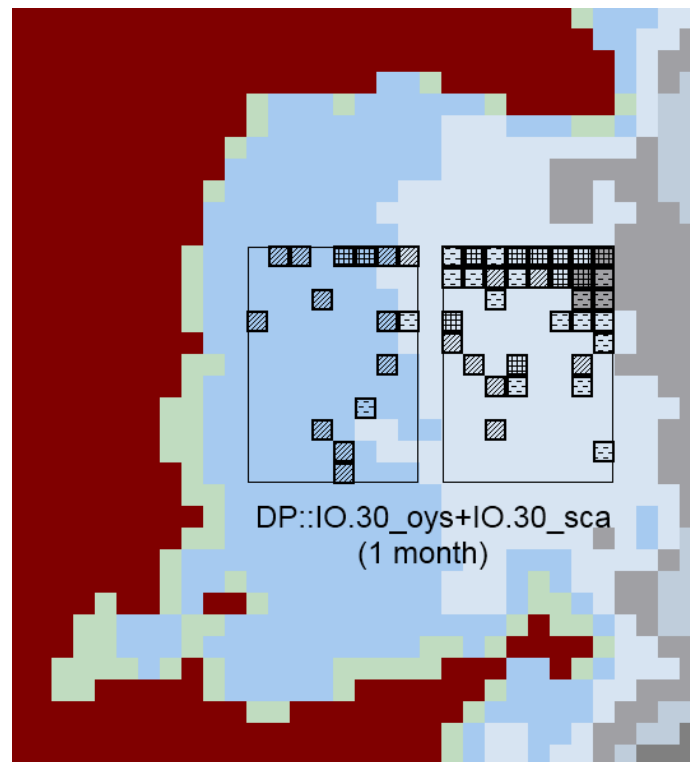


Figure 5-27: Graphical representation of the best solution for experiment DP::IO.30_oys+IO.30_sca (1 month) – 12 cells with mix culture are represented by small grid

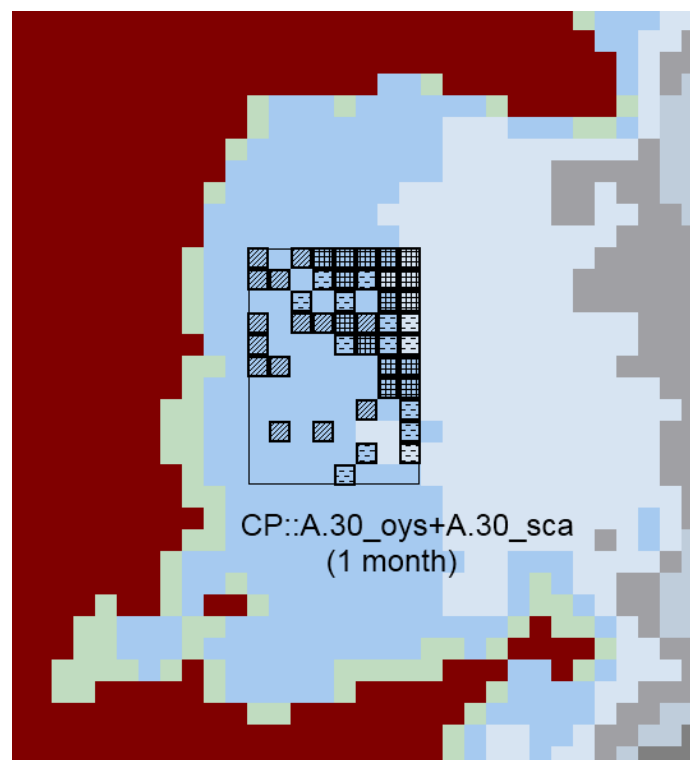


Figure 5-28: Graphical representation of the best solution for experiment CP::A.30_oys+A.30_sca (1 month) – 16 cells with mix culture are represented by small grid

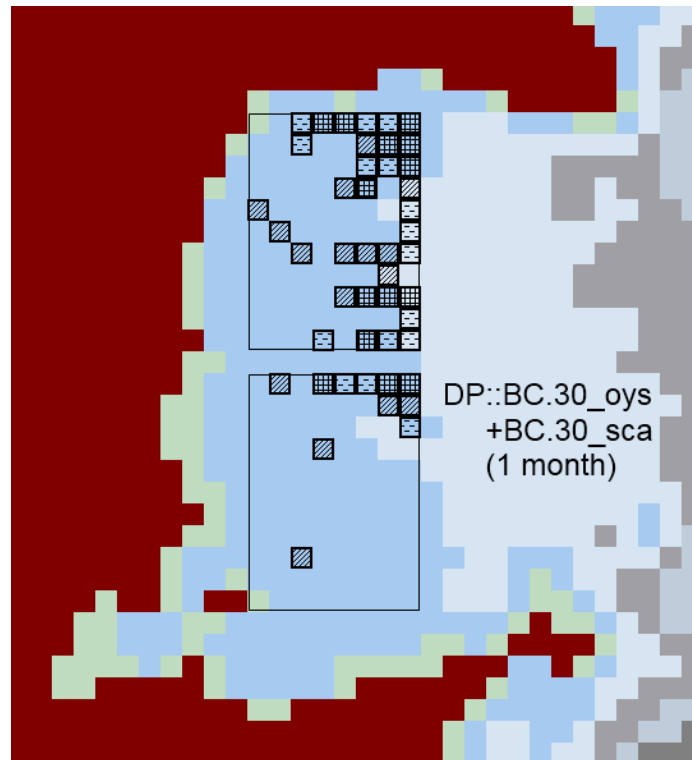


Figure 5-29: Graphical representation of the best solution for experiment DP::BC.30_oys+BC.30_sca (1 month) - 14 cells with mix culture are represented by small grid

CRITICAL ANALYSIS OF THE RESULTS

Some of the questions posed when this experimental set was launched can be answered looking to the results, but a critical analysis must be done.

To verify which is more profitable – monoculture of oysters or scallops, or polyculture – it is necessary to have market values, since the results obtained are measured in total biomass harvested for each species. In a simulation representing one month of real time it seems that oysters grow more quickly than scallops – oysters are seeded with a smaller size than scallops, and the rate of growth is, generally, faster in smaller individuals.

Monoculture of oysters gives better results when seeding area is concentrated in the inner region, while monoculture of scallops gives better results when they are spread over the outer region. When polyculture is adopted, best results point to oysters separated from scallops – oysters in the inner region and scallops in the outer region, reinforcing the monoculture trends.

Fifth experimental set

To verify if the trends revealed in previous experiments were reliable, based on one month real time simulations, a series of one year and a half simulations were realized. Inner and outer

areas were the focus in the longer experiments and Table 5-10 lists the experiments that were replicated, now with a realistic complete simulation, but with a minor number of iterations.

Table 5-10: Experiments extended to one year and a half

| Experiment | Description |
|-------------------------|--|
| DM::IO.60_oys | Oysters spread over inner and outer areas |
| DM::I.30_oys+O.30_oys | Oysters spread equally between inner and outer areas |
| DM::IO.60_sca | Scallops spread over inner and outer areas |
| DM::I.30_sca+O.30_sca | Scallops spread equally between inner and outer areas |
| DP::IO.30_oys+IO.30_sca | Oysters and scallops spread over inner and outer areas |
| DP::I.30_sca+O.30_oys | Scallops in inner area and oysters in outer area |
| DP::I.30_oys+O.30_sca | Oysters in inner area and scallops in outer area |

The results obtained are presented in the following sections.

MONOCULTURE EXPERIMENTS - OYSTERS

Table 5-11 makes a summary of the best results obtained (related to the initial values of the production indicators - Table 5-6). A first analysis to the numbers, after one complete simulation, suggests that the outer area seems to be better for oyster culture than inner area.

Table 5-11: Results from monoculture experiments – oysters (60 cells, 1 555 200 steps – 1.5 year)

| Task | Best value (P.I.) | Partial total I (#cells) | Partial total O (#cells) | Absolute Gain (P.I.) | Gain Ratio | Worst Value (P.I.) |
|-----------------------|----------------------|-----------------------------|-----------------------------|-------------------------|---------------|-----------------------|
| DM::IO.60_oys | 2 947 106 | 706 797 (22) | 2 240 309 (38) | 2 946 527 | 5094.6 | 2 494 308 |
| DM::I.30_oys+O.30_oys | 2 867 246 | 941 772 (30) | 1 925 474 (30) | 2 866 668 | 4956.5 | 2 670 683 |

The difference between maximum and minimum values for the production indicator (related with the minimum) differs according the experiment:

- DM::IO.60_oys – 18.2%
- DM::I.30_oys+O.30_oys – 7.4%

The best solution, obtained in experiment DM::IO.60_oys, tends to occupy more cells in the outer area, which contradicts the results from one month simulation (cf. Table 5-7). Figure 5-30 shows the distribution of cells with oysters from experiments DM::IO.60_oys and DM::I.30_oys+O.30_oys. There is no evident trend for the oysters locations, although in each

area they tend to occupy the locations where the water dynamics is more intense (Duarte et al., 2005b).

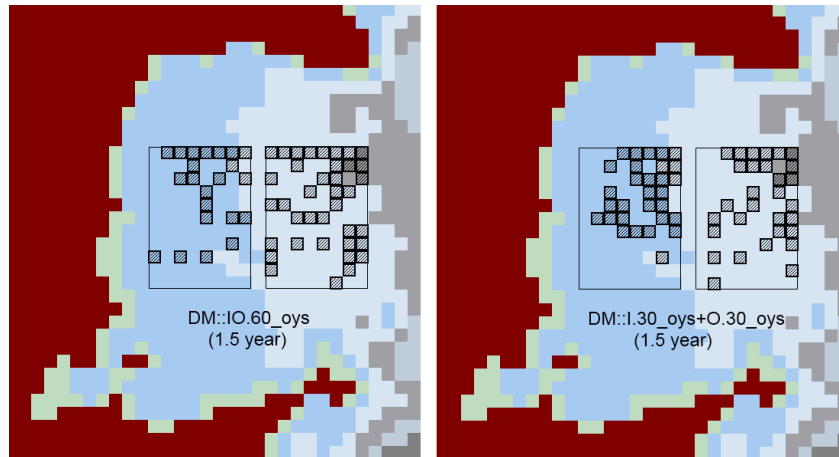


Figure 5-30: Graphical representation of the best solutions for experiments DM::IO.60_oys and DM::I.30_oys+O.30_oys (1.5 year)

MONOCULTURE EXPERIMENTS - SCALLOPS

Table 5-12 makes a summary of the best results obtained (related to the initial values of the production indicators - Table 5-6). A first analysis to the numbers, after one complete simulation, suggests that the outer area is better for scallops culture than inner area.

Table 5-12: Results from monoculture experiments – scallops (60 cells, 1 555 200 steps – 1.5 year)

| Task | Best value (P.I.) | Partial total I (#cells) | Partial total O (#cells) | Absolute Gain (P.I.) | Gain Ratio | Worst Value (P.I.) |
|-----------------------|----------------------|-----------------------------|-----------------------------|-------------------------|---------------|-----------------------|
| DM::IO.60_sca | 10 311 170 | 1 082 796 (7) | 9 228 375 (53) | 9 249 093 | 8.7 | 7 090 869 |
| DM::I.30_sca+O.30_sca | 8 194 084 | 2 356 906 (30) | 5 837 178 (30) | 7 132 006 | 6.7 | 7 079 137 |

The difference between maximum and minimum values for the production indicator (related with the minimum) differs according the experiment:

- DM::IO.60_sca – 45%
- DM::I.30_sca+O.30_sca – 15.7%

and the best solution, obtained in experiment DM::IO.60_sca, tends to distribute almost all cells in the outer area, which confirms the results from one month simulation (cf. Table 5-8).

Figure 5-31 shows the distribution of cells with scallops from experiments DM::IO.60_sca and DM::I.30_sca+O.30_sca.

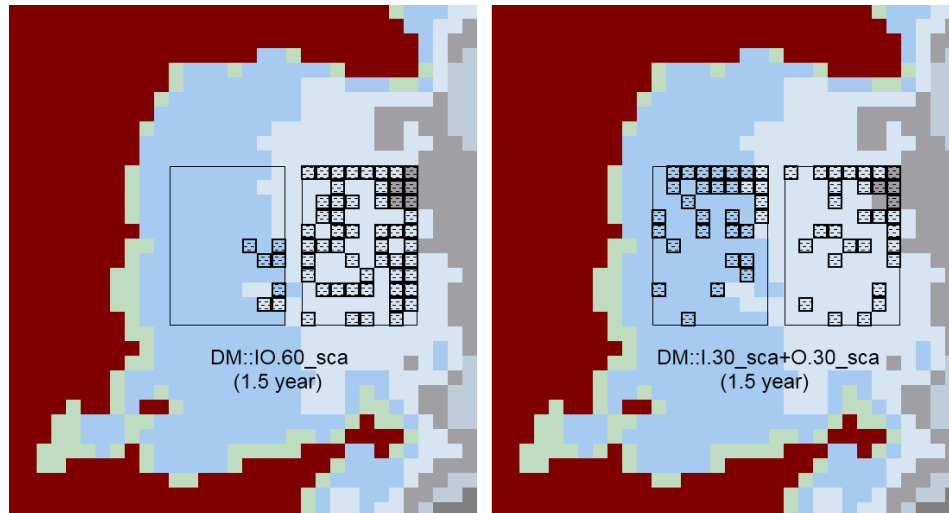


Figure 5-31: Graphical representation of the best solutions for experiments DM::IO.60_sca and DM::I.30_sca+O.30_sca (1.5 year)

POLY CULTURE EXPERIMENTS – SCALLOPS AND OYSTERS

Table 5-13 makes a summary of the best results obtained (related to the initial values of the production indicators - Table 5-6).

Table 5-13: Results from polyculture experiments - oysters (30 cells) and scallops (30 cells) [1555200 steps, 1 year and half]

| Task | Best value (P. I.) | Partial total I (#cells) | Partial total O (#cells) | Absolute Gain Oyster/Scallop (P.I.) | Gain Ratio Oyster/Scallop | Worst value (P.I.) |
|-----------------------------|-----------------------|---|--|---|------------------------------|--------------------------|
| DP::IO.30_oys +IO.30_sca | 5 932 032 | oys: 597 361 (16) sca: 733 506 (9) | oys :698 149 (14) sca: 3 903 015 (21) | 1 295 221 / 4 105 482 | 4479 / 7.73 | 4 878 624 |
| DP::I.30_sca +O.30_oys | 3 972 678 | sca: 2 036 264 | oys: 1 936 414 | 1 936 125 / 1 505 225 | 6695 / 2.83 | 3 172 454 |
| DP::I.30_oys +O.30_sca | 6 422 176 | oys: 1 073 752 | sca: 5 348 423 | 1 073 463 / 4 817 384 | 3714 / 9.07 | 5 646 960 |

The difference between maximum and minimum values for the production indicator (related with the minimum) differs according the experiment:

- DP::IO.30_oys+IO.30_sca – 21.6%
- DP::I.30_sca+O.30_oys – 25.2%
- DP::I.30_oys+O.30_sca – 13.7%

Although the best solution was obtained when oysters were in inner area and scallops in outer area, obtained in experiment DP::I.30_oys+O.30_sca, outer area reveals better performance for oysters and scallops grow, which aligns with the trends revealed by monoculture experiments.

Figure 5-32 and Figure 5-33 show the distribution of cells with oysters and scallops from experiments DP::I.30_oys+O.30_sca, DP::I.30_sca+O.30_oys and DP::IO.30_oys+IO.30_sca.

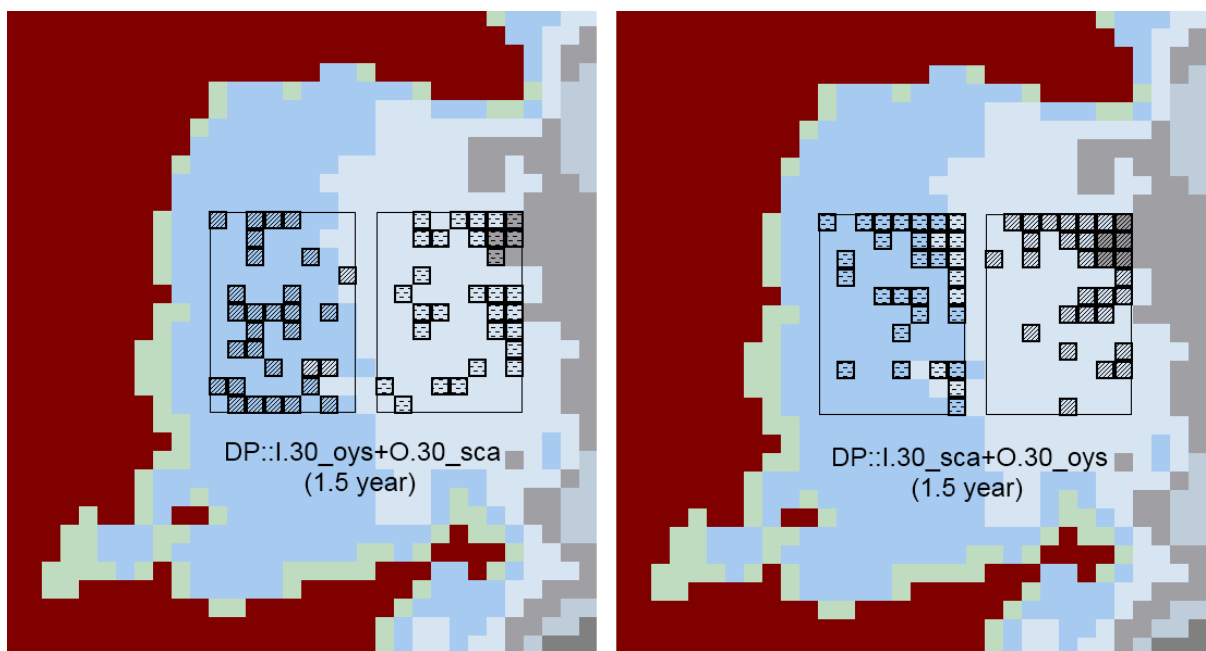


Figure 5-32: Graphical representation of the best solutions for experiments DP::I.30_oys+O.30_sca and DP::I.30_sca+O.30_oys (1.5 year)

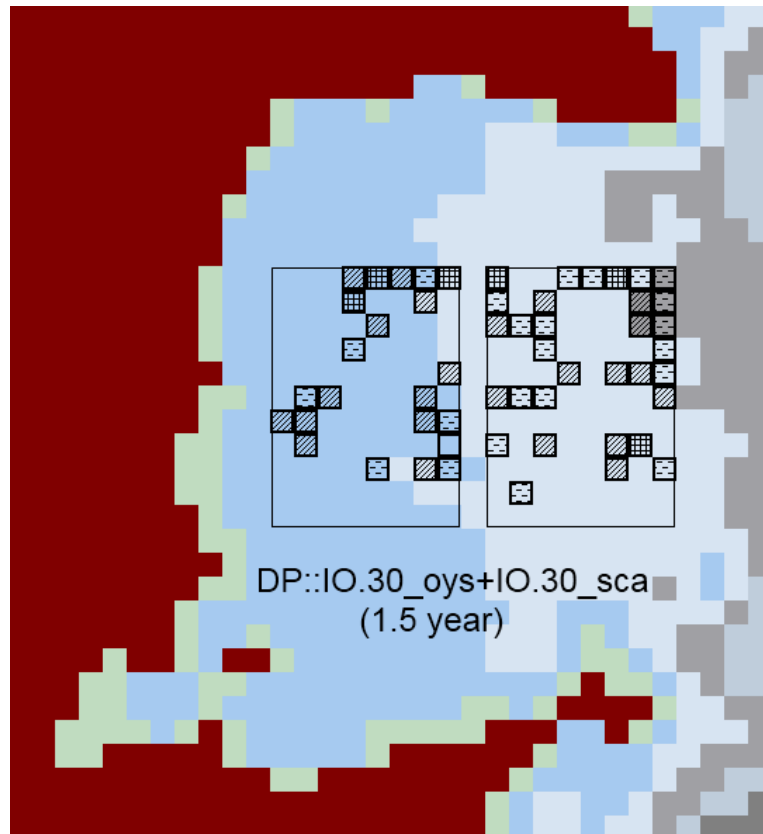


Figure 5-33: Graphical representation of best solution for experiment DP::IO.30_oys+IO.30_sca (1.5 year)

CRITICAL ANALYSIS OF THE RESULTS

In the case of oysters culture, the results obtained with a partial real time simulation (one month) reveals a trend that is not confirmed with a total real time simulation. In the first case, inner area had better results and in the second, outer area is more profitable.

In the case of scallops culture, the results obtained by both real time experiments coincide.

5.4.2 Carrying Capacity

The experiences on carrying capacity were based on the results from the work of Duarte et al. (2003) in Sungo Bay.

Some simple experiments were done with oysters, taking advantage from its rapid growth in the early stages of the simulation. The study focused in the inner area of the lagoon (Figure 5-14) with 88 areas available, where multiples of five areas were seeded, from five to 60, with two different densities of oysters – the value used in the previous experiences (55 individuals/m²) and the double (110 individuals/m²). For each density value, seven simulations with different time spans were performed, as resumed in the Table 5-14.

Table 5-14: Carrying capacity experiments

| Number of steps | Simulation time | Number of cells to seed |
|-----------------|----------------------|---|
| 5 | Initial value | 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60 |
| 1000 | 8 hrs 20 min | |
| 5000 | 41 hrs 40 min | |
| 25000 | 8 days 16 hrs 20 min | |
| 50000 | 17 days 8 hrs 40 min | |
| 86400 | 30 days | |
| 172800 | 60 days | |

The results are shown graphically in the following figures.

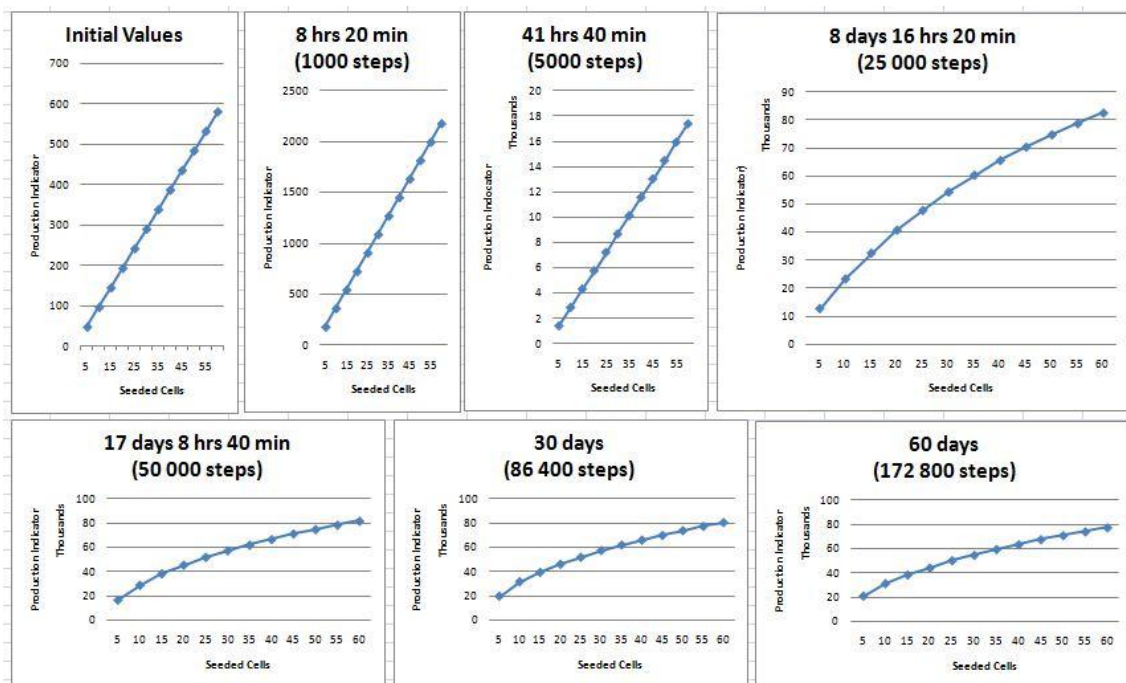


Figure 5-34: Carrying capacity studies for oysters (density = 55 individuals/m²)

It is easily seen that while in Figure 5-34 (normal density) the rate of growth is more or less linear for each simulation, when the density is doubled the growth rate loses its linearity when more than eight days are simulated (Figure 5-35). After 60 days, the result obtained with 60 seeded cells is practically the same than the obtained with 10 seeded cells, perfectly demonstrating the negative feedback to oysters' growth resulting from food limitation.

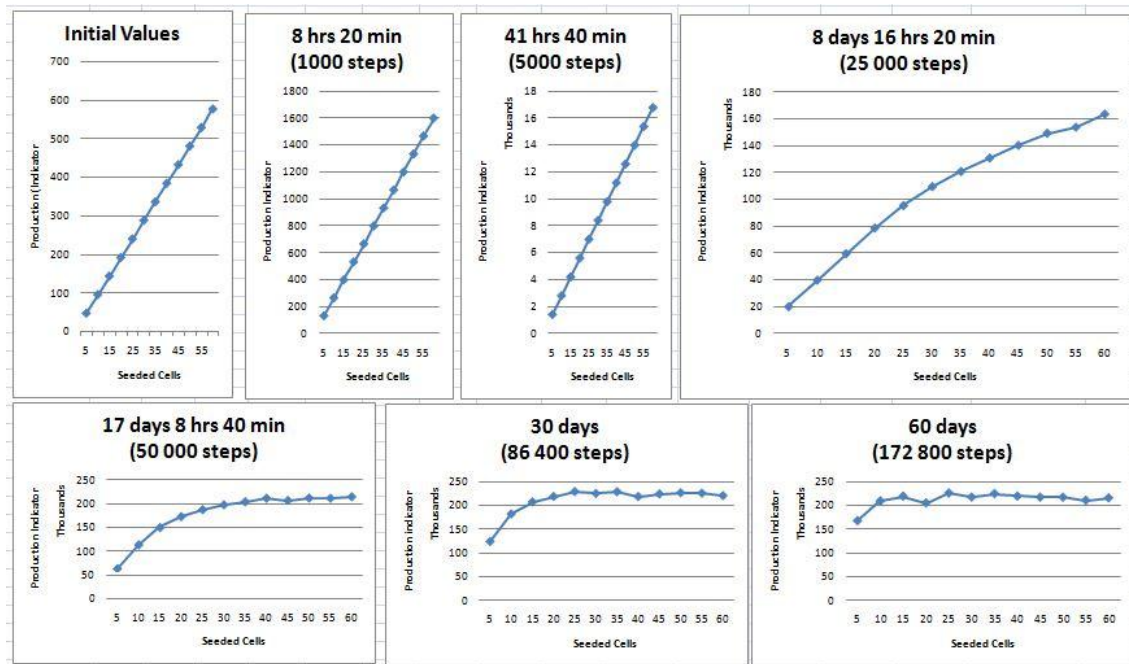


Figure 5-35: Carrying capacity studies for oysters (density = 110 individuals/m²)

5.5 Environmental DSS with AHP

The experiments made to test the AHP integration in the platform involved scenarios applied to the management of clam farming in Ria Formosa lagoon (Figure 5-36), integrating three opposing criteria: one simple economic criterion, one environmental criterion and one merely ecologic criterion.

The decision process integrated aquaculture revenue (euro / kg) as indicator for the economic criterion, water quality as indicator for the environmental criterion and chlorophyll concentration as indicator for the ecologic criterion.

The goal here is to maximize the economic criterion, measured by clams biomass ("*Ruditapes decussatus* biomass") per m² multiplied by the price per kg (considered as 5.00 €/kg, typical value in Portugal), the ecologic criterion, measured by "Phytoplankton biomass", while minimizing the water quality criterion, measured by the "Ammonia" variable. The AHP analysis assumes that all criteria must be maximized or minimized – therefore, the indicator for water quality criterion was changed to maximize the inverse of ammonia concentration (1 / Ammonia).

To distinguish the proposals sent to AHP, different criteria weights were defined:

- Consider that all criteria have the same importance – matrix A₁;

- Consider the environmental criterion (concentration of ammonia) more important than the others – matrix A_2 ;
- Consider the economic criterion (value of harvested clams) more important than the others – matrix A_3 .

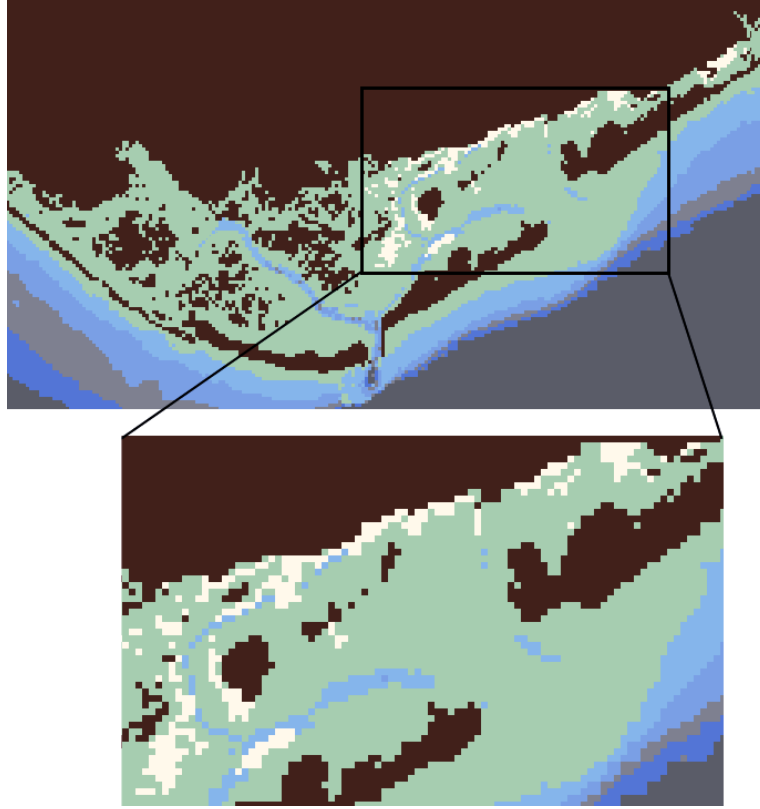


Figure 5-36: Ria Formosa clam farming areas (white cells)

The three different pairwise comparison matrices (PCM) obtained (A_1 , A_2 and A_3) are presented in Figure 5-37:

$$A_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 2 & 3 \\ 1/2 & 1 & 2 \\ 1/3 & 1/2 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 2 & 1 & 1/2 \\ 3 & 2 & 1 \end{bmatrix}$$

Figure 5-37: PCM matrices defined to AHP analysis

Three scenarios were generated for the experiments:

- The first scenario simulated is referred as Business As Usual (BAU) scenario, considering the usual exploration of the system (normal density of clams in licensed cultivation areas);

- The second scenario considers the hypothetical double density exploration of clams farming in licensed areas;
- The third scenario considers triple density of clams farming, maintaining the number of licensed areas where clams are cultivated – the intertidal areas of the lagoon.

The simulation results for the different density concentrations of clams are shown in Table 5-15, considering the mean values of the variables for the lagoon area:

Table 5-15: Simulation results for different clams' density

| Scenario / Criterion | Ammonia ($\mu\text{mol l}^{-1}$) | Phytoplankton ($\mu\text{g Chl l}^{-1}$) | R. decussatus (kg (DW) m^{-2}) |
|----------------------|---------------------------------------|---|---|
| Normal density | 11.4 | 0.33 | 1.4 |
| Double density | 17.7 | 0.29 | 2.2 |
| Triple density | 24.7 | 0.27 | 3.0 |

For a more realistic analysis, one controllable variable was defined: the clams price per kg was considered fixed in 5.00€/kg or elastic, decreasing 10% per kg if clams biomass exceeds 2kg m^{-2} and decreasing more 10% if clams biomass exceeds 3kg m^{-2} , reflecting the market price variation when the offer increases.

The normal, double and triple densities combined with the controllable variable generated six scenarios: scenarios 1 to 3 with fixed price, scenarios 4 to 6 with elastic price, although scenarios 1 and 4 are coincident because the price remains unchanged with normal density. The values inserted for the indicators in each scenario generated the following results:

Table 5-16: Scenarios with indicators values

| Scenario / Indicator | 1/Ammonia | Phytoplankton | Clam production x price |
|-------------------------------------|-----------|---------------|-------------------------|
| Sc1: Normal density – fixed price | 0.0877 | 0.33 | 7.00 |
| Sc2: Double density – fixed price | 0.0565 | 0.29 | 11.00 |
| Sc3: Triple density – fixed price | 0.0405 | 0.27 | 15.00 |
| Sc4: Normal density – elastic price | 0.0877 | 0.33 | 7.00 |
| Sc5: Double density – elastic price | 0.0565 | 0.29 | 9.90 |
| Sc6: Triple density - elastic price | 0.0405 | 0.27 | 12.15 |

Running the AHP application (cf. 4.5.2) combining scenarios results and pairwise comparison matrices, produced the following results:

Table 5-17: AHP scores under fixed and elastic prices with PCM A_1 , A_2 and A_3

| Scenario / PCM | A_1 | A_2 | A_3 |
|-------------------------------------|---------------|---------------|---------------|
| Sc1: Normal density – fixed price | 0.5265 | 0.6447 | 0.3831 |
| Sc2: Double density – fixed price | 0.1888 | 0.1821 | 0.1970 |
| Sc3: Triple density – fixed price | 0.2846 | 0.1731 | 0.4199 |
| Sc4: Normal density – elastic price | 0.5261 | 0.6445 | 0.3824 |
| Sc5: Double density – elastic price | 0.1945 | 0.1849 | 0.2062 |
| Sc6: Triple density - elastic price | 0.2794 | 0.1706 | 0.4114 |

The graphical outputs generated by the AHP application gives a quick visual information of the results, as showed in Figure 5-38, where scores for scenarios with fixed prices appear in column (a) and scores for scenarios with elastic prices appear in column (b).

Several conclusions may be drawn from these results. Fixed or elastic prices have similar behaviours, considering the rules enounced for elastic prices. The BAU scenario (Sc1/Sc4) is the preferred exploration scenario, except when the economic criterion is considered the major factor for decision (A_3) – in this case, there is a small advantage from scenarios with triple density of clam culture. Whatever the criteria importance considered, the scenario with double density should be avoided because it seems the more disadvantageous.

This study is not a complete assessment of the consequences of the scenarios referred, but solely an attempt to approach their effects at the lagoon scale. One possible evolution of this approach could be the integration of metaheuristics methods with AHP methodology in order to maximize the sustainability of the scenarios. In experiments realized, criteria used in AHP were based in single indicators derived from single variables. In a more realistic study, indicators should be calculated integrating values from more than one state variable. For instance, the water quality criterion depends not only from ammonia level but also from nutrients, oxygen saturation, chlorophyll, etc., and there are indicators that combine different variables by weighting each (Austoni et al., 2004; Duarte et al., 2007).

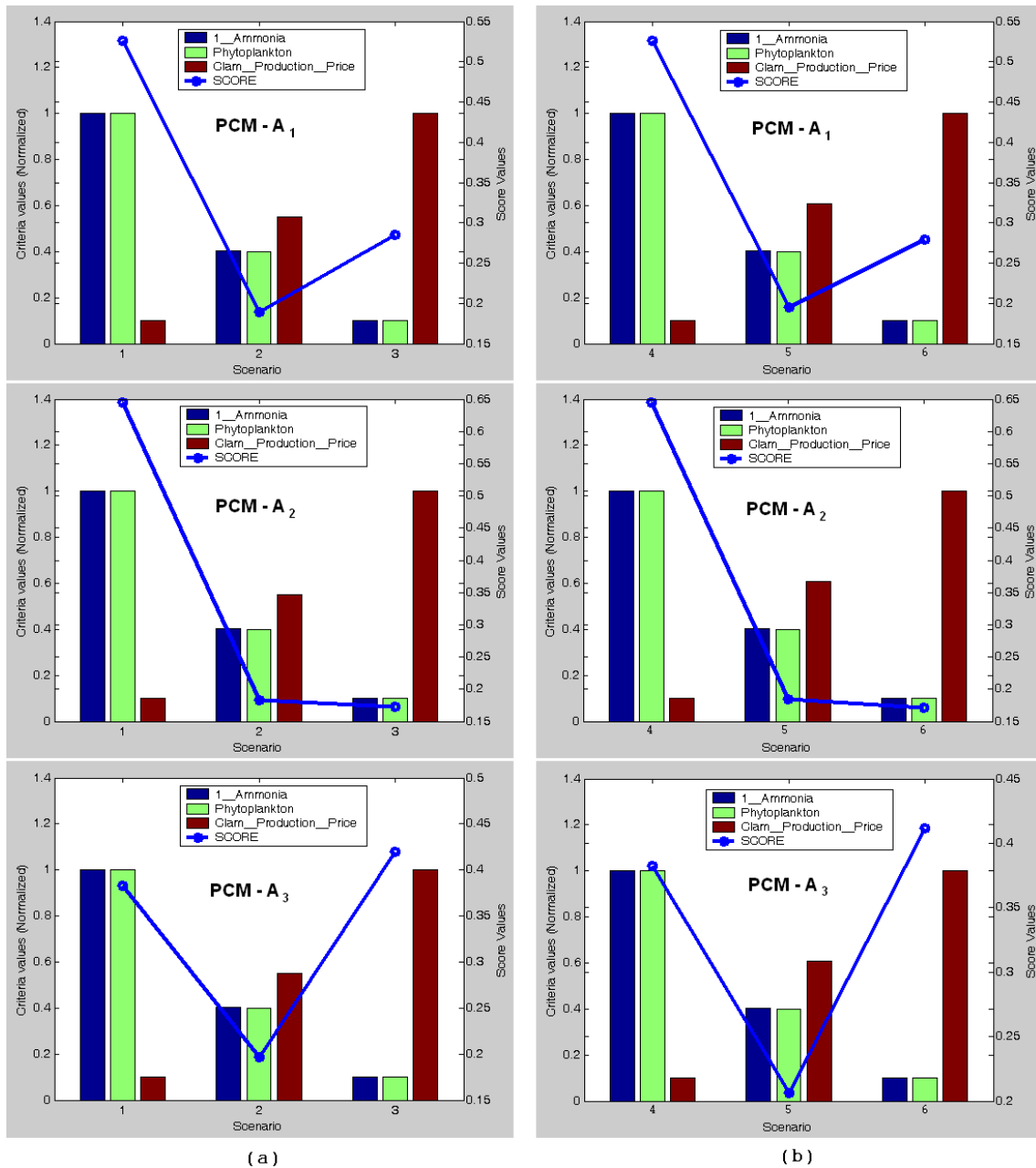


Figure 5-38: Graphical representation of AHP scenarios analysis: (a) fixed prices; (b) elastic prices

5.6 Conclusions

The experiments described in this chapter and the results obtained show that simulation of complex ecosystems don't have to be incomprehensible to the end users.

Simple aquaculture optimization experiments were presented, with several alternatives, to demonstrate the flexibility and the power of the developed applications that constitute the EcoSimNet framework for ecological simulations. The existence of an agent with optimization algorithms to determine the carrying capacity or the optimal scenario for bivalves exploitation,

was a demonstration of the concept, together with the applicability of the AHP methodology to integrate distinct and antagonistic criteria in ecological management decisions.

Considering the complex network of processes and feedbacks that maintain coastal ecosystems, tools designed with high-level of communication and online supervision are necessary for managers and stakeholders, to help design management practices towards sustainability, without hampering economic activities, leisure and environmental preservation.

6 Conclusions and Future Work

*“We are what we do, but we are mainly what we do to
change what we are.”*

Eduardo Galeano (Uruguayan writer)

6.1 Summary and Conclusions

The integrated management of water resources and coastal ecosystems in order to achieve a sustainable development is an attractive but, almost, academic idea: there are hardly any real world successful cases. This type of management must be able to integrate constantly new factors that could influence decisions – factors related with new knowledge about the ecological processes, or related with new human actions over the environment. Computational systems, that help managers and decision makers to foreseen consequences of the decisions over the ecosystems, are welcome to improve the quality, the coherence and to reduce the time needed to take a decision.

The research presented in this work reveals some highlights to facilitate the development of an environmental decision support system (EDSS) integrating simulation tools, agents to generate and optimize exploration scenarios and a methodology to weight different and antagonistic criteria in ecological management decisions. Thinking about the need for the integration of different simulators, developed by different research teams, and different applications that support ecosystems management, the framework EcoSimNet was centred in one common communication language (ECOLANG) that facilitates the interface between all

the participants in the network, and the interoperability between heterogeneous applications and users.

An important feature exhibited by this work is the facility with which people without expertise in the ecological domain interact with the framework by modifying existing configurations, generating hypothetical exploration scenarios, defining their own interests and analyzing the results, getting an idea about the consequences of actions devised.

The simulator is the fundamental element in any computational platform that intends to study an ecosystem. It integrates all the knowledge about the physical and biogeochemical processes, as well as their interrelationships and feedbacks. In this work, the simulator EcoDynamo was designed using a modular structure, to enable the integration of new communications, commands and multi-thread modules – the multi-thread module is responsible for the parallelization of the simulation processes, accelerating the simulation execution, the communications module is responsible for the interfaces with other applications, and the commands module acts as the semantic module, responsible for the interface with other applications, giving meaning to the commands received and reflecting to the exterior the internal dynamics of the simulation.

All the simulation processes belong to a library of dynamic linkable objects, easily expandable to include new processes or knowledge about the ecosystem. The design also included the creation of new interfaces in those classes, enabling their usage by other simulators, written in different programming languages and developed by different research teams.

To include the entities that lead and have interest in an aquatic ecosystem, a multi-agent simulation system was developed to integrate the simulator and applications representing partners involved with the system – stakeholders and users. The framework received the name of EcoSimNet and the applications of the computational system are connected by TCP/IP sockets, over which ECOLANG messages are exchanged. The applications that represent local partners could be developed as agents, containing built-in optimization algorithms, that have goals to achieve in behalf of its represented. The flexibility of EcoSimNet enables the existence of several simulators in parallel and provides support for the coexistence of clusters (for simulation or optimization), used for multi-objective simulations and management issues.

The agent is the main interface to configure the simulations. Hydrologic models for the watershed flow and forcing functions generate the water dynamics to run the model. The results of the ecological model for the different scenarios can be compared with

environmental and economical indicators, through a decision support system included in the agent or in external applications, and help the stakeholders in the decision making process.

To test the framework and its functionalities, different scenarios on two different models were simulated and the results obtained worked as proofs of concept. Scenarios for aquaculture optimization and for determination of the system carrying capacity were simulated with the Sungo Bay model, as well as a brief experiment in ecosystem management done with the model of Ria Formosa.

From a rationalist point of view, the idea of environmental decision support systems (EDSS) is attractive, and DSS are vastly successful in fields that operate predominantly in rational principles, like scheduling, air traffic, and management of order flow or manufacturing plans. When the domain is politically dominated, like the natural environment, the rational argument suffers a setback because it has to deal with multiple and conflicting criteria, expectations, perceptions and hidden agenda. The politician decision maker has to take decisions that are not only based on rationality, and he doesn't want a system that helps him to take decisions: he needs a system that could justify the decisions. The EDSS could be relevant for policy making since it involves politicians, population and stakeholders, empowered by shared reliable information.

Another purpose of any EDSS is to make the participants in the decision process to think in a structured way about the problem. A common language must emerge to describe the basic elements of the decision process, clarifying the definition of criteria, objectives and constraints. The formulation of good questions highlights the discovery of good answers and contributes to a shared responsibility for any decision taken. Data compilation, model building and scenario analysis should be focused to the requirements of the EDSS to ensure that information finally available is complete and meaningful. The design of an EDSS should primarily answer the questions of "what for" and "why", to reach the answer to the question "how".

An EDSS should be used routinely in the decision processes, and its usage is the measure of its success. The institutional integration of an EDSS, the change and learning processes can take many years. The expectation is that the framework described in this document can be used by the institutional authorities and private environmental enterprises in order to help the national and international management of coastal and aquatic ecosystems.

6.2 Future Work

The future of the system will include the formation of coalitions between similar agents in order to better achieve their objectives without undermining sustainable development in the region and in the ecosystem in particular. This type of coalitions represent associations of producers of shellfish, for example, associations of municipalities or associations for tourism or owners.

The integration of all these applications and agents in the system will generate more distinguishable scenarios and bring the simulations closer to the reality, and will provide a more comprehensive decision-making process focused on the management of coastal ecosystems.

Certainly, the methods and procedures adopted in this work could be applied to other ecosystems, and with the appropriate approaches, to other decision' systems with multiple and conflicting interests in confront.

One interesting application of these methodologies could be tested in the project of a Calibration Agent (CA) with automatic and optimized processes for the calibration of "any" model, without access to its database or the mathematical equations that describes the model behaviour. This agent could be integrated in the EcoSimNet framework supplying the system with the VV&A phases of the simulation process (cf. 2.2.1), communicating through ECOLANG messages and implementing some ideas conceptualized in Pereira et al. (2004a).

References

- Adriaenssens, V., Goethals, P. L. M., and De Pauw, N. 2006. "Fuzzy knowledge-based models for prediction of *Asellus* and *Gammarus* in watercourses in Flanders (Belgium)". *Ecological Modelling*, 195 (1-2): 3-10.
- Amirjanov, A. 2006. "The development of a changing range genetic algorithm". *Computer Methods in Applied Mechanics & Engineering*, 195 (19-22): 2495-2508.
- Aoki, I., Komatsu, T., and Hwang, K. 1999. "Prediction of response of zooplankton biomass to climatic and oceanic changes". *Ecological Modelling*, 120 (2-3): 261-270.
- Arakawa, A., and Lamb, V. R. 1977. "Computational design of the basic dynamical process of the UCLA general circulation model". *Methods in Computational Physics*, 17: 173-265.
- Arnold, J. G., and Fohrer, N. 2005. "SWAT2000: current capabilities and research opportunities in applied watershed modelling". *Hydrological Processes*, 19 (3): 563-572.
- Austoni, M., Viaroli, P., Giordani, G., and Zaldívar, J. M. 2004. *Intercomparison among the test sites of the DITTY project using the IFREMER classification scheme for coastal lagoons*. EUR 21286 EN. European Commission - Directorate-General - Joint Research Centre - Institute for Environmental Sustainability.
- Bacher, C. 1989. "Capacité trophique du bassin de Marennes-Oléron: couplage d'un modèle de transport particulaire et d'un modèle de croissance de l'huître *Crassostrea gigas*". *Aquatic Living Resources*, 2 (4): 199-214.
- Bacher, C., Duarte, P., Ferreira, J. G., Héral, M., and Raillard, O. 1998. "Assessment and comparison of the Marennes-Oléron Bay (France) and Carlingford Lough (Ireland) carrying capacity with ecosystem models". *Aquatic Ecology*, 31 (4): 379-394.
- Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., Wijngaarden, A. v., and Woodger, M. 1960. "Report on the algorithmic language ALGOL 60". *Commun. ACM*, 3 (5): 299-314.
- Baray, C. 1998. "Effects of population size upon emergent group behavior". *Complexity International*, 6.
- Barciela, R. M., García, E., and Fernández, E. 1999. "Modelling primary production in a coastal embayment affected by upwelling using dynamic ecosystem models and artificial neural networks". *Ecological Modelling*, 120 (2-3): 199-211.
- Baretta, J. W., and Ruardij, P. 1988. *Tidal Flat Estuaries: Simulation and Analysis of the Ems Estuary*. 1 vols. Vol. 71, Springer-Verlag.
- Bax, N., and Eliassen, J.-E. 1990. "Multispecies analysis in Balsfjord, northern Norway: solution and sensitivity analysis of a simple ecosystem model". *ICES J. Mar. Sci.*, 47 (2): 175-204.

- Beckers, J. M., Rixen, M., Brasseur, P., Brankart, J. M., Elmoussaoui, A., Crépon, M., Herbaut, Martel, F., Van den Berghe, F., Mortier, L., Lascaratos, A., Drakopoulos, P., Korres, G., Nittis, K., Pinardi, N., Masetti, E., Castellari, S., Carini, P., Tintore, J., Alvarez, A., Monserrat, S., Parrilla, D., Vautard, R., and Speich, S. 2002. "Model intercomparison in the Mediterranean: MEDMEX simulations of the seasonal cycle". *Journal of Marine Systems*, 33-34: 215-251.
- Bellman, R. 1958. "Dynamic programming and stochastic control processes". *Information and Control*, 1 (3): 228-239.
- Bellman, R. E. 1957. *Dynamic Programming and Stochastic Control Processes*. RM-1904-PR.
- Bellman, R. E. 2003. *Dynamic Programming*. Princeton, University Press.
- Black, P. E. 1998. *Dictionary of Algorithms and Data Structures* (Sep 4, 1998). NIST, Mar 4, 2009 [cited Sep 4 2009]. Available from <http://www.itl.nist.gov/div897/sqg/dads/>.
- Blackmore, S. 2004. *Consciousness: An introduction*. New York, Oxford University Press.
- Blum, C., and Roli, A. 2003. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". *ACM Comput. Surv.*, 35 (3): 268-308.
- Blumberg, A. F., and Mellor, G. L. 1987. "A description of a coastal ocean circulation model". In *Three Dimensional Ocean Models*, N. S. Heaps (Eds.), vol. American Geophysical Union Inc., Washington, DC, 1-16.
- Bonabeau, E., Dorigo, M., and Theraulaz, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*, Oxford.
- Bonabeau, E. 2002. "Agent-based modeling: Methods and techniques for simulating human systems". *Proceedings of the National Academy of Sciences of the United States of America*, 99 (Suppl 3): 7280-7287.
- Bousquet, F., Cambier, C., and Morand, P. 1994. "Distributed artificial intelligence and object-oriented modelling of a fishery". *Mathematical and Computer Modelling*, 20 (8): 97-107.
- Bousquet, F., Bakam, I., Proton, H., and Le Page, C. 1998. "Cormas: Common-pool resources and multi-agent systems", vol., 826-837.
- Bousquet, F., and Le Page, C. 2004. "Multi-agent simulations and ecosystem management: a review". *Ecological Modelling*, 176 (3-4): 313-332.
- Brito, A. E. S. C., and Teixeira, J. M. F. 2001. *Simulação por Computador: Fundamentos e implementação de código em C e C++*. 1ª ed. Porto, Publindústria Edições Técnicas.
- Brock, T. D. 1981. "Calculating solar radiation for ecological studies". *Ecological Modelling*, 14 (1-2): 1-19.
- Brooks, R. A. 1991. "Intelligence without representation". *Artificial Intelligence*, 47: 139-159.

- Brosse, S., Guegan, J.-F., Tourenq, J.-N., and Lek, S. 1999. "The use of artificial neural networks to assess fish abundance and spatial occupancy in the littoral zone of a mesotrophic lake". *Ecological Modelling*, 120 (2-3): 299-311.
- Burke, E., Causmaecker, P. D., Petrovic, S., and Berghe, G. V. 2004. "Variable neighborhood search for nurse rostering problems". In *Metaheuristics: computer decision-making*, vol. Kluwer Academic Publishers, 153-172.
- Cardon, A., Galinho, T., and Vacher, J.-P. 2000. "Genetic algorithms using multi-objectives in a multi-agent system". *Robotics and Autonomous Systems*, 33 (2-3): 179-190.
- Chapelle, A., Bacher, C., Duarte, P., Fiandrino, A., Galbiati, L., Marinov, D., Martinez, J., Norro, A., Pereira, A., Plus, M., Rodriguez, S., Tsirtsis, G., and Zaldívar, J. M. 2005a. *Coastal lagoon modelling: An integrated approach*. EUR 21816 EN. European Commission - Directorate-General - Joint Research Centre - Institute for Environmental Sustainability.
- Chapelle, A., Duarte, P., Esteve, M. A., Fiandrino, A., Galbiati, L., Marinov, D., Martinez, J., Norro, A., Plus, M., Salles, C., Somma, F., Tournoud, M.-G., Tsirtsis, G., and Zaldívar, J.-M. 2005b. *Comparison Between Different Modelling Approaches for Coastal Lagoons*. EUR 21817 EN. European Commission - Directorate-General - Joint Research Centre - Institute for Environmental Sustainability.
- Chen, D.-G., and Hare, S. R. 2006. "Neural network and fuzzy logic models for pacific halibut recruitment analysis". *Ecological Modelling*, 195 (1-2): 11-19.
- Collins, R. J., and Jefferson, D. R. 1992. "AntFarm: Towards Simulated Evolution". In *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen (Eds.), vol. Addison-Wesley, 579-602.
- Cortés, U., Sánchez-Marrè, M., and Poch, M. 2000. "Guest Editorial: Artificial Intelligence and Environmental Applications". *Applied Intelligence*, 13 (1): 5-6.
- Crainic, T. G., and Toulouse, M. 2003. "Parallel Strategies for Meta-Heuristics". In *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger (Eds.), vol. Kluwer Academic Publishers, 475-513.
- Cruz, F., Pereira, A., Valente, P., Duarte, P., and Reis, L. P. 2007. "Intelligent Farmer Agent for Multi-agent Ecological Simulations Optimization". In *Proceedings of the 13th Portuguese Conference on Artificial Intelligence*, (Guimaraes, Portugal). 593-604.
- Dagorn, L., Menczer, F., Bach, P., and Olson, R. J. 2000. "Co-evolution of movement behaviours by tropical pelagic predatory fishes in response to prey environment: a simulation model". *Ecological Modelling*, 134 (2-3): 325-341.
- DeAngelis, D. L., and Gross, L. J., eds. 1992. *Individual-Based Models and Approaches in Ecology*. Chapman and Hall, London.
- Dorigo, M., Maniezzo, V., and Colorni, A. 1991. *Positive Feedback as a Search Strategy*. Technical Report 91-016. Dipartimento di Elettronica, Politecnico di Milano, Milano (June).

- Dorigo, M., Maniezzo, V., and Coloni, A. 1996. "Ant system: optimization by a colony of cooperating agents". *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26 (1): 29-41.
- Dorigo, M., and Gambardella, L. M. 1997. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". *IEEE Transactions on Evolutionary Computation*, 1 (1): 53-66.
- Dorigo, M., Caro, G. D., and Gambardella, L. M. 1999. "Ant algorithms for discrete optimization". *Artificial Life*, 5 (2): 137-172.
- Drogoul, A., and Ferber, J. 1992. "Multi-Agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies". In *Proceedings of the Artificial Social Systems, 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'92*, (S. Martino al Cimino, Italy). Springer, 3-23.
- Duarte, P., Meneses, R., Hawkins, A. J. S., Zhu, M., Fang, J., and Grant, J. 2003. "Mathematical modelling to assess the carrying capacity for multi-species culture within coastal waters". *Ecological Modelling*, 168 (1-2): 109-143.
- Duarte, P. 2005. "Photosynthesis-Irradiance Relationships in Marine Microalgae". In *Algal Cultures, Analogues of blooms and Applications*, D. V. S. Rao (Eds.), vol. 2. Science Publishers Enfield (NH), USA, Plymouth, UK, 639-670.
- Duarte, P., Azevedo, B., and Pereira, A. 2005a. *Hydrodynamic Modelling of Ria Formosa (South Coast of Portugal) with EcoDynamo*. University Fernando Pessoa, Porto (January).
- Duarte, P., Hawkins, A. J. S., and Pereira, A. 2005b. "How does estimation of environmental carrying capacity for bivalve culture depend upon spatial and temporal scales?". In *Comparative Roles of Suspension-Feeders in Ecosystems*, R. F. Dame and S. Olenin (Eds.). NATO Science Series IV Earth and Environmental Sciences, vol. 47, Nida, Lithuania, 121-135.
- Duarte, P., Azevedo, B., Ribeiro, C., Pereira, A., Falcão, M., Serpa, D., Bandeira, R., and Reia, J. 2007. "Management oriented mathematical modelling of Ria Formosa (South Portugal)". *Transitional Water Monographs*, 1 (1): 13-51.
- Duarte, P., Azevedo, B., Guerreiro, M., Ribeiro, C., Bandeira, R., Pereira, A., Falcão, M., Serpa, D., and Reia, J. 2008. "Biogeochemical modelling of Ria Formosa (South Portugal)". *Hydrobiologia*, 611: 115-132.
- Dumbauld, B. R., Ruesink, J. L., and Rumrill, S. S. 2009. "The ecological role of bivalve shellfish aquaculture in the estuarine environment: A review with application to oyster and clam culture in West Coast (USA) estuaries". *Aquaculture*, 290 (3-4): 196-223.
- Durfee, E. H., Lesser, V. R., and Corkill, D. D. 1989. "Trends in Cooperative Distributed Problem Solving". *IEEE Transactions on Knowledge and Data Engineering*, 1 (1): 63-83.
- Džeroski, S. 2001. "Applications of symbolic machine learning to ecological modelling". *Ecological Modelling*, 146 (1-3): 263-273.

- EC, E. C. 2000. Directive 2000/60/EC of the European Parliament and of the Council of 23 October 2000 establishing a framework for Community action in the field of water policy, edited by C. European Parliament: Official Journal L 327, 1-73.
- EC, E. C. 2003. Development of an Information Technology Tool for the Management of European Southern Lagoons under the influence of the river-basin runoff - the DITTY project, edited by J. R. Centre: Institute for Environment and Sustainability.
- Eglese, R. W. 1990. "Simulated annealing: A tool for operational research". *European Journal of Operational Research*, 46 (3): 271-281.
- Engelbart, D. C. 1962. *Augmenting Human Intellect: a Conceptual Framework*. AFOSR-3223. Air Force Office of Scientific Research, Washington DC (October 1962).
- Evans, B. M., Lehning, D. W., Corradini, K. J., Petersen, G. W., Nizeyimana, E., Hamlett, J. M., Robillard, P. D., and Day, R. L. 2002. "A Comprehensive GIS-based Modeling Approach for Predicting Nutrient Loads in Watersheds". *Journal of Spatial Hydrology*, 2 (2): 18.
- Falcão, M. 1996. Dinâmica dos nutrientes na Ria Formosa: efeitos da interacção da laguna com as interfaces na reciclagem do azoto, fósforo e sílica. PhD Thesis, University of Algarve, Faro.
- Falcão, M., Duarte, P., Matias, D., Joaquim, S., Fontes, T., and Meneses, R. 2000. *Gestão do cultivo de bivalves na Ria Formosa com recurso à modelação matemática*. Instituto de Investigação das Pescas e do Mar, Olhão, Portugal.
- Falcão, M., Fonseca, L., Serpa, D., Matias, D., Joaquim, S., Duarte, P., Pereira, A., Martins, C., and Guerreiro, M. J. 2003. *Synthesis Report - Ria Formosa*. INIAP/IPIMAR, CEMAS/UFP, Olhão, Portugal (September).
- Fasham, M. J. R., Ducklow, H. W., and McKelvie, S. M. 1990. "A nitrogen-based model of plankton dynamics in the oceanic mixed layer". *Journal of Marine Research*, 48 (3): 591-639.
- Fedra, K. 2005. *DITTY Project - DSS Evaluation Report*. DSS State-of-the-art Report. Environmental Software & Services GmbH.
- Feo, T. A., and Resende, M. G. C. 1989. "A probabilistic heuristic for a computationally difficult set covering problem". *Operations Research Letters*, 8 (2): 67-71.
- Feo, T. A., and Resende, M. G. C. 1995. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization*, 6 (2): 109-133.
- Ferber, J. 1999. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Reading, MA, Addison-Wesley.
- Ferreira, J. G. 1995. "ECOWIN - an object-oriented ecological model for aquatic ecosystems". *Ecological Modelling*, 79 (1-3): 21-34.
- Ferreira, J. G., Duarte, P., and Ball, B. 1997. "Trophic capacity of Carlingford Lough for oyster culture – analysis by ecological modelling". *Aquatic Ecology*, 31 (4): 361-378.

- Ferreira, J. G., Hawkins, A. J. S., Monteiro, P., Moore, H., Service, M., Pascoe, P. L., Ramos, L., and Sequeira, A. 2008. "Integrated assessment of ecosystem-scale carrying capacity in shellfish growing areas". *Aquaculture*, 275 (1-4): 138-151.
- Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McGuire, J., Shapiro, S., and Beck, C. 1993. *Specification of the KQML Agent-Communication Language*. The DARPA Knowledge Sharing Initiative External Interfaces Working Group, (1993/06/15).
- FIPA. 2001a. *FIPA ACL Agent Software Integration Specification*. XC00079B. Foundation for Intelligent Physical Agents, Geneva, Switzerland (2001/08/15).
- FIPA. 2001b. *FIPA ACL Ontology Service Specification*. XC00086D. Foundation for Intelligent Physical Agents, Geneva, Switzerland (2001/08/10).
- FIPA. 2002. *FIPA ACL Message Structure Specification*. SC00061G. Foundation for Intelligent Physical Agents, Geneva, Switzerland (2002/12/03).
- Franklin, S., and Graesser, A. 1997. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents". In *Intelligent Agents III. Agent Theories; Architectures, and Languages*, J. P. Muller, M. J. Wooldridge and N. R. Jennings (Eds.). Lecture Notes in Artificial Intelligence, vol. Springer-Verlag, 21-35.
- Fulton, E. A., Smith, A. D. M., and Johnson, C. R. 2004. "Effects of spatial resolution on the performance and interpretation of marine ecosystem models". *Ecological Modelling*, 176 (1-2): 27-42.
- Gertsev, V. I., and Gertseva, V. V. 2004. "Classification of mathematical models in ecology". *Ecological Modelling*, 178 (3-4): 329-334.
- Ghahramani, Z. 2004. "Unsupervised Learning". In *Advanced Lectures on Machine Learning - LNAI 3176*, O. Bousquet, G. Rätsch and U. v. Luxburg (Eds.). Lecture Notes in Artificial Intelligence, vol. Springer-Verlag, 72-112.
- Gilbert, N., and Troitzsch, K. G. 2005. *Simulation for the Social Scientist*. 2nd ed. New York, Open University Press.
- Glover, F. 1986. "Future paths for integer programming and links to artificial intelligence". *Computers & Operations Research*, 13 (5): 533-549.
- Glover, F. 1989. "Tabu search - Part I". *ORSA Journal on Computing*, 1: 190-206.
- Glover, F. 1990. "Tabu search - Part II". *ORSA Journal on Computing*, 2 (1): 4-32.
- Glover, F., and Laguna, M. 1997. *Tabu Search*, Kluwer Academic Publishers.
- Glover, F., Laguna, M., and Martí, R. 2000. "Fundamentals of Scatter Search and Path Relinking". *Control and Cybernetics*, 39 (3): 653-684.
- Glover, F., Laguna, M., and Martí, R. 2003. "Scatter Search". In *Advances in Evolutionary Computation: Theory and Applications*, A. Ghosh and S. Tsutsui (Eds.), vol. Springer-Verlag, New York, NY, USA, 519-537.

- Gorry, A., and Scott-Morton, M. 1971. "A Framework for Information Systems". *Sloan Management Review*, 13 (1, Fall 1971): 56-79.
- Haith, D. A., and Shoemaker, L. L. 1987. "Generalized Watershed Loading Functions for Stream Flow Nutrients". *Water Resources Bulletin*, 23 (3): 471-478.
- Hansen, P., and Jaumard, B. 1990. "Algorithms for the maximum satisfiability problem". *Computing*, 44 (4): 279-303.
- Hansen, P., and Mladenović, N. 1999. "An introduction to variable neighborhood search". In *Metaheuristics: Advances and trends in local search paradigms for optimization*, S. Voss, S. Martello, I. H. Osman and C. Roucairol (Eds.), vol. Kluwer Academic Publishers, 433-458.
- Hansen, P., Mladenović, N., Jaumard, B., and Parreira, A. D. 2000. *Variable neighbourhood search for maximum weighted satisfiability problem*. Technical Report G-2000-62. Les Cahiers du GERAD, Group for Research in Decision Analysis.
- Hansen, P., and Mladenović, N. 2001. "Variable neighborhood search: Principles and applications". *European Journal of Operational Research*, 130 (3): 449-467.
- Hansen, P., Mladenović, N., and Perez-Britos, D. 2001. "Variable Neighborhood Decomposition Search". *Journal of Heuristics*, 7 (4): 335-350.
- Hansen, P., and Mladenović, N. 2003. "Variable Neighborhood Search". In *Handbook of Heuristics*, F. Glover and G. A. Kochenberger (Eds.). International series in Operations Research & Management Science, vol. 57. Kluwer Academic Publishers, New York, 145-184.
- Hart, J. P., and Shogan, A. W. 1987. "Semi-greedy heuristics: An empirical study". *Operations Research Letters*, 6 (3): 107-114.
- Hart, P. E., Nilsson, N. J., and Raphael, B. 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *Systems Science and Cybernetics, IEEE Transactions on*, 4 (2): 100-107.
- Harvey, W. D., and Ginsberg, M. L. 1995. "Limited Discrepancy Search". In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI95*, (Montréal, Québec, Canada). Morgan Kaufmann, 607-613.
- Hawkins, A. J. S., Duarte, P., Fang, J. G., Pascoe, P. L., Zhang, J. H., Zhang, X. L., and Zhu, M. Y. 2002. "A functional model of responsive suspension-feeding and growth in bivalve shellfish, configured and validated for the scallop *Chlamys farreri* during culture in China". *Journal of Experimental Marine Biology and Ecology*, 281 (1-2): 13-40.
- Hermans, B. 1996. *Intelligent Software Agents on the Internet: an inventory of currently offered functionality in the information society & a prediction of (near-)future developments*. PhD Thesis, Tilburg University, Tilburg, The Netherlands.
- Hidrográfico, I. 2001. *Proj. OC4102/01, Ria Formosa, Relatório Técnico Final*. TF. OC 04/2001. Instituto Hidrográfico, Divisão de Oceanografia, Monitorização Ambiental.

- Hogeweg, P., and Hesper, B. 1983. "The ontogeny of the interaction structure in bumble bee colonies: A MIRROR model". *Behavioral Ecology and Sociobiology*, 12 (4): 271-283.
- Hogg, T., and Huberman, B. A. 1993. "Better Than The Best: The Power of Cooperation". In *1992 Lectures in omplex Systems*, vol. Addison-Wesley, 165-184.
- Hogg, T., and Williams, C. P. 1993. "Solving the Really Hard Problems with Cooperative Search". In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-93)*, (Washington, D.C.). AAAI Press, 231-236.
- Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, The University of Michigan Press.
- Hong, J. H., Kim, J. H., and Kim, T. G. 2007. Design and implementation of time management service for IEEE 1516 HLA/RTI. In *Proceedings of the 2007 summer computer simulation conference*. San Diego, California: Society for Computer Simulation International, 379-385.
- Hu, Y. H., Xue, Q., and Tompkins, W. J. 1991. "Structural simplification of a feed-forward, multilayer perceptron artificial neural network". In *Proceedings of the ICASSP-91 - IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Toronto, Canada). 1061-1064.
- Huhns, M. N., and Stephens, L. M. 1999. "Multiagent Systems and Societies of Agents". In *Multiagent Systems: A Modern Approach for Distributed Artificial Intelligence*, G. Weiss (Eds.), vol. The MIT Press, Cambridge, Massachusetts, 79-120.
- Huston, M., DeAngelis, D., and Post, W. 1988. "New Computer Models Unify Ecological Theory". *BioScience*, 38 (10): 682-691.
- Hutchinson, G. E. 1978. *An Intriduction to Population Ecology*. New Haven, Yale University Press.
- IEEE, 2000a. Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template. Std. IEEE 1516.2 - 2000.
- IEEE, 2000b. Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification. Std. IEEE 1516.1 - 2000.
- IEEE, 2000c. Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules. Std. IEEE 1516 - 2000.
- Iglesias, C., González, J., and Velasco, J. 1996. "MIX: A general purpose multiagent architecture". In *Intelligent Agents II Agent Theories, Architectures, and Languages*, M. Wooldridge, J. P. Müller and M. Tambe (Eds.). Lecture Notes in Computer Science, vol. 1037/1996. Springer Berlin / Heidelberg, 251-266.
- INE, I. N. d. E.-. 2008. *Statistical Yearbook of Portugal 2007*. 1st ed. 1 vols. Lisboa, Instituto Nacional de Estatística, IP.
- Jennings, N. R. 2000. "On agent-based software engineering". *Artificial Intelligence*, 117 (2): 277-296.

- Johnson, M. E. 1987. *Multivariate Statistical Simulation*. New York, John Wiley & Sons, Inc.
- Jones, M. T. 2005. *AI Application Programming*. Edited by A. I. D. processing. 2nd ed. 1 vols, Charles River Media.
- Jørgensen, S. E. 1992a. "Parameters, ecological constraints and exergy". *Ecological Modelling*, 62 (1-3): 163-170.
- Jørgensen, S. E. 1992b. "Development of models able to account for changes in species composition". *Ecological Modelling*, 62 (1-3): 195-208.
- Jørgensen, S. E., and Bendoricchio, G. 2001. *Fundamentals of Ecological Modelling*. 3rd ed. 1 vols, Elsevier.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. 1996. "Reinforcement Learning: A Survey". *Journal of Artificial Intelligence Research*, 4: 237-285.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. 1983. "Optimization by Simulated Annealing". *Science*, 220 (4598): 671-680.
- Knauss, J. A. 1997. *Introduction to Physical Oceanography*. 2nd ed, Prentice Hall.
- Kohler, T. A., Gumerman, G. J., and Reynolds, R. G. 2005. Simulating Ancient Societies. *Scientific American*, July.
- Kompare, B., and Džeroski, S. 1995. "Getting more out of data: Automated modelling of algal growth with machine learning". In *Proceedings of the International Conference on Coastal Ocean Space Utilization*, (University of Hawaii). 209-220.
- Kompare, B., Džeroski, S., and Karalič, A. 1997a. "Identification of the Lake of Bled ecosystem with the artificial intelligence tools M5 and FORS". In *Proceedings of the Fourth International Conference on Water Pollution*, (Southampton). Computational Mechanics Publications, 789-798.
- Kompare, B., Džeroski, S., and Križman, V. 1997b. "Modelling the growth of algae in the Lagoon of Venice with the artificial intelligence tool GoldHorn". In *Proceedings of the Fourth International Conference on Water Pollution*, (Southampton). Computational Mechanics Publications, 799-808.
- Kooijman, S. A. L. M. 1993. *Dynamic Energy and Mass Budgets in Biological Systems*. Cambridge, UK, Cambridge University Press.
- Kovačević-Vujčić, V. V., Čangalović, M. M., Ašić, M. D., Ivanović, L., and Dražić, M. 1999. "TABU search methodology in global optimization". *Computers & Mathematics with Applications*, 37 (4-5): 125-133.
- Kreft, J.-U., Booth, G., and Wimpenny, J. W. T. 1998. "BacSim, a simulator for individual-based modelling of bacterial colony growth". *Microbiology*, 144 (12): 3275-3287.

- Lansing, J. S., and Kremer, J. N. 1993. "Emergent Properties of Balinese Water Temple Networks: Coadaptation on a Rugged Fitness Landscape". *American Anthropologist*, 95 (1): 97-114.
- Law, A. M. 2007. *Simulation Modeling and Analysis*. Edited by K. E. Case and P. M. Wolfe. Fourth ed. New York, McGraw-Hill.
- Lawler, E. L., and Wood, D. E. 1966. "Branch-and-Bound Methods: A Survey". *OPERATIONS RESEARCH*, 14 (4): 699-719.
- Lazure, P., and Jegou, A.-M. 1998. "3D modelling of seasonal evolution of Loire and Gironde plumes on Biscay Bay continental shelf". *Oceanologica Acta*, 21 (2): 165-177.
- Lazure, P., and Dumas, F. 2008. "An external-internal mode coupling for a 3D hydrodynamical model for applications at regional scale (MARS)". *Advances in Water Resources*, 31 (2): 233-250.
- Lenhart, H. J., Radach, G., Backhaus, J. O., and Pohlmann, T. 1995. "Simulations of the north sea circulation, its variability, and its implementation as hydrodynamical forcing in ERSEM". *Netherlands Journal of Sea Research*, 33 (3-4): 271-299.
- Lippman, S. B., Lajoie, J., and Moo, B. E. 2005. *C++ Primer*. 4th ed. Boston, Addison-Wesley.
- Liu, J., and Ashton, P. S. 1998. "FORMOSAIC: an individual-based spatially explicit model for simulating forest dynamics in landscape mosaics". *Ecological Modelling*, 106 (2-3): 177-200.
- Lorek, H., and Sonnenschein, M. 1998. "Object-oriented support for modelling and simulation of individual-oriented ecological models". *Ecological Modelling*, 108 (1-3): 77-96.
- Lotka, A. J. 1956. *Elements of Mathematical Biology*. New York, Dover.
- Luyten, P. J., Jones, J. E., Proctor, R., Tabor, A., Tette, P., and Wild-Allen, K. 1999. *COHERENS - A coupled hydrodynamic-ecological model for regional and shelf seas: User Documentation*. MUMM Report, Management Unit of the Mathematical Models of the North Sea.
- Macal, C. M., and North, M. J. 2006. "Tutorial on Agent-Based Modeling and Simulation Part 2: How to Model with Agents". In *Proceedings of the Winter Simulation Conference 2006* ACM-IEE, 73-83.
- Maes, P. 1996. "Intelligent Software: Easing the Burdens that Computers Put on People". *IEEE Expert Systems*, 11 (6): 62-63.
- Malone, T. W., and Crowston, K. 1994. "The interdisciplinary study of coordination". *ACM Comput. Surv.*, 26 (1): 87-119.
- Manktelow, K. 2000. *Reasoning and Thinking*. Edited by G. Altmann and S. E. Gathercole, Psychology Press.
- MAOT. 2000. Plano de Bacia Hidrográfica das Ribeiras do Algarve - Caracterização Geral da Bacia Hidrográfica, 1ª Fase - Análise e Diagnóstico da Situação de Referência.

- MAOTDR. 2007. GIZC - Bases para a Estratégia da Gestão Integrada da Zona Costeira Nacional: MAOTDR.
- Mathevet, R., Bousquet, F., Le Page, C., and Antona, M. 2003. "Agent-based simulations of interactions between duck population, farming decisions and leasing of hunting rights in the Camargue (Southern France)". *Ecological Modelling*, 165 (2-3): 107-126.
- McDonald, M. G., and Harbaugh, A. W. 1984. *A modular three-dimensional finite-difference ground-water flow model*. U.S. Geological Survey.
- Mesplé, F., Troussellier, M., Casellas, C., and Legendre, P. 1996. "Evaluation of simple statistical criteria to qualify a simulation". *Ecological Modelling*, 88 (1-3): 9-18.
- Michalski, R. S., and Kodratoff, Y. 1990. "Research in Machine Learning; Recent Progress, Classification of Methods, and Future Directions". In *Machine Learning: An Artificial Intelligence Approach, Volume III*, R. S. Michalski and Y. Kodratoff (Eds.), vol. Morgan Kaufmann Publishers, Inc., San Mateo, California, 3-30.
- Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., eds. 1984. *Machine Learning: An Artificial Intelligence Approach*. Symbolic Computation. 1 vols. Springer-Verlag, Berlin.
- Minar, N., Burkhart, R., Langton, C. G., and Askenazi, M. 1996. *The swarm simulation system: a toolkit for building multiagent simulations*. Working Paper 96-06-042. Santa Fe Institute.
- Miranda, R., Braunschweig, F., Leitão, P., Neves, R., Martins, F., and Santos, A. 2000. "MOHID 2000 - A coastal integrated object oriented model". In *Hydraulic Engineering Software VIII*, W. R. Blain and C. A. Brebbia (Eds.). Water Studies, vol. WIT Press, 480.
- Mishra, N., Prakash, Tiwari, M. K., Shankar, R., and Chan, F. T. S. 2005. "Hybrid tabu-simulated annealing based approach to solve multi-constraint product mix decision problem". *Expert systems with applications*, 29 (2): 446-454.
- Mitchell, T. 1997. *Machine Learning*, McGraw Hill.
- Moreira, P. M. d. V. 2008. Intelligent Optimization Methodologies Applied to the Visualization of Virtual Environments. PhD Thesis, DEEC, Universidade do Porto, Porto.
- Neitsch, S. L., Arnold, J. G., Kiniry, J. R., Srinivasan, R., and Williams, J. R. 2002a. *Soil and Water Assessment Tool User's Manual - version 2000*. Grassland, Soil & Water Research Laboratory, Blackland Research and Extension Center, Temple, Texas (2002).
- Neitsch, S. L., Arnold, J. G., Kiniry, J. R., Williams, J. R., and King, K. W. 2002b. *Soil and Water Assessment Tool Theoretical Documentation - version 2000*. Grassland, Soil & Water Research Laboratory, Blackland Research and Extension Center, Temple, Texas (2002).
- Neves, R. J. J. 1985. Étude expérimentale et modélisation mathématique des circulations transitoire et résiduelle dans l'estuaire du Sado. PhD Thesis, Liège.

- Nobre, A. M., Ferreira, J. G., Newton, A., Simas, T., Icely, J. D., and Neves, R. 2005. "Management of coastal eutrophication: Integration of field data, ecosystem-scale simulations and screening models". *Journal of Marine Systems*, 56 (3-4): 375-390.
- Nunes, J. P., Ferreira, J. G., Gazeau, F., Lencart-Silva, J., Zhang, X. L., Zhu, M. Y., and Fang, J. G. 2003. "A model for sustainable management of shellfish polyculture in coastal bays". *Aquaculture*, 219 (1-4): 257-277.
- Odum, H. T. 1973. "An energy circuit language for ecological and social systems: its physical basis". In *Systems Analysis and Simulation in Ecology*, B. C. Patten (Eds.), vol. II. Academic Press, London, 139-211.
- Odum, H. T. 1983. *Systems Ecology: An Introduction*. Toronto, Wiley.
- Olsen, S. B., Page, G. G., and Ochoa, E. 2009. *The Analysis of Governance Responses to Ecosystem Change: A Handbook for Assembling a Baseline*. LOICZ Reports & Studies No.34. GKSS Research Center, Geesthacht.
- Osman, I. H., and Laporte, G. 1996. "Metaheuristics: A bibliography". *Annals of Operations Research*, 63: 513-623.
- Paz Betancourt, B., Franchesquin, N., Hervé, D., Rivière, G., and Treuil, J.-P. 1996. "Décisions Collectives, Contrats et Négociations: Une Expérience de Modélisation Multi-Agents d'un Contexte Agro-Pastoral dans les Andes Boliviennes". In *Proceedings of the Journées du Programme Environnement, Vie et Sociétés*, (Paris). CNRS, Centre National de la Recherche Scientifique, 61-66.
- Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, Massachussets, Addison-Wesley.
- Peregrine, D. H. 1967. "Long waves on a beach". *Journal of Fluid Mechanics Digital Archive*, 27 (4): 815-827.
- Pereira, A., Duarte, P., and Reis, L. P. 2004a. "Agent-Based Ecological Model Calibration - On the Edge of a New Approach". In *Proceedings of the International Conference on Knowledge Engineering and Decision Support*, (Porto, Portugal). ISEP, 107-113.
- Pereira, A., Duarte, P., and Reis, L. P. 2004b. "Agent-Based Simulation of Ecological Models". In *Proceedings of the 5th Workshop on Agent-Based Simulation*, (Lisbon, Portugal). SCS Publishing House, 135-140.
- Pereira, A., and Duarte, P. 2005. *EcoDynamo - Ecological Dynamics Model Application*. University Fernando Pessoa, Porto (January).
- Pereira, A., Duarte, P., and Reis, L. P. 2005. "ECOLANG - A communication language for simulations of complex ecological systems". In *Proceedings of the 19th European Conference on Modelling and Simulation (ECMS 2005)*, (Riga, Latvia). 493-500.
- Pereira, A., Duarte, P., and Norro, A. 2006. "Different modelling tools of aquatic ecosystems: A proposal for a unified approach". *Ecological Informatics*, 1 (4): 407-421.

- Pereira, A., Duarte, P., and Reis, L. P. 2007. "An Integrated Ecological Modelling and Decision Support Methodology". In *Proceedings of the 21st European Conference on Modelling and Simulation*, (Prague, Czech Republic). 497-502.
- Pereira, A. 2008. *ECOLANG - Communications Language for Ecological Simulations Network*. TR-LIACC-FEUP-AMCP 01.1. Dep. of Informatics Engineering, Faculty of Engineering - University of Porto, Porto (May).
- Pesant, G., and Gendreau, M. 1996. "A view of local search in constraint programming". In *Principles and Practice of Constraint Programming - CP96*. Lecture Notes in Computer Science, vol. 1118. Springer-Verlag, Berlin, 353-366.
- Pesant, G., and Gendreau, M. 1999. "A Constraint Programming Framework for Local Search Methods". *Journal of Heuristics*, 5 (3): 255-279.
- Phillips, N. A. 1957. "A coordinate system having some special advantages for numerical forecasting". *Journal of Meteorology*, 14: 184-185.
- Plus, M., Chapelle, A., Ménesguen, A., Deslous-Paoli, J.-M., and Auby, I. 2003. "Modelling seasonal dynamics of biomasses and nitrogen contents in a seagrass meadow (*Zostera noltii* Hornem.): application to the Thau lagoon (French Mediterranean coast)". *Ecological Modelling*, 161 (3): 213-238.
- Portela, L. I., and Neves, R. 1994. "Modelling temperature distribution in the shallow Tejo estuary". In *Proceedings of the Second European Conference on Advances in Water Resources Technology and Management*, (Lisbon, Portugal). EWRA, Balkema, Rotterdam, 457-463.
- Power, D. J. 2007. *A Brief History of Decision Support Systems* (DSSResources.COM) [World Wide Web], March 10, 2007 [cited June 2007]. Available from <http://dssresources.com/history/dsshhistory.html>.
- Quinlan, J. R. 1986. "Induction of Decision Trees". *Machine Learning*, 1: 81-106.
- Raillard, O., and Ménesguen, A. 1994. "An ecosystem box model for estimating the carrying capacity of a macrotidal shellfish system". *Marine Ecology Progress Series*, 115: 117-130.
- Ram, D. J., Sreenivas, T. H., and Subramaniam, K. G. 1996. "Parallel Simulated Annealing Algorithms". *Journal of Parallel and Distributed Computing*, 37 (2): 207-212.
- Realff, M. J., and Stephanopoulos, G. 1998. "On the Application of Explanation-Based Learning to Acquire Control Knowledge for Branch and Bound Algorithms". *INFORMS JOURNAL ON COMPUTING*, 10 (1): 56-71.
- Recknagel, F., French, M., Harkonen, P., and Yabunaka, K.-I. 1997. "Artificial neural network approach for modelling and prediction of algal blooms". *Ecological Modelling*, 96 (1-3): 11-28.
- Reeves, C. R. 1993. "Evaluation of heuristic performance". In *Modern heuristic techniques for combinatorial problems*, C. R. Reeves (Eds.), vol. John Wiley & Sons, Inc., 304-315.

- Reis, L. P., and Lau, N. 2002. "COACH UNILANG - A Standard Language for Coaching a (Robo)Soccer Team". In *Proceedings of the Robocup 2001 - Robot Soccer World Cup*, (Seattle, WA, USA). Springer-Verlag, 183-192.
- Reis, L. P. 2003. Coordenação em sistemas multi-agente: Aplicações na gestão universitária e futebol robótico. PhD Thesis, DEEC, Universidade do Porto, Porto.
- Resnick, M. 1996. "Beyond the Centralized Mindset". *Journal of the Learning Sciences*, 5 (1): 1 - 22.
- Reynolds, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*: ACM, 25-34.
- Rouchier, J., Bousquet, F., Requier-Desjardins, M., and Antona, M. 2001. "A multi-agent model for describing transhumance in North Cameroon: Comparison of different rationality to develop a routine". *Journal of Economic Dynamics and Control*, 25 (3-4): 527-559.
- Roughgarden, J. 1998. *Primer of Ecology Theory*. Upper Saddle River, New Jersey, Prentice-Hall.
- Russel, S. J., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach*. 2nd ed. New Jersey, Prentice Hall.
- Saaty, T. L. 1980. *The Analytic Hierarchy Process, Planning, priority Setting, Resource Allocation*. New York, McGraw-Hill.
- Salski, A., and Holsten, B. 2006. "A fuzzy and neuro-fuzzy approach to modelling cattle grazing on pastures with low stocking rates in Central Europe". *Ecological Informatics*, 1 (3): 269-276.
- Salski, A., and Holsten, B. 2009. "Fuzzy knowledge- and data-based models of damage to reeds by grazing of Greylag Geese". *Ecological Informatics*, 4 (3): 156-162.
- Sargent, R. G. 1991. "Simulation model verification and validation". In *Proceedings of the 1991 Winter Simulation Conference*, (Phoenix, Arizona, United States). IEEE Computer Society, 37-47.
- Scardi, M., and HardingJr, L. W. 1999. "Developing an empirical model of phytoplankton primary production: a neural network case study". *Ecological Modelling*, 120 (2-3): 213-223.
- Schleiter, I. M., Borchardt, D., Wagner, R., Dapper, T., Schmidt, K.-D., Schmidt, H.-H., and Werner, H. 1999. "Modelling water quality, bioindication and population dynamics in lotic ecosystems using neural networks". *Ecological Modelling*, 120 (2-3): 271-286.
- Scholten, H., and Tol, M. W. M. v. d. 1998. "Quantitative validation of deterministic models: when is a model acceptable?". In *Proceedings of the Summer Computer Simulation Conference - Simulation and Modeling Technology for the Twenty-First Century*, M. S. Obaidat, F. Davoli and D. DeMarinis (Eds.). Proceedings of the Summer Computer Simulation Conference, vol. Society of Computer Simulation, San Diego, 404-409.

- Scholten, H., Tol, M. W. M. v. d., and Smaal, A. C. 1998. "Models or measurements? Quantitative validation of an exophysiological model of mussel growth and reproduction". In *Proceedings of the ICES Annual Science Conference*, (Cascais, Portugal).
- Serment, J. 2007. Une infrastructure d'intégration à base d'agents logiciels pour l'élaboration de systèmes d'aide à la décision environnementale: Application à la gestion hydraulique camarguaise. PhD Thesis, Faculté des Sciences et Techniques, Université Paul Cézanne Aix - Marseille III, Marseille.
- Serpa, D. 2004. Macroalgal (*Enteromorpha* spp. and *Ulva* spp.) Primary Productivity in the Ria Formosa Lagoon. MSc Thesis, Universidade Nova de Lisboa.
- Shan, Y., Paull, D., and McKay, R. I. 2006. "Machine learning of poorly predictable ecological data". *Ecological Modelling*, 195 (1-2): 129-138.
- Shin, Y.-J., and Cury, P. 2001. "Exploring fish community dynamics through size-dependent trophic interactions using a spatialized individual-based model ". *Aquatic Living Resources*, 14 (2): 65-80.
- Siarry, P., Berthiau, G., Durdin, F., and Haussy, J. 1997. "Enhanced simulated annealing for globally minimizing functions of many-continuous variables". *ACM Transactions on Mathematical Software*, 23 (2): 209-228.
- Siena. 2005. *The Analytic Hierarchy Process in the architecture of the DSS*. WP8: DSS Development - Draft Report. Centre for Complex Systems Studies, University of Siena, Siena (2005-09-25).
- Siena. 2006. *Development of a Decision Support System for the Management of Southern European lagoons*. DITTY WP8 Draft Report. Centre for Complex Systems Studies, University of Siena, Siena.
- Skogen, M. D., and Moll, A. 2000. "Interannual variability of the North Sea primary production: comparison from two model studies". *Continental Shelf Research*, 20 (2): 129-151.
- Smith, M. B., Seo, D.-J., Koren, V. I., Reed, S. M., Zhang, Z., Duan, Q., Moreda, F., and Cong, S. 2004. "The distributed model intercomparison project (DMIP): motivation and experiment design". *Journal of Hydrology*, 298 (1-4): 4-26.
- Smith, R. D. (Nature Publisher Group), 1998a. Simulation Article. Nature Publisher Group, pp.
- Smith, R. D. 1998b. "Essential techniques for military modeling and simulation". In *Proceedings of the 1998 Winter Simulation Conference*, (Washington, D.C., United States). IEEE Computer Society Press, 805-812.
- Stroustrup, B. 2000. *The C++ Programming Language*. 3rd ed, Addison-Wesley Pub Co.
- Stützle, T., and Hoos, H. H. 1997. "MAX-MIN Ant System and local search for the traveling salesman problem". In *Proceedings of the IEEE International Conference on Evolutionary Computation*, 309-314.

- Stützle, T. 1998. Parallelization Strategies for Ant Colony Optimization. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, edited by A. E. Eiben, T. Bäck, M. Schoenauer and H.-P. Schwefel: Springer-Verlag, 722-731.
- Stützle, T., and Hoos, H. H. 2000. "MAX-MIN Ant System". *Future Generation Computer Systems*, 16 (8): 889-914.
- Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. 1 vols, The MIT Press.
- Taylor, A. H. 1993. "Modelling climatic interactions of the marine biota". In *Modelling oceanic climate interactions*, J. Willebrand and D. L. T. Anderson (Eds.). NATO ASI Series, Series I: Global Environmental Change, vol. I. Springer-Verlag, 373-413.
- Todorovski, L., Džeroski, S., and Kompare, B. 1998. "Modelling and prediction of phytoplankton growth with equation discovery". *Ecological Modelling*, 113 (1-3): 71-81.
- Turban, E. 1995. *Decision support and expert systems (4th ed.): management support systems*. Upper Saddle River, NJ, USA, Prentice-Hall, Inc.
- Turban, E., McLean, E. R., and Wetherbe, J. C. 2001. *Information Technology for Management: Making Connections for Strategic Advantage*. 3rd ed, John Wiley & Sons, Inc.
- van der Tol, M., and Scholten, H. 1997. "A model analysis on the effect of decreasing nutrient loads on the biomass of benthic suspension feeders in the Oosterschelde ecosystem (SW Netherlands)". *Aquatic Ecology*, 31 (4): 395-408.
- Vidal, J. M. 2007. *Fundamentals of Multiagent Systems with NetLogo Examples*: e-book.
- Volterra, V. 1926. "Fluctuations in the Abundance of a Species considered Mathematically". *Nature*, 118: 558-560.
- Vreugdenhil, C. B. 1989. *Computational Hydraulics: An Introduction*, Springer.
- Watson, R. T., Zinyowera, M. C., and Moss, R. H. 1996. *Climate Change 1995 - Impacts, adaptations and mitigation of climate change, Scientific-Technical Analyses*. Edited by I.-I. P. o. C. Change. Vol. 1. Cambridge, UK, Cambridge University Press.
- Wegner, D. M. 2003. *The Illusion of Conscious Will*. Cambridge, MA, MIT Press.
- Weiss, G., ed. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, Massachusetts.
- Weiss, M. A. 2000. *Data Structures and Problem Solving Using C++*. 2nd ed, Addison-Wesley.
- Westphal, K. 2000. *Decision Support System for Water Resources Management*. CDM [cited 15 September 2009. Available from http://www.cdm.com/knowledge_center/interview/decision_support_systems_for_water_resources_management.htm.

- Wiegert, R. G. 1979. "Population Models: experimental tools for the analysis of ecosystems". In *Proceedings of the Colloquium on Analysis of Ecosystems* Ohio State University Press, 239-275.
- Wolfram, S. 1984. "Cellular automata as models of complexity". *Nature*, 311 (5985): 419-424.
- Wooldridge, M., and Jennings, N. R. 1995. "Intelligent Agents: Theory and Practice". *Knowledge Engineering Review*, 10: 115-152.
- Wooldridge, M. 1999. "Intelligent Agents". In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss (Eds.), vol. The MIT Press, Cambridge, Massachusetts, 27-77.
- Wooldridge, M. 2002. *An Introduction to MultiAgent Systems*. 1 vols. West Sussex, England, John Wiley & Sons Ltd.
- Yagiura, M., and Ibaraki, T. 2001. "On metaheuristic algorithms for combinatorial optimization problems". *Systems and Computers in Japan*, 32 (3): 33-55.
- Youssef, H., Sait, S. M., and Adiche, H. 2001. "Evolutionary algorithms, simulated annealing and tabu search: a comparative study". *Engineering Applications of Artificial Intelligence*, 14 (2): 167-181.
- Zadeh, L. A. 1965. "Fuzzy Sets". *Information and Control*, 8: 338-353.
- Zimmermann, H. J. 1987. *Fuzzy Sets, Decision Making, and Expert Systems*. Edited by J. P. Ignizio. Boston, Kluwer Academic Publishers.

Annexes

Annex 1 – EcoDynamo User’s Manual

The complete user’s manual of EcoDynamo is accessible in the help section of the application and a copy of it is shown in this section.

What is EcoDynamo?

EcoDynamo (Ecological Dynamics Model) is an application built to enable physical and biogeochemical simulation processes of aquatic ecosystems. It is an object oriented program application, written in C++ language, with a shell that manages the graphical user interface, the communications between classes and the output devices where the simulation results are saved.

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: definitions as morphology, geometric representation of the model, dimensions, number of cells, classes, variables, parameter initial values and ranges are present.

The user can choose between file, chart or table to store the simulation results. These output formats are compatible with some commercial software (like MatLab®) products, enabling their posterior treatment.

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.

Simulation runs can be traced in real time, observing the threads activity and the inter-classes messages, or analyzed a posteriori with the help of log files, activated previously before the simulation run.

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

The messages interchanged with other applications can be monitored activating the trace of protocol messages.

Command Line Arguments

EcoDynamo can be launched from the command line enabling the customization of some runtime values. The usage is:

EcoDynamo [options]

Where the options are:

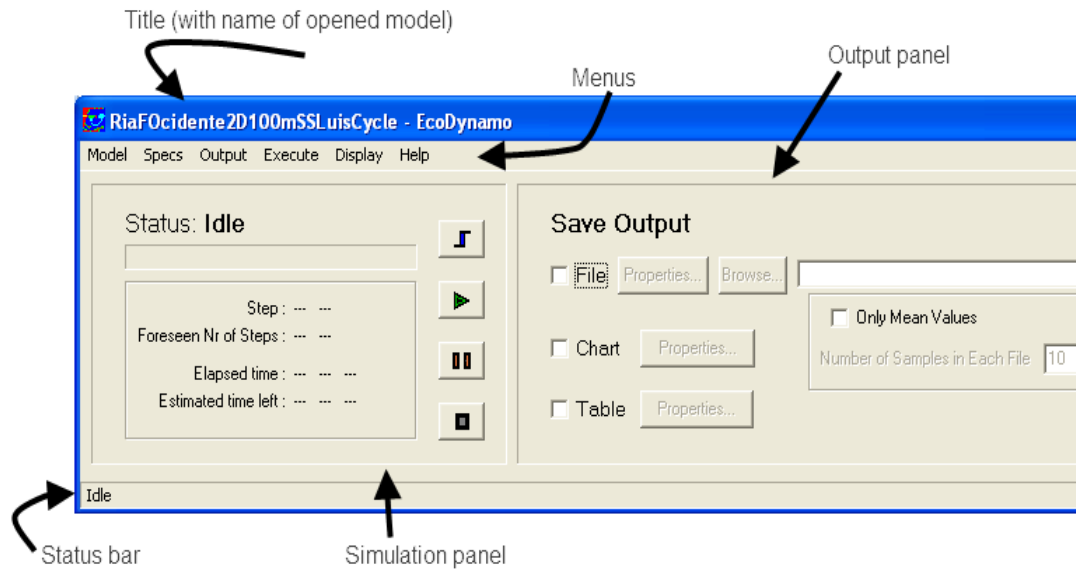
| Option | Significance | Default value |
|------------------------------------|--|---------------|
| -port <portNumber> | Listen port for ECOLANG messages (TCP/IP) | 45000 |
| -name <simulatorName> | Simulator name for EcoSimNet framework – name without spaces | EcoDynamo |
| -recover | Recover last model configuration from “LastConfiguration” folder | |

When the simulator is launched by double clicking the executable file, the default values are assumed. The easiest way to change default values is to define a batch file with the command line filled with the new values, and run the batch file. Example of a batch file:

EcoDynamo -name EcoDynamo_2 -port 46000 -recover

Main Window

The main window of EcoDynamo is the privileged interface for the user to manipulate the models simulation.



When a model is opened its name is added to the name of the application in the title area.

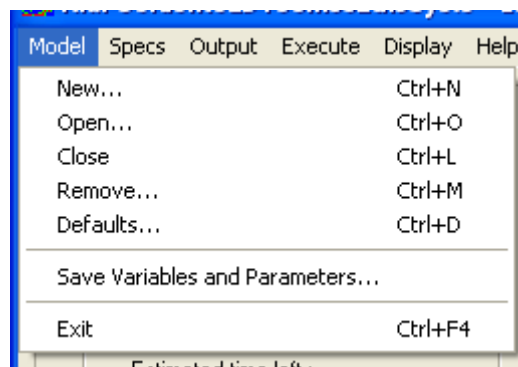
The Main Window has 5 principal areas: Menus, Simulation Panel (also called Execute Panel), Output Panel, Communications Area and Status Bar.

The Communications Area is hidden when the application starts and becomes visible when options of the Display Menu are activated.

Model Menu

The Model Menu is used to define new models, open and remove existing models, close the working model and set the default model when the application starts. There is also an option to save the model configuration with the current values of variables and parameters.

The application exit command is also present in this menu.



New... - define a new model with proper Configuration Files.

Open... - open a defined model.

Close - close the working model.

Remove... - remove an existing model.

Defaults... - set the default model for the application.

Save Variables and Parameters... - save the current model configuration.

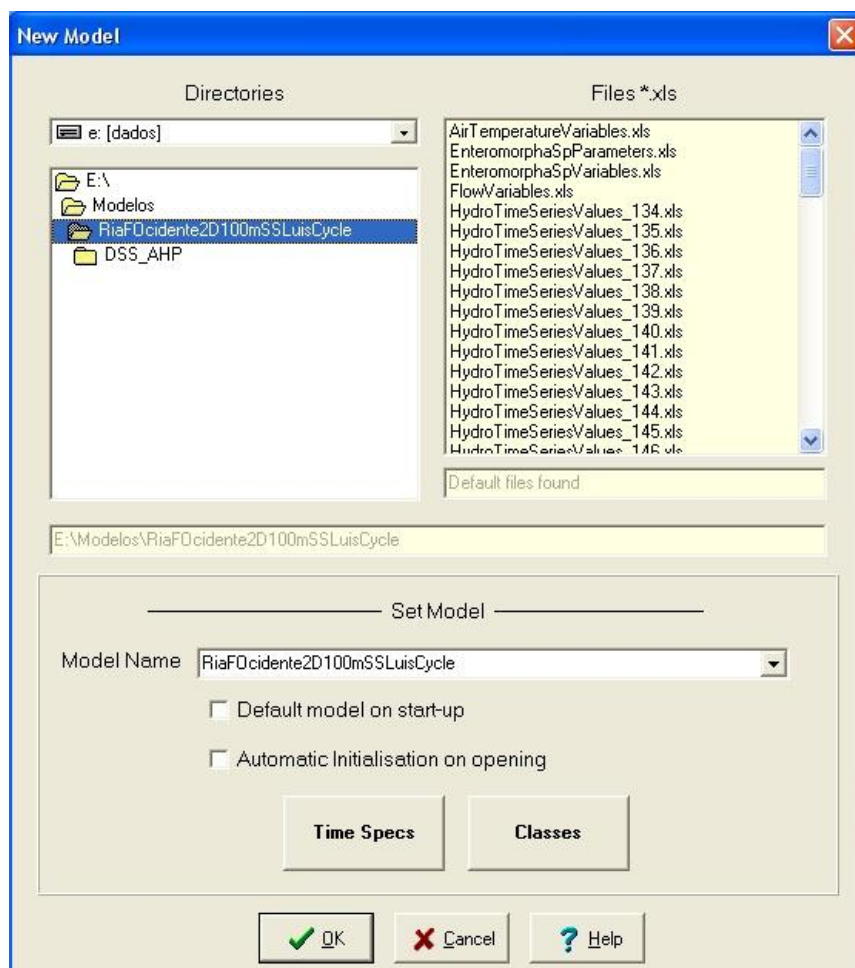
Exit - exit EcoDynamo application.

New Model

The New submenu opens a dialog where the user defines a new model for simulation.

In this dialog the user must browse for the directory where the model is defined (Configuration Files present), supply a name for the model, define if the model will be the default model on start-up of the EcoDynamo application and if the model is automatically initialized on opening.

When all the mandatory Configuration Files are present in the browsed directory, the dialog looks like the following figure (if some configuration file misses, background will be gray):



The user can select the Classes, variables and Time specs for the simulation.

After click OK button one entry is added to the Model Properties file.

Open Model

The Open submenu opens a dialog where the user selects one already created model for simulation (saved in Model Properties file).

If the model was defined as automatically initialized after opened, it is initialized after the click on the OK button.



If one model was in use the EcoDynamo Properties file is saved in the model directory with the simulation properties (Time specs, Classes and Variables).

Close Model

The Close menu closes the working model.

The EcoDynamo Properties file, with the simulation properties, is saved in the model directory.

Remove Model

The Remove submenu opens a dialog where the user selects one already created model for deletion.

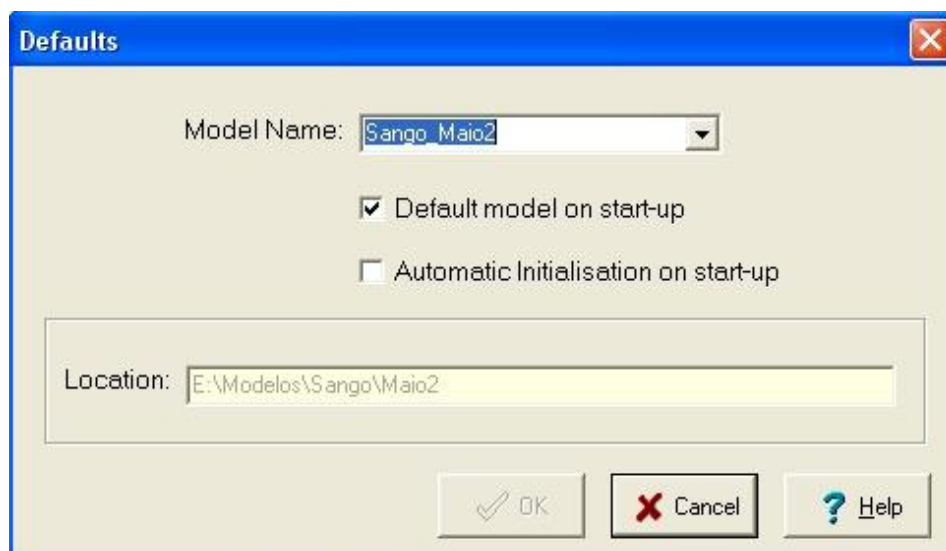
The corresponding entry in the Model Properties file is removed.



Defaults Model

The Defaults submenu opens a dialog where the user selects the default behaviour for each model and the default model on start-up.

The corresponding entries in the Model Properties file are changed.

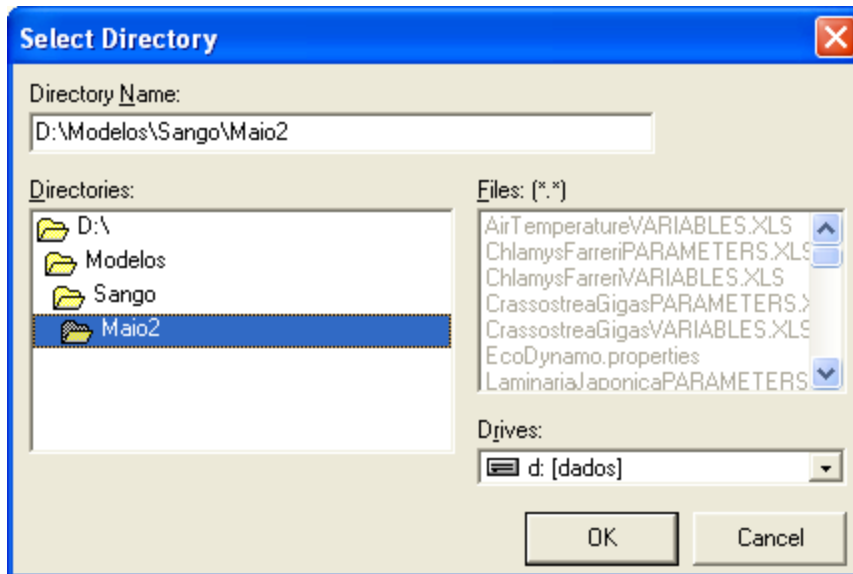


For the default model on start-up the background color of the Location textbox is changed to smooth yellow.

Save Model Configuration

The Save Variables and Parameters option provides the facility of saving the model configuration, with the current values of variables and parameters, time step and classes selected.

The user should define the folder in which the new model database will be saved.



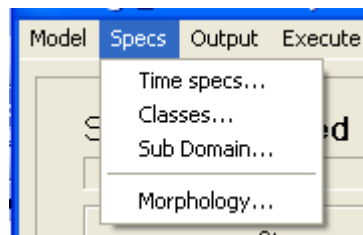
Exit

This options exits the EcoDynamo application.

If one model is in use the corresponding EcoDynamo Properties file is saved.

Specs Menu

The Specs Menu is used to define the simulation options for the model run, like time specs, classes selected, variables to output and sub domain of the model.



Time specs... - defines time specs for simulation and output register.

Classes... - selects Classes for simulation and Variables for output.

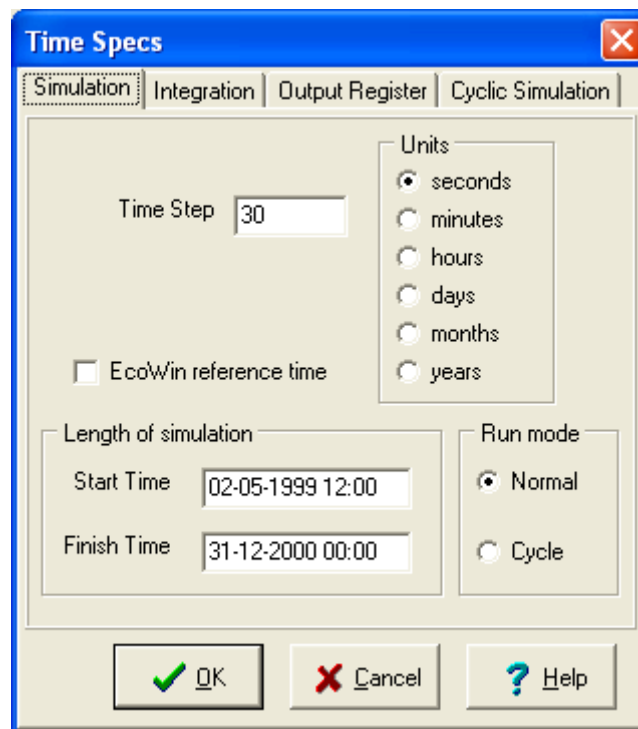
Sub Domain... - selects domain for simulation.

Morphology... - checks morphology values

Time specs

The Time Specs submenu opens a dialog where the user defines the time specifications, for simulation and output, in different tabs. The user can also specify the configuration of cyclic simulations:

The Simulation tab select the simulation time step and duration:



If the checkbox EcoWin reference time is selected the time for each time step is based on the EcoWin® reference time.

The Run mode option is linked with cyclic simulation, but it is not completely integrated with the facilities developed in the "Cyclic Simulation" tab.

The other options of this menu are:

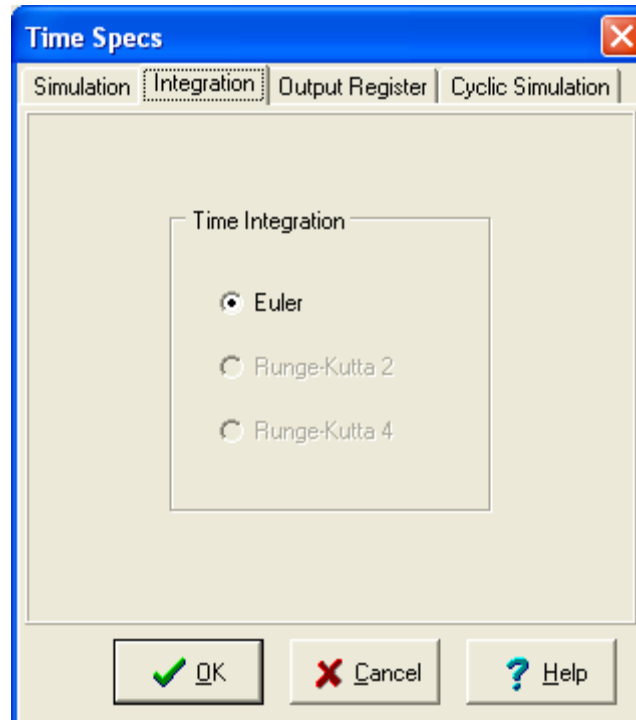
Integration tab - selects type of time integration

Output Register tab - defines output register frequency

Cyclic Simulation tab - activates cyclic simulation and define number of periods

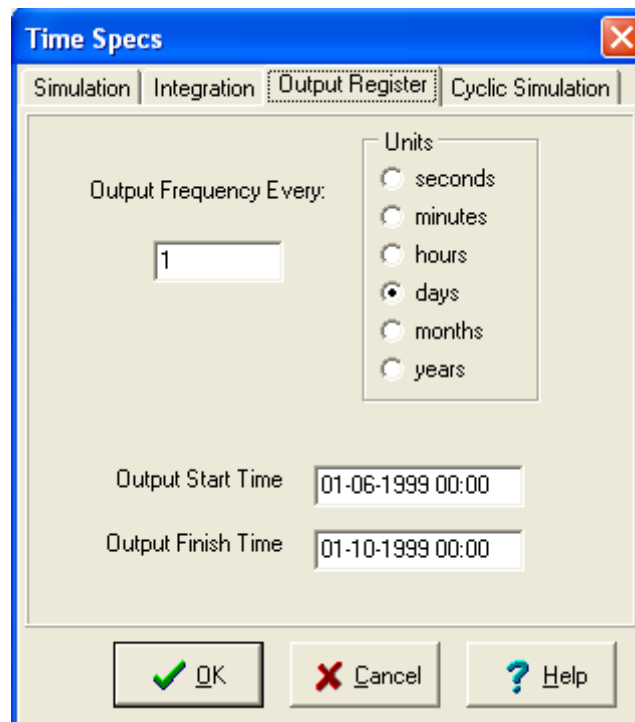
Integration

The Integration tab only allows the "Euler" integration. The two types of Runge-Kutta time integration will be implemented in future releases.



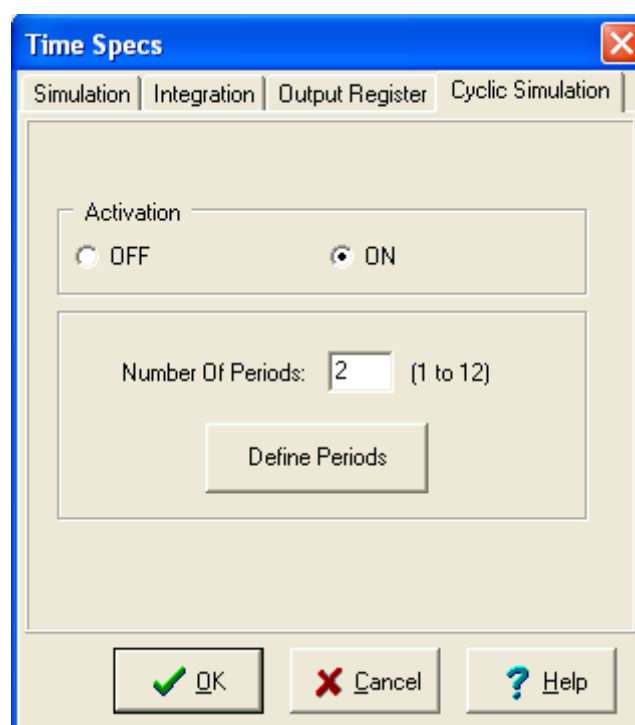
Output Time

The Output Register tab select the output register frequency and the output start and finish time:



Cyclic Simulation

The Cyclic Simulation tab enables the activation of cyclic simulations and the definition of the number of cyclic periods:



Pressing the "Define Periods" button will open the Periodic Simulations Configuration window.

Periodic Simulations Definitions

In this window the user can configure the periods of the periodic simulations. The number of periods is configured in the Cyclic Simulation tab of the Time specs window.

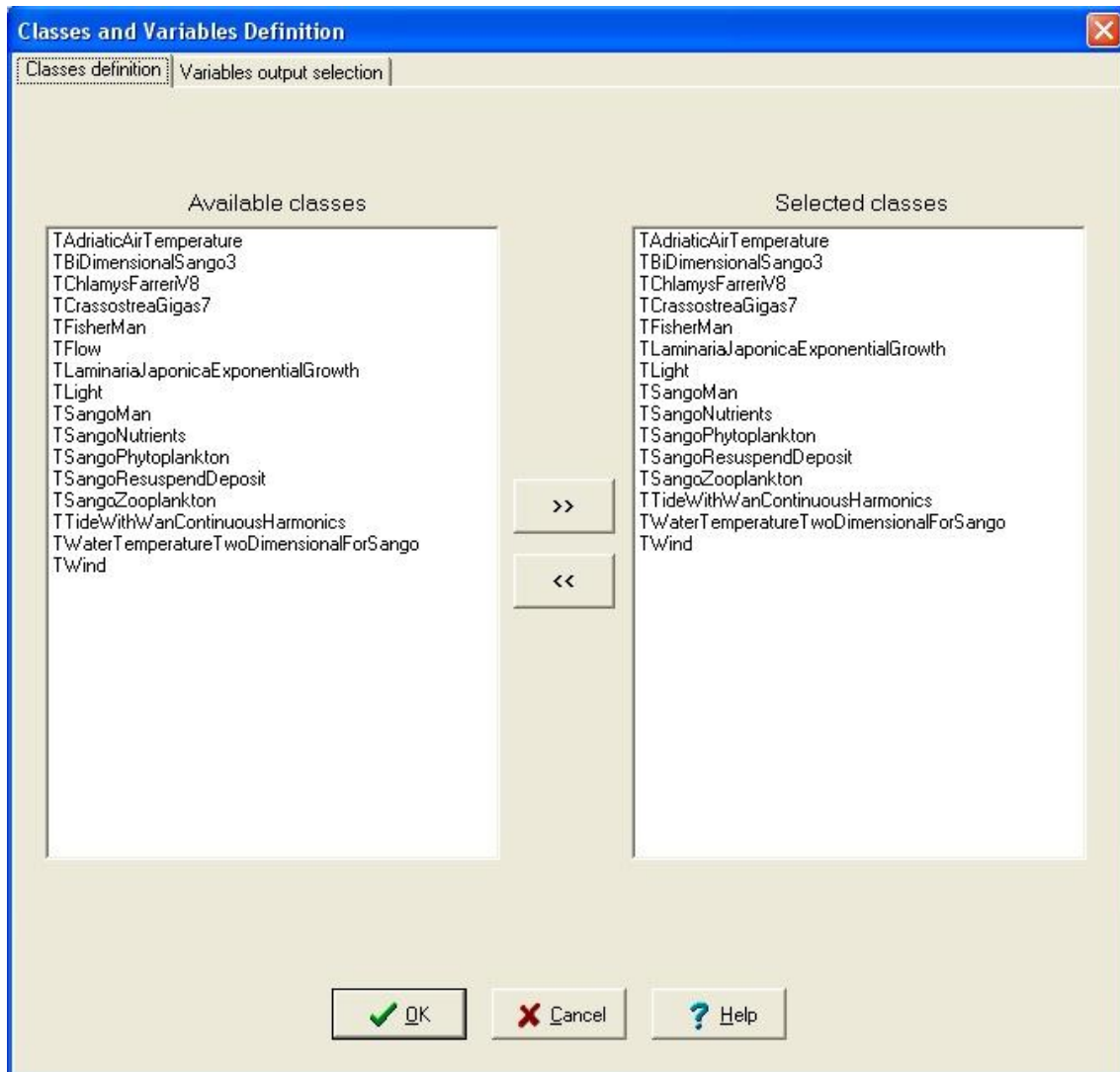
In each line the fields Start Time and Finish Time refer the simulation time, the Cycle and Time Unit fields refer the cycle duration and the First File Index field refers the name of the first file that must be considered to read the values of the hydrodynamic variables. The file names start by "HydroTimeSeriesValues_" and are appended by the file index (description in Mean Values Files

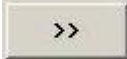
| | Start Time | Finish Time | Cycle | Time Unit | First File Index |
|-----------------|------------------|------------------|---------|-------------|------------------|
| Period 1 | 06-01-2002 03:08 | 31-12-2002 00:00 | 1252200 | second | 144 |
| Period 2 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 3 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 4 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 5 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 6 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 7 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 8 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 9 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 10 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 11 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |
| Period 12 | 01-01-1970 00:00 | 01-01-1970 00:00 | 0 | -- -- -- -- | 0 |

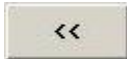
Classes

The Classes submenu opens a dialog where the user select the classes for simulation and the variables to output results in different tabs.

The Classes definition tab selects the classes for simulation:



The  button select classes for simulation (transfers the classes marked in the "Available classes" list to the "Selected classes" list).

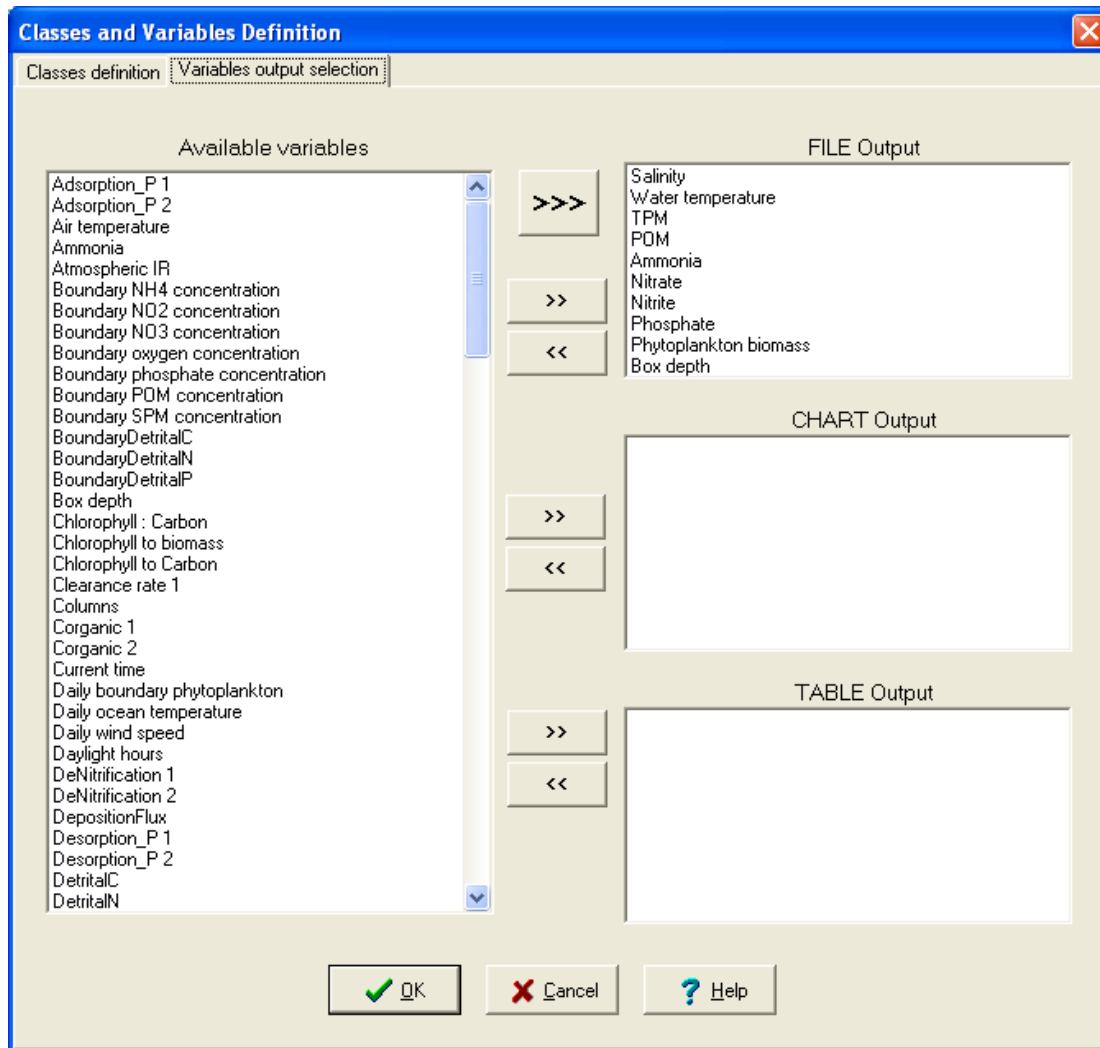
The  button deselect the classes marked in the "Selected classes" list.


In the "Variables output selection" tabbed panel the user selects the variables for output register.

Variables


From the Classes submenu the user selects the variables for output results.

The tab for variables output selection is:




The  button transfers the selected variables in the "Available variables" area to all output options.



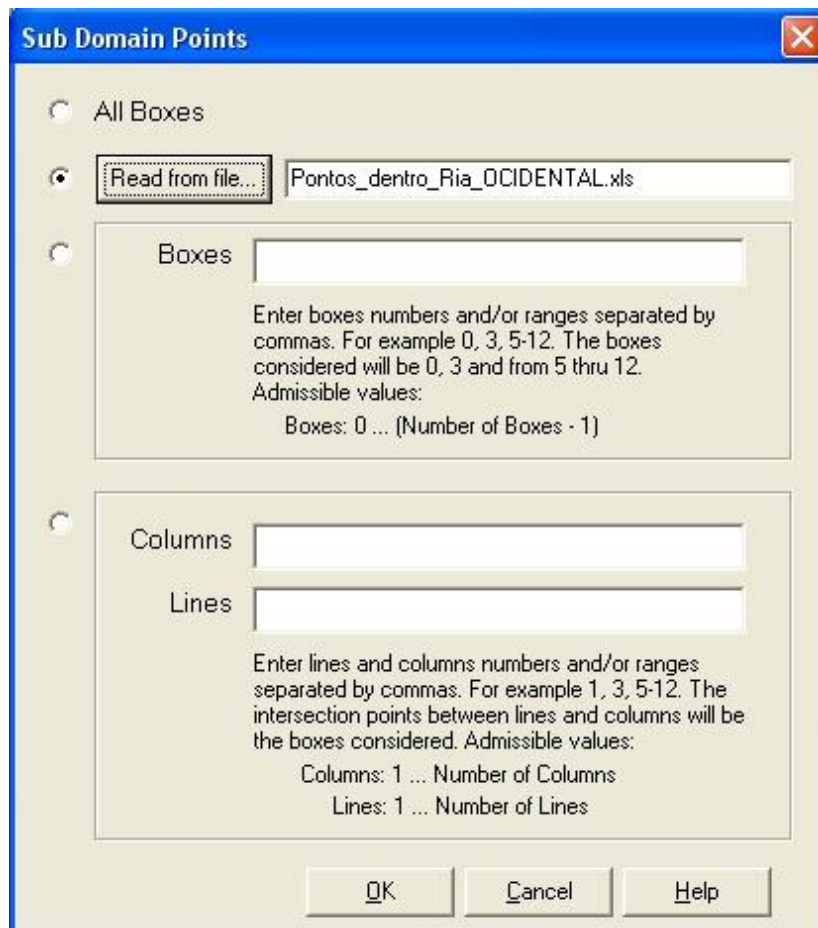
Each  button copies the variables selected in "Available variables" list to the related output area.



Each  button removes the variables selected from the corresponding output area.

Sub Domain

The Sub domain submenu opens a dialog where the user select the domain for the simulation:

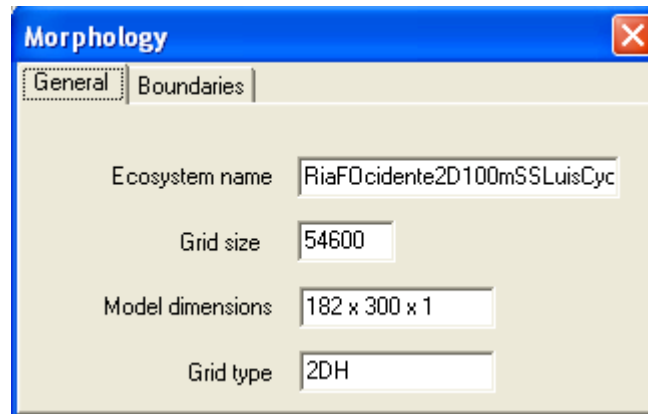


The different exclusive options for points selection are:

- All Boxes - complete domain.
- Read from file... - read sub domain points from a file. Button click opens a file dialog enabling the user to browse the file's location. Files must have the format described in Points File to be considered.
- Boxes - select only the boxes listed in the textbox.
- Columns and Lines - the boxes selected are the intersection of the line and column numbers listed in the textboxes.

Morphology

This dialog shows the model's general morphology - name, grid size and type, number of lines, number of columns and number of layers.



The information of this panel is filled only after the model initialisation.

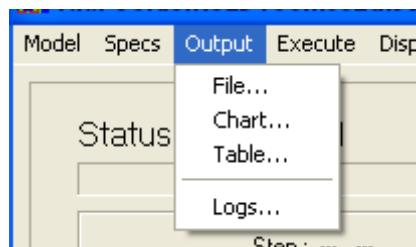
The tab Boundaries is not used in this version of EcoDynamo.

Output Menu

The Output Menu is used to select the options for register the model output results.

The user can choose different output devices like file, chart and table.

It is also possible to log the messages exchanged between the different classes and at different simulation steps.



File... - select output file, variables and points.

Chart... - select output charts, variables and points.

Table... - select output table, variables and points.

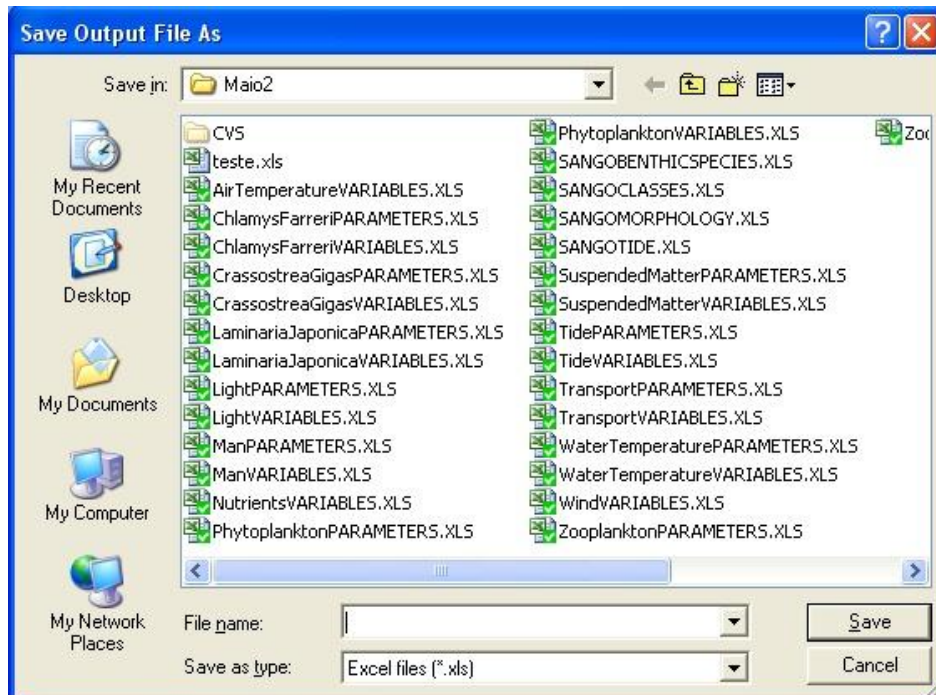
Logs... - select log simulation steps and file type

The selection of the output options can be done in the Output Panel

Output File

The File submenu selects the file and the boxes for register the simulation variables output results in two steps.

Step 1 - Select file name



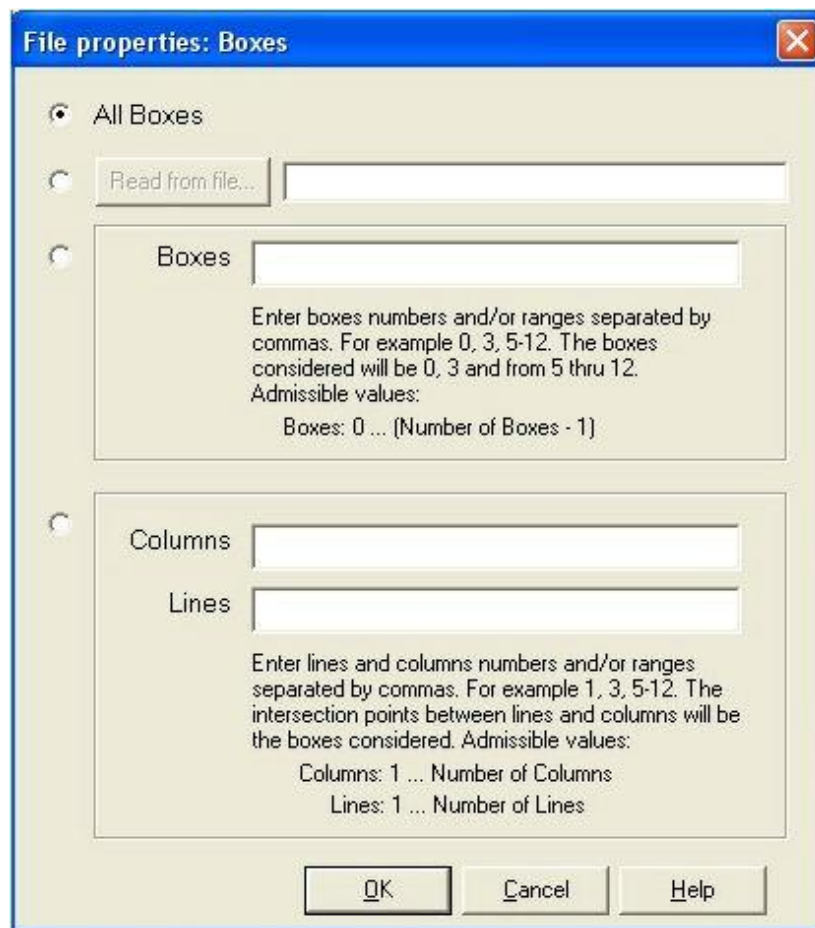
The file types can be ".xls", ".hdf" or ".txt".

The xls and txt files are formatted with Tab Separated Values.

The ".hdf" format follows the HDF specifications - see NCSA http server (<http://hdf.ncsa.uiuc.edu>)

Step 2 - Select points to register

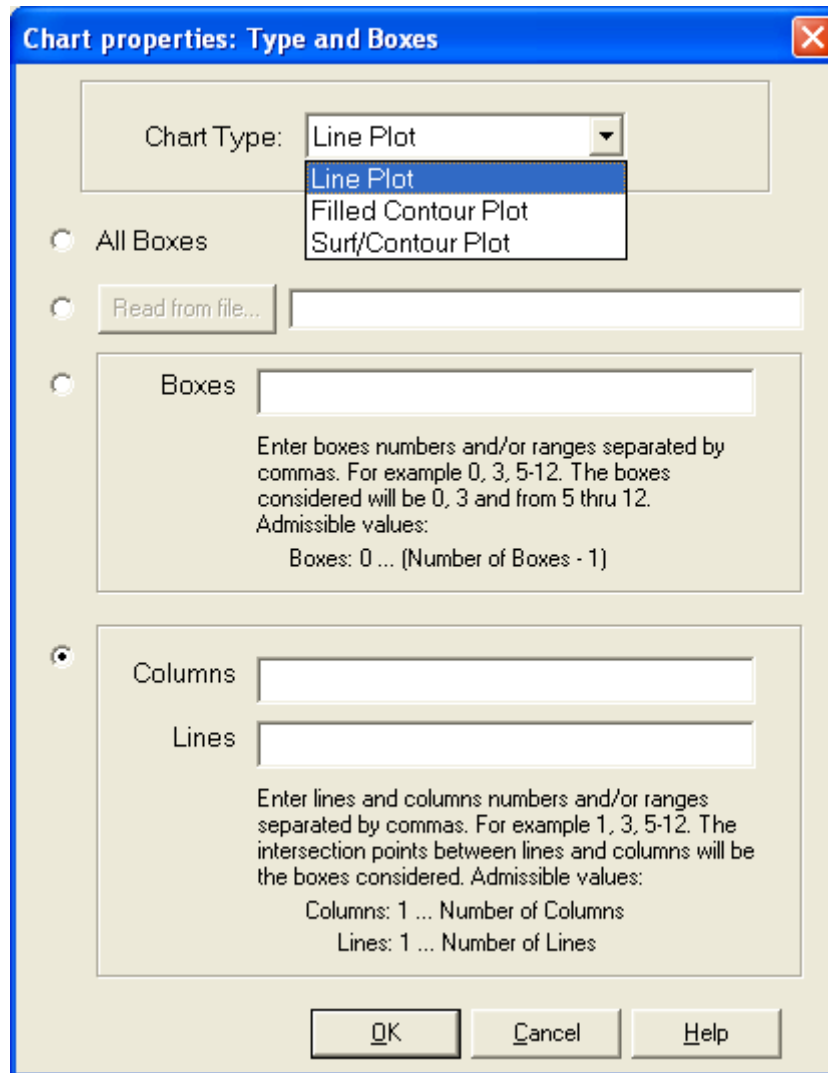
After entering the filename and click on the "Save" button, one dialog (similar to Sub Domain) appears to select the register points:



The name of the file chosen will be displayed in the Output Panel.

Output Chart

The Chart submenu selects the boxes for register the simulation variables output results in chart format and the type of chart. One dialog (similar to Sub Domain) appears for select the register points:

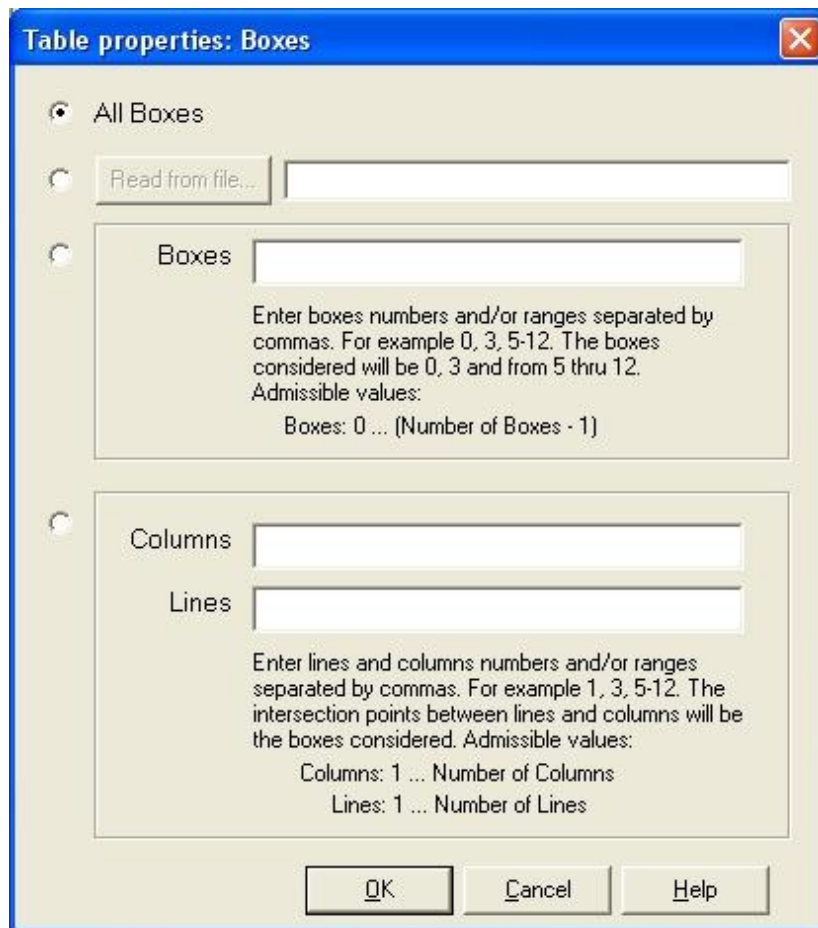


The chart types available are Line Plot, Filled Contour Plot and Surf/Contour Plot.

To create the output charts, one for each variable, the EcoDynamo launches the MatLab® application and benefits from all the features that MatLab® supplies to manipulate the chart images.

Output Table

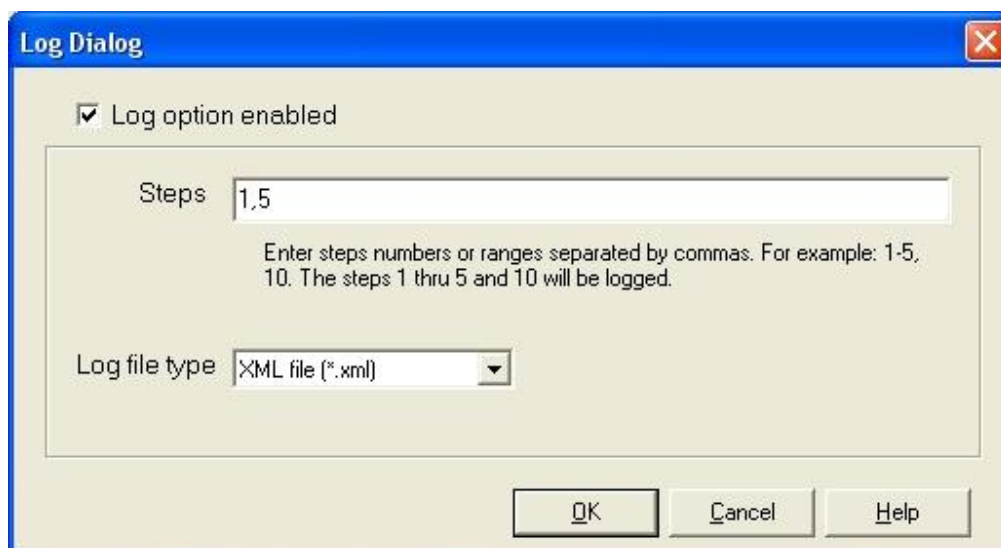
The Table submenu selects the boxes for register the simulation variables output results in table format. One dialog (similar to Sub Domain) appears for select the register points:



The table output format is not implemented in this version.

Output Logs

The Logs submenu presents a dialog to specify the steps where the messages exchanged between the different classes will be logged:



The log file format can be specified by the user within "xml", "xls" and "txt".

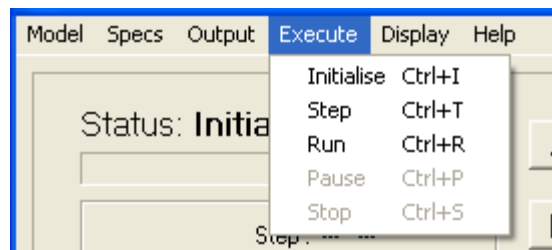
The xls and txt formats are saved with Tab Separated Values.

The xml format is more elaborated and is used for more sophisticated pos-processing methods.

The format of the file behind each type is described in Log File.

Execute Menu

The Execute Menu controls the simulation runs.



Initialise - initialise model for simulation.

Step - run the simulation one step.

Run - run the simulation.

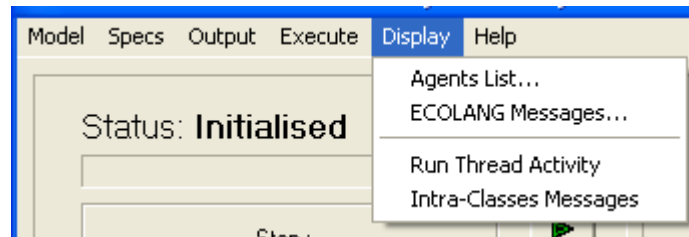
Pause - pause the simulation.

Stop - stop/end the simulation.

The control of the simulation runs can be done in the Simulation Panel

Layout Display Menu

The actions of the Display Menu enable the user to see information related with simulation execution (trace simulation) and to display information related with the communications between the EcoDynamo and agents, behind the EcoDynamo protocol, and the list of agents known by the EcoDynamo.



The Agents List option opens one frame with a table of agents known by the EcoDynamo.

The ECOLANG Messages option opens one frame that enables the trace of the received and transmitted messages between EcoDynamo and agents.

The Run Thread Activity and Intra-Classes Messages expand the main window of the EcoDynamo in order to include two areas with the trace to simulation activity.

Agents List

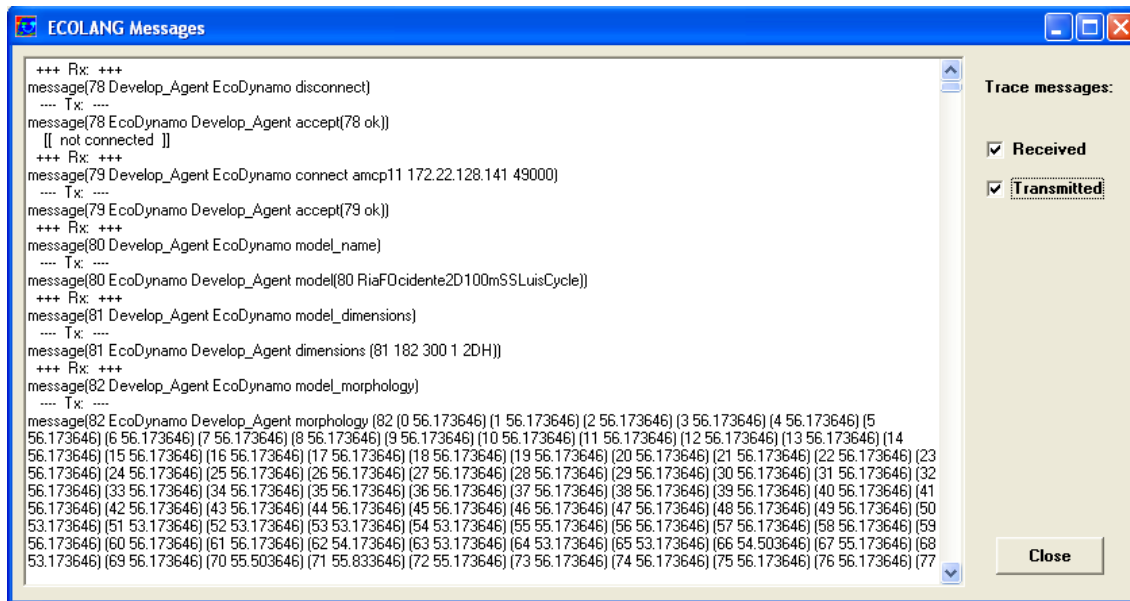
This frame shows the list of agents known by the EcoDynamo. The columns of the table represent the index of the agent entry in the list of agents, the identifier of the agent, the agent name, the name and the address of the host computer of the agent, the port number where the agent listens for connections and the connection state.

| List of Known Agents | | | | | | |
|----------------------|----|---------------|-----------|----------------|-------|-----------|
| Index | ID | Agent Name | Host Name | Host Address | Port | Connected |
| 0 | 0 | EcoDynamo | amcp11 | 172.22.128.141 | 45000 | true |
| 1 | 1 | Develop_Agent | amcp11 | 172.22.128.141 | 49000 | true |

Close

ECOLANG Messages

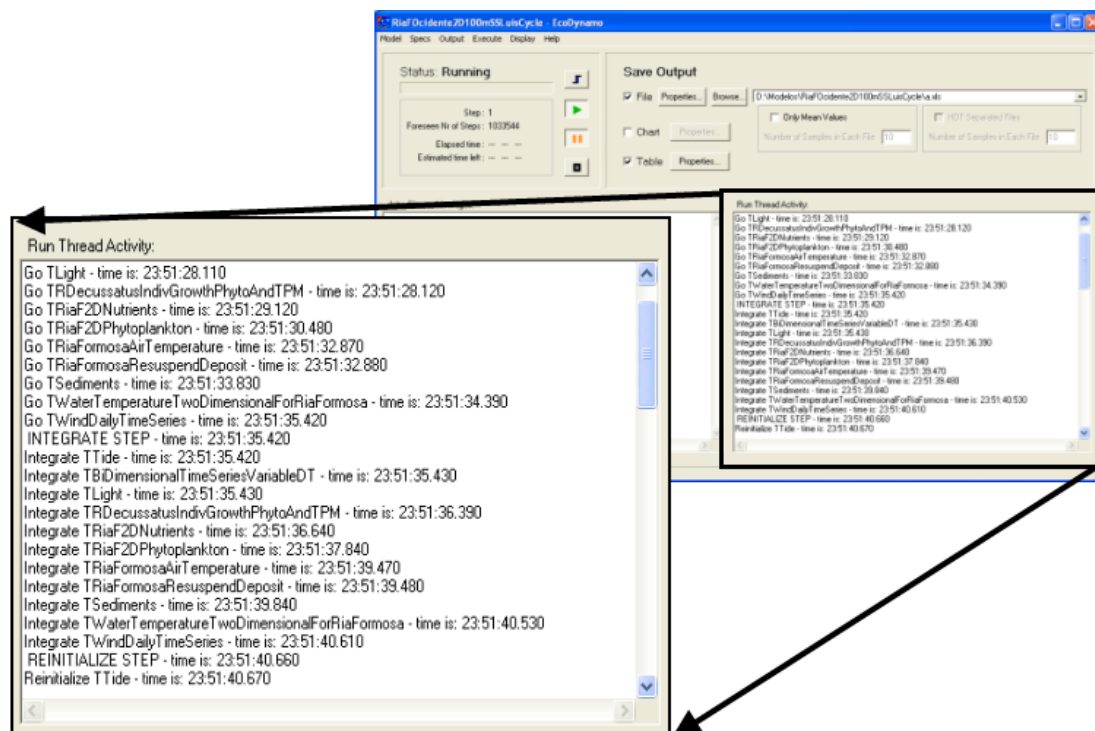
This frame displays the ECOLANG messages exchanged between EcoDynamo and the agents connected to it. The user can select the trace to the received and / or transmitted messages.



The messages received are headed by +++ Rx +++ and the messages transmitted are headed by --- Tx ---, respectively.

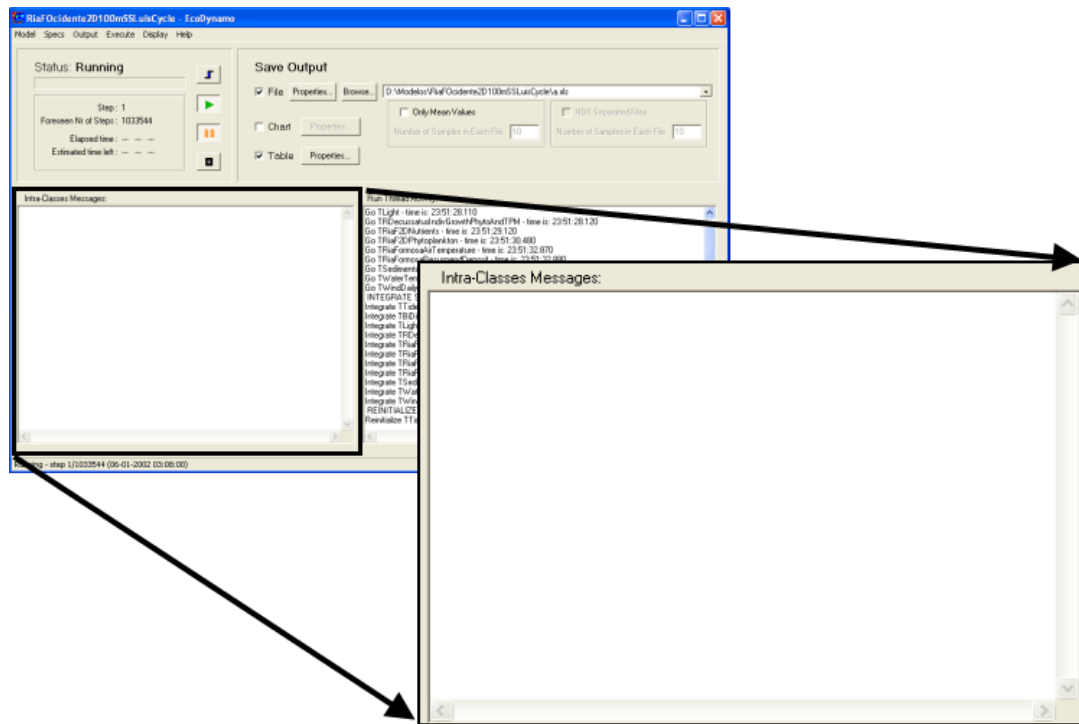
Run Thread Activity

The run thread activity is shown in the Main Window, that is expanded when the option Run Thread Activity is selected in the Display menu. When toggled on, the main window creates one text area to display the activity of the Run Thread, namely the calls to the simulation functions in each class (Go, Integrate, Reinitialize).



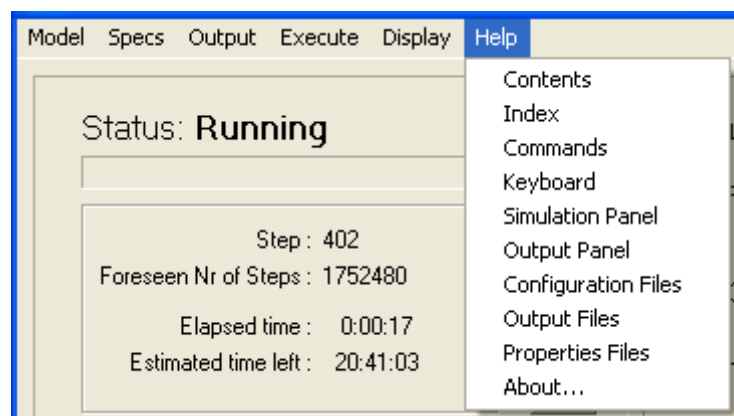
Intra-Classes Messages Area

The intra-classes messages area is shown in the Main Window, that is expanded when the option Intra-Classes Messages is selected in the Display menu. When toggled on, the main window creates one text area to display internal messages of the simulation objects, since they invoke the DebugMessage in the base class.



Help Menu

The Help Menu presents the contents of the users manual and the about screen.



- Contents - presents this help manual.
- Index - presents the index of the users manual

- Commands - presents the main window commands
- Keyboard - presents the keyboard shortcuts
- Simulation Panel - describes the layout of the simulation panel
- Output Panel - describes the layout of the output panel
- Configuration Files - describes the format of the model configuration files
- Output Files - describes the format of the output files
- Properties Files - describes the format of the properties files used by EcoDynamo
- About - presents the about screen information.

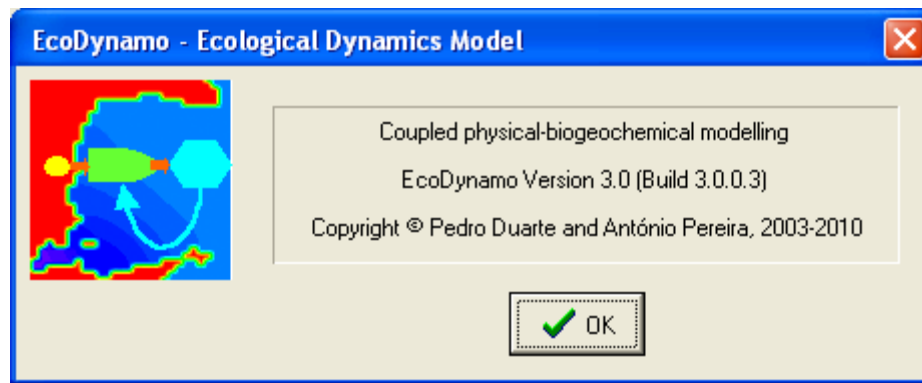
Keyboard Shortcuts

Some commands are mapped with keyboard shortcuts. The current list is the following:

| Command | Shortcut |
|------------------|-----------|
| New Model... | Ctrl + N |
| Open Model... | Ctrl + O |
| Close Model | Ctrl + L |
| Remove Model... | Ctrl + M |
| Default Model... | Ctrl + D |
| | |
| Initialise | Ctrl + I |
| Step | Ctrl + T |
| Run | Ctrl + R |
| Pause | Ctrl + P |
| Stop | Ctrl + S |
| | |
| Exit EcoDynamo | Ctrl + F4 |

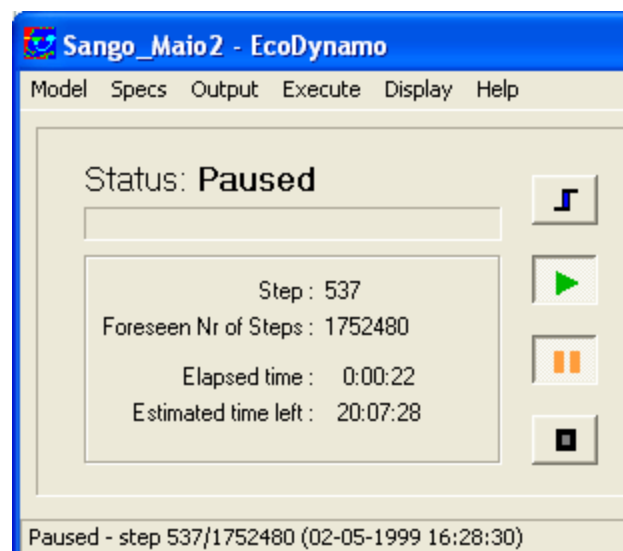
About

The About Menu presents the about screen with information about the application version, developers and copyright.



Simulation Panel

The Simulation Panel presents the simulation state and enables the control of the simulation runs. This panel synthesizes the Execute Menu.



The buttons have the general symbols used in hi-fi control panels:



Step button - run one step of the simulation.



Run button - execute the simulation until stopped or paused.



Toggled on when model running.



Pause button - exchange between pause and run mode.



Toggled on when model paused.



Stop button - stops the simulation run.

The labels has the following meaning:

Status - simulation status (Idle, Running, Paused, Stopped).

Step - simulation step number.

Foreseen Nr of Steps - expected number of steps for all simulation.

Elapsed time - simulation time from the beginning.


Estimated time left - presents the expected time for simulation end.


The Status Bar complements the information of the simulation panel, adding the date and time of the simulated step in the model.

Output Panel

The Output Panel presents the outputs selected for the simulation run and enables the control of the output type.

The checkboxes enable/disable the corresponding output device (File, Chart and Table).

Each  button opens the dialog to select the points for the corresponding output (see File, Chart and Table).

The  button opens the dialog to select the filename of the output file (see File). The filename is printed in the combo-box area.

The checkbox Only Mean Values enables a special kind of output register that saves only the mean values (see Mean Values File). The number of samples in each file is configured in the respective field.

The checkbox HDF Separated Files saves the outputs in several HDF formatted files (follow the HDF specifications - see NCSA http server (<http://hdf.ncsa.uiuc.edu>). The number of samples in each file is configured in the respective field. When this option is selected one sequence number is appended to the filename.

This panel is an extension of the Output Menu.

Configuration Files

The configuration files are the files used to initialise and run the model for simulation. Each file has its own significance and utility.

The mandatory files are:

Morphology File - describe the morphology of the ecosystem, including the number of cells, localisation, geometry and boundary type of each one.

Classes File - list all classes that can be used by the simulation.

Variables File - each class has its own list of variables treated and their initial values.

Parameters File - each class has its own list of parameters and their initial, minimum, maximum and increment values.

The following files are not mandatory but can be useful in some ecosystems or simulations:

Loads File - requested when the loads process is active.

River Loads File - requested when there are rivers loading to the model.

Losses File - requested when the model flows to the exterior.

Sea Boundaries File - file with the sea boundaries definitions.

Benthic Species File - configures where there are benthic species spread by the model.

Sediments File - configures the type of sediments existent in the model.

Points File - list sub domain points.

All these files are saved in text format (tab or comma separated values) and can be accessed and modified by any text editor or commercial application (Excel, Word, Wordpad, Notepad, ...).

Morphology File

The morphology file must terminate with the suffix "Morphology.xls".

It must have the format showed in the next figure:

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|----|---------------------------------|-------|--------|----------|-----------|----------|--------------|---------|-----------|-----------|-----------|-----------|---|
| 1 | NumberOfColumns | 32 | | | | | | | | | | | |
| 2 | NumberOfLines | 35 | | | | | | | | | | | |
| 3 | NumberOfLayers | 1 | | | | | | | | | | | |
| 4 | NumberOfBoxes | 1120 | | | | | | | | | | | |
| 5 | Critical depth for land boundar | 0.02 | | | | | | | | | | | |
| 6 | ModelType | 2DH | | | | | | | | | | | |
| 7 | Columns | Lines | Layers | BoxDepth | BoxLength | BoxWidth | BoxElevation | BoxType | Nboundary | Eboundary | Sboundary | Wboundary | |
| 8 | | 1 | 35 | 1 | -10 | 500 | 500 | 1.6878 | 1 | 0 | 0 | 4 | 4 |
| 9 | | 2 | 35 | 1 | -10 | 500 | 500 | 1.6891 | 1 | 0 | 0 | 4 | 0 |
| 10 | | 3 | 35 | 1 | -10 | 500 | 500 | 1.6904 | 1 | 0 | 0 | 4 | 0 |
| 11 | | 4 | 35 | 1 | -10 | 500 | 500 | 1.6917 | 1 | 0 | 0 | 4 | 0 |
| 12 | | 5 | 35 | 1 | -10 | 500 | 500 | 1.6931 | 1 | 0 | 0 | 4 | 0 |
| 13 | | 6 | 35 | 1 | -10 | 500 | 500 | 1.6944 | 1 | 0 | 0 | 4 | 0 |
| 14 | | 7 | 35 | 1 | -10 | 500 | 500 | 1.6958 | 1 | 0 | 0 | 4 | 0 |

The field names NumberOfColumns, NumberOfLines and NumberOfBoxes are mandatories and the columns that follow each one of them define the dimensions of the model grid:

- NumberOfColumns - number of cells in the longitude direction
- NumberOfRows - number of cells in the latitude direction
- NumberOfLayers - number of layers in 3-D models (can be omitted if it is 1)
- NumberOfBoxes - product of the three previous values
- Critical depth for land boundary - minimum value of water height considered for hydrodynamic calculations
- ModelType - type of model (0D, 1D, 2DH, 2DV, 3D)

After the model initialisation these values are displayed in the dialog of the Morphology menu.

There must be an head line with the following mandatory labels:

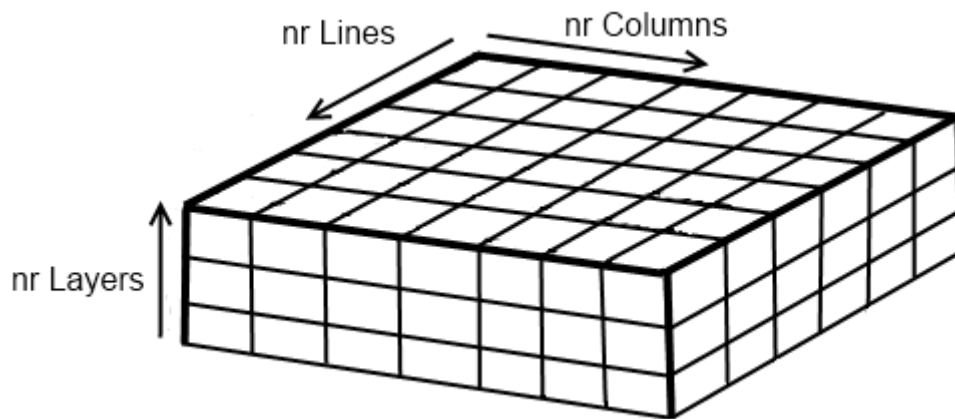
- Columns - number of cell column - increases from West to East (left-right)
- Lines - number of cell line - increases from North to South (top-down)
- Layers - number of cell layer - increases from deep to surface (bottom-up) - can be omitted if NumberOfLayers = 1
- BoxDepth - cell depth (meters) - negative values for land
- BoxLength - cell length (meters)
- BoxWidth - cell width (meters)
- BoxElevation - cell elevation of free surface (sea 0-level)
- BoxType - cell type (1 - border cell, 0 - inner cell)
- Nboundary - cell North boundary type

- Eboundary - cell East boundary type
- Sboundary - cell South boundary type
- Wboundary - cell West boundary type

Boundary types are coded like:

- 0 - no frontier;
- 1 - river boundary;
- 2 - sea boundary;
- 4 - solid boundary (earth).

Next picture shows the notation followed for the model grid numbering.



Classes File

The classes file must terminate with the suffix "Classes.xls".

It must have the format showed in the next figure:

| | A | B |
|----|---|----------|
| 1 | Number of Classes | 16 |
| 2 | Class name | Status |
| 3 | TTideWithWanContinuousHarmonics | Active |
| 4 | TWaterTemperatureTwoDimensionalForSango | Active |
| 5 | TAdriaticAirTemperature | Active |
| 6 | TWind | Active |
| 7 | TLight | Active |
| 8 | TFlow | Inactive |
| 9 | TBiDimensionalSango3 | Active |
| 10 | TSangoNutrients | Active |
| 11 | TSangoResuspendDeposit | Active |
| 12 | TLaminariaJaponicaExponentialGrowth | Active |
| 13 | TSangoPhytoplankton | Active |
| 14 | TSangoZooplankton | Active |
| 15 | TChlamysFarreriV8 | Active |
| 16 | TCrassostreaGigas7 | Active |
| 17 | TSangoMan | Active |
| 18 | TFisherMan | Active |
| 19 | | |

The labels Number of Classes, Class name and Status are mandatory.

The column that follows the label Number of Classes must define the number of classes that will be used by the model.

The line with labels Class name and Status head those with the classes names and their initial state of activation. There must be an agree between the number of classes defined and the number of lines with class names.

The names of the classes are the names of the class objects defined by each DLL, included with the EcoDynamo code, that implements the simulation of the different processes.

In this version of EcoDynamo the field Status is not taken into account because the class activation is done in the Classes menu and activation changes are saved when the model is closed in the EcoDynamo Properties File.

Variables File

Each class type has its own variables file. The file lists all the class variables, possibly with their initial values, and each class imposes its own format for the variables filling.

The figure shows one example for the class Light:

| | A | B | C | D | E |
|----|-------|----|--------------------------------------|----|---|
| 1 | Light | 15 | Total surface irradiance | | |
| 2 | | | PAR surface irradiance | | |
| 3 | | | Daylight hours | | |
| 4 | | | Mean horizontal water irradiance | | |
| 5 | | | Mean vertical water irradiance | | |
| 6 | | | Mean horizontal water PAR irradiance | | |
| 7 | | | Mean vertical water PAR irradiance | | |
| 8 | | | Noon surface PAR | | |
| 9 | | | Photic depth | | |
| 10 | | | Sub-surface irradiance | | |
| 11 | | | Sub-surface PAR irradiance | | |
| 12 | | | Atmospheric IR | | |
| 13 | | | Latitude | 37 | |
| 14 | | | Julian day | | |
| 15 | | | Current time | | |
| 16 | | | | | |

Detailed information about each class is documented by the DLL class developer team.

Parameters File

Each class has its own parameters file. The file lists all the class parameters and their initial values. Each class imposes its own format for the parameters filling, and the minimum, maximum and increment values can also be found as auxiliary values for the model calibration, as well as the parameter units.

The figure shows one example for the class Phytoplankton with the initial values and units:

| | A | B | C | D | E |
|----|---------------|----|---|-------------|-----|
| 1 | Phytoplankton | 13 | Pmax | 0.0504 | h-1 |
| 2 | | | Iopt | 491.427302 | |
| 3 | | | Maintenance respiration | 0.02 | d-1 |
| 4 | | | Respiration Coefficient | 0.3 | % |
| 5 | | | Dissolved organic carbon loss | 0.05 | |
| 6 | | | Dissolved organic carbon nutrient stress loss | 0 | |
| 7 | | | Death loss | 0.05 | d-1 |
| 8 | | | KAmmonia | 0.5 | |
| 9 | | | KNitrate | 0.5 | |
| 10 | | | Ks | 1.19 | |
| 11 | | | TemperatureAugmentationRate | 0.016954654 | |
| 12 | | | RedfieldCFactor | 6 | |
| 13 | | | RedfieldNFactor | 1 | |
| 14 | | | | | |

Detailed information about each class is documented by the DLL class developer team.

Loads File

The loads file must terminate with the suffix "Loads.xls".

It must follow the format showed in the next figure:

| | A | B | C | D | E | F | G | H |
|----|---------------|----------------------|-----------|-------------|---|---------|-------|-------|
| 1 | NumberOfLoads | NumberOfDaysForLoads | LoadLines | LoadColumns | LoadName | | | |
| 2 | 6 | 365 | 90 | 77 | ETAR Municipal da zona Noroeste de Faro | | | |
| 3 | | | 107 | 110 | ETAR Municipal da zona Nascente de Faro | | | |
| 4 | | | 95 | 156 | ETAR Municipal Poente | | | |
| 5 | | | 80 | 183 | ETAR Municipal Nascente | | | |
| 6 | | | 60 | 251 | ETAR Municipal da Fuzeta | | | |
| 7 | | | 89 | 21 | ETAR Municipal Quinta do Lago | | | |
| 8 | | | | | | | | |
| 9 | Flow1 | Flow2 | Flow3 | Flow4 | Flow5 | Flow6 | Flow7 | Flow8 |
| 10 | 0.030509259 | 0.347222222 | 0.462963 | 0.00520833 | 0 | 0.02653 | 0 | 0 |
| 11 | 0.030509259 | 0.347222222 | 0.462963 | 0.00520833 | 0 | 0.02653 | 0 | 0 |
| 12 | 0.030509259 | 0.347222222 | 0.462963 | 0.00520833 | 0 | 0.02653 | 0 | 0 |
| 13 | 0.030509259 | 0.347222222 | 0.462963 | 0.00520833 | 0 | 0.02653 | 0 | 0 |
| 14 | 0.030509259 | 0.347222222 | 0.462963 | 0.00520833 | 0 | 0.02653 | 0 | 0 |

The field names NumberOfLoads, NumberOfDaysForLoads, LoadLines, LoadColumns and LoadName are mandatories and act as headers:

- NumberOfLoads - The line below contains the number of load discharges points.
- NumberOfDaysForLoads - The line below contains the number of days with load discharges.
- LoadLines - The lines below contain the line numbers of each load discharge point. The number of lines filled must agree with the number of loads defined.
- LoadColumns - The lines below contain the column numbers of each load discharge point. The pair Line x Column defines the cell point. The number of lines filled must agree with the number of loads defined.
- LoadName - The lines below contain the names of the load discharges entities. The number of lines filled must agree with the number of loads defined.

For each load discharge point one column labeled FlowN must be added to the file (N is the flow number) with NumberOfDaysForLoads lines, one load value for each day.

The label NumberOfHoursForLoads can replace the label NumberOfDaysForLoads with the correspondent significance.

River Loads File

The rivers loads file must terminate with the suffix "Rivers.xls".

The rivers loads file follows the idea of the loads file, and the format can be seen in the next figure:

| | A | B | C | D | E | F |
|----|----------------|-------------------------------|----------------|------------------|----------|--------------|
| 1 | NumberOfRivers | NumberOfDaysForFlowTimeSeries | InputFlowLines | InputFlowColumns | Boundary | RiverName |
| 2 | 5 | 365 | 88 | 24 | 3 | River S Lour |
| 3 | | | 90 | 110 | 3 | River Seco |
| 4 | | | 88 | 25 | 3 | River Corgo |
| 5 | | | 85 | 77 | 3 | River Biogal |
| 6 | | | 89 | 149 | 3 | Rio Bmandil |
| 7 | | | | | | |
| 8 | Flow1 | Flow2 | Flow3 | Flow4 | Flow5 | |
| 9 | 0 | 0 | 0 | 0 | 0 | |
| 10 | 0 | 0 | 0 | 0 | 0 | |
| 11 | 0 | 0 | 0 | 0 | 0 | |
| 12 | - | - | - | - | - | - |

The field names NumberOfRivers, NumberOfDaysForFlowTimeSeries, InputFlowLines, InputFlowColumns, Boundary and RiverName are mandatories and act as headers:

- NumberOfRivers - The line below contains the number of rivers.
- NumberOfDaysForFlowTimeSeries - The line below contains the number of days with river flows.
- InputFlowLines - The lines below contain the line numbers of each river point. The number of lines filled must agree with the number of rivers defined.
- InputFlowColumns - The lines below contain the column numbers of each river point. The pair Line x Column defines the cell point. The number of lines filled must agree with the number of rivers defined.
- Boundary - Direction of the river flow inside the cell:
 - 1 - Flow to North
 - 2 - Flow to East
 - 3 - Flow to South
 - 4 - Flow to West
- RiverName - The lines below contain the names of the rivers. The number of lines filled must agree with the number of rivers defined.

For each river one column labeled FlowN must be added to the file (N is the flow number) with NumberOfDaysForFlowTimeSeries lines, one flow value for each day.

The label `NumberOfHoursForFlowTimeSeries` can replace the label `NumberOfDaysForFlowTimeSeries` when the flows are registered by hour.

Losses File

The losses file must terminate with the suffix "Losses.xls".

The losses loads file follows the idea of the loads file, as can be seen in the next figure:

| | A | B | C | D | E |
|---|-----------------------------|------------------------------------|--------------------------|----------------------------|-------------------------|
| 1 | <code>NumberOfLosses</code> | <code>NumberOfDaysForLosses</code> | <code>LossesLines</code> | <code>LossesColumns</code> | <code>LossesName</code> |
| 2 | 1 | 651 | 1 | 1 | TankLoss |
| 3 | | | | | |
| 4 | <code>Disch1</code> | | | | |
| 5 | 0.005 | | | | |
| 6 | 0.005 | | | | |
| 7 | 0.005 | | | | |
| 8 | 0.005 | | | | |

The field names `NumberOfLosses`, `NumberOfDaysForLosses`, `LossesLines`, `LossesColumns` and `LossesName` are mandatory and act as headers:

- `NumberOfLosses` - The line below contains the number of losses discharges points.
- `NumberOfDaysForLosses` - The line below contains the number of days with losses discharges.
- `LossesLines` - The lines below contain the line numbers of each loss discharge point. The number of lines filled must agree with the number of losses defined.
- `LossesColumns` - The lines below contain the column numbers of each loss discharge point. The pair Line x Column defines the cell point. The number of lines filled must agree with the number of losses defined.
- `LossesName` - The lines below contain the names of the losses discharges entities. The number of lines filled must agree with the number of losses defined.

For each loss discharge point one column labeled `DischN` must be added to the file (N is the discharge number) with `NumberOfDaysForLosses` lines, one loss discharge value for each day.

The label `NumberOfHoursForLosses` can replace the label `NumberOfDaysForLosses` with the corresponding significance.

Benthic Species File

The benthic species file must terminate with the suffix "BenthicSpecies.xls".

It defines the cells and the species names where benthic species are spread within the model area. One example of this file is shown in the figure:

| | A | B | C | D | E |
|---|---------------------------------|----------------|----------------------|---|---|
| 1 | NumberOfCellsWithBenthicSpecies | | | | |
| 2 | 2824 | | | | |
| 3 | ColumnCoordinate | LineCoordinate | SpeciesName | | |
| 4 | 146 | 119 | Ruditapes decussatus | | |
| 5 | 147 | 119 | Ruditapes decussatus | | |
| 6 | 148 | 119 | Ruditapes decussatus | | |

The field names NumberOfCellsWithBenthicSpecies, ColumnCoordinate, LineCoordinate and SpeciesName are mandatory and act as headers:

- NumberOfCellsWithBenthicSpecies - The line below contains the number of cells with benthic species.
- ColumnCoordinate - The lines below contain the numbers of the cell columns with benthic species.
- LineCoordinate - The lines below contain the numbers of the cell lines with benthic species. The number of lines and columns filled must agree with the number of cells defined with benthic species.
- SpeciesName - The lines below contain the names of the benthic species present in the referred cell. The number of lines filled must agree with the number of cells defined with benthic species. Some examples of implemented benthic species:
 - Enteromorpha sp
 - Laminaria japonica
 - SaltMarsh
 - Ulva sp
 - Zostera marina
 - Zostera noltii
 - Crassostrea gigas
 - Chlamys farreri
 - Mytilus galloprovincialis
 - Ruditapes decussatus

Sediments File

The sediments file must terminate with the suffix "Sediments.xls" and follows the format that can be seen in the next figure:

| | A | B | C | D | E | F | G | H |
|---|----------------------------|---------------|--------------|---------|----------|-------------|--------------------|---|
| 1 | NumberOfCellsWithSediments | | | | | | | |
| 2 | 13510 | | | | | | | |
| 3 | SedimentColumns | SedimentLines | SedimentName | Density | Porosity | LayerHeight | SedimentWaterRatio | |
| 4 | 99 | 137 | Vasa | 2.3 | 0.6354 | 0.003 | 1341 | |
| 5 | 99 | 136 | Vasa | 2.3 | 0.6354 | 0.003 | 1341 | |
| 6 | 99 | 135 | Vasa | 2.3 | 0.6354 | 0.003 | 1341 | |
| 7 | 99 | 134 | Vasa | 2.3 | 0.6354 | 0.003 | 1341 | |
| 8 | 100 | 134 | Vasa | 2.3 | 0.6354 | 0.003 | 1341 | |
| 9 | 99 | 133 | Vasa | 2.3 | 0.6354 | 0.003 | 1341 | |

The field names NumberOfCellsWithSediments, SedimentColumns, SedimentLines, SedimentName, Density, Porosity, LayerHeight and SedimentWaterRatio are mandatory and act as headers:

- NumberOfCellsWithSediments - The line below contains the number of cells with sediments.
- SedimentColumns - The lines below contain the numbers of the cell columns with sediments.
- SedimentLines - The lines below contain the numbers of the cell lines with sediments. The number of lines and columns filled must agree with the number of cells defined with sediments.
- SedimentName - The lines below contain the names of the sediment type present in the referred cell. Some examples of implemented sediment types:
 - Vasa
 - Vasa Arenosa
 - Areia
 - Areia Vasosa
- Density, Porosity, LayerHeight and SedimentWaterRatio contain the values of the correspondent fields.

Sea Boundaries File

The sea boundaries file must terminate with the suffix "SeaBoundaries.xls" and follows the format that can be seen in the next figure:

| | A | B | C | D | |
|----|-----------------------|------------------------------|-----------------|------------|------------|
| 1 | NumberOfSeaBoundaries | NumberOfDaysForSeaBoundaries | | | |
| 2 | 53 | 365 | | | |
| 3 | | | | | |
| 4 | InputFlowLines | InputFlowColumns | InputFlowLayers | Boundary | |
| 5 | 1 | 27 | 1 | 1 | |
| 6 | 1 | 28 | 1 | 2 | |
| 56 | 35 | 31 | 1 | 52 | |
| 57 | 35 | 32 | 1 | 53 | |
| 58 | | | | | |
| 59 | Velocity_1 | Velocity_2 | Velocity_3 | Velocity_4 | Velocity_5 |

The field names NumberOfSeaBoundaries, NumberOfDaysForSeaBoundaries, InputFlowLines, InputFlowColumns, Boundary are mandatory and act as headers:

- NumberOfSeaBoundaries - The line below contains the number of cells with sea boundaries.
- NumberOfDaysForSeaBoundaries - The line below contains the number of days with forced sea boundaries flows.
- InputFlowLines - The lines below contain the numbers of the cell lines with sea boundaries.
- InputFlowColumns - The lines below contain the numbers of the cell columns with sea boundaries.
- InputFlowLayers - The lines below contain the numbers of the cell layers with sea boundaries. This field is only mandatory if the model has the number of layers greater than one. The number of lines and columns and layers filled must agree with the number of cells defined with sea boundaries.
- Boundary - The lines below contain the number of the boundary flow.

For each sea boundary point, one column labeled Velocity_N must be added to the file (N is the sea boundary flow number) with NumberOfDaysForSeaBoundaries lines, one load value for each day.

The label NumberOfHoursForSeaBoundaries can replace the label NumberOfDaysForSeaBoundaries with the correspondent significance.

The use of this file is rare and it is only necessary when there are no class to simulate the sea tide.

Points File

The points file should have a format like the one showed in the next figure:

| | A | B | C | D |
|----|--------|------|-----------|----------|
| 1 | X | Y | COLUNA_X_ | LINHA_Y_ |
| 2 | 222000 | -400 | 131 | 151 |
| 3 | 222100 | -400 | 132 | 151 |
| 4 | 222200 | -400 | 133 | 151 |
| 5 | 221700 | -300 | 128 | 150 |
| 6 | 221800 | -300 | 129 | 150 |
| 7 | 221900 | -300 | 130 | 150 |
| 8 | 222000 | -300 | 131 | 150 |
| 9 | 222100 | -300 | 132 | 150 |
| 10 | 222200 | -300 | 133 | 150 |
| 11 | 222300 | -300 | 134 | 150 |
| 12 | 222400 | -300 | 135 | 150 |
| 13 | 222500 | -300 | 136 | 150 |
| 14 | 222600 | -300 | 137 | 150 |

The field names COLUNA_X_ and LINHA_Y_ are mandatories:

- COLUNA_X_ - The lines behind contain the column numbers of each point. The pair Line x Column defines the cell point.
- LINHA_Y_ - The lines behind contain the line numbers of each point. The pair Line x Column defines the cell point.

In the example shown above the values of columns COLUNA_X_ and LINHA_Y_ were built from X and Y (p.e. extracted with the help of one GIS application).

Output Files

The output files are the files generated by the EcoDynamo application during the simulation.

The results can be saved in General Output File or Mean Values Files.

The Log File saves the communications between classes during the simulation in the specified steps.

General Output File

The General Output File have, by default, 3 formats: "xls", "hdf" or "txt".

The "xls" and "txt" formats are text files saved with tabs separating values. The "xls" format is used for Excel application quick view with Tab Separated Values. The format is:

| | A | B | C | D | E | F | G | H | I | J |
|----|-----------|-------------|----------|------------|----------|-----------|-----------|-----------|----------------|-----|
| 1 | Time(UTC) | Time(hours) | TimeStep | GridColumn | GridLine | GridLayer | BoxNumber | Box depth | Dynamic height | U \ |
| 2 | 925646400 | 12 | 1 | 1 | 35 | 1 | 0 | -10 | 1.6878 | |
| 23 | 925646400 | 12 | 1 | 22 | 35 | 1 | 21 | 8.42069 | 2.49826621 | |
| 24 | 925646400 | 12 | 1 | 23 | 35 | 1 | 22 | 10.7534 | 2.49826621 | |
| 25 | 925646400 | 12 | 1 | 24 | 35 | 1 | 23 | 10.6201 | 2.49826621 | |
| 26 | 925646400 | 12 | 1 | 25 | 35 | 1 | 24 | 9.66476 | 2.49826621 | |
| 27 | 925646400 | 12 | 1 | 26 | 35 | 1 | 25 | 10.153 | 2.49826621 | |
| 28 | 925646400 | 12 | 1 | 27 | 35 | 1 | 26 | 13.7969 | 2.49826621 | |

The header row contains the fields significance. The first seven columns are fixed for all registers: time (UTC), time (in hours), register step, column, line, layer and cell numbers.

After the seventh column the variables appear in the order selected in Variables Selection Dialog to file output.

Only the cells selected in the File Points Dialog are registered.

The "hdf" format follows the HDF specifications - see HDF Group [http server](http://www.hdfgroup.org/) (<http://www.hdfgroup.org/>). When this option is chosen all the points of the model domain are saved.

Mean Values Files

The option to generate only Mean Values Files enables the user to run the model with a small time step (normally only the hydrodynamic part of the model) and save the mean values of flows and velocities in files. To do that the classes file must include classes with this feature implemented.

The user will use this files to run the model later with a time step greater than the used in the previous simulation.

The Mean Values Files are named as "xls" files, but the values are saved as Tab Separated Values, and their names follow a special order. The number of steps saved in each file is configured in the Output Panel.

The first file to be saved has the name "HydroTimeSeriesValues_0.xls", the second "HydroTimeSeriesValues_1.xls", and so on.

| | A | B | C | D | E | F | G | H |
|---|------------------|---------------|----------|-----------|-------------|-----------------|-------------|-----------------|
| 1 | Date_Time | Time(seconds) | TimeStep | BoxNumber | Mean U Flow | Mean U Velocity | Mean V Flow | Mean V Velocity |
| 2 | 06-01-2002 09:48 | 1010310480 | 1521 | 9430 | 0 | 0 | 0 | 0 |
| 3 | 06-01-2002 09:48 | 1010310480 | 1521 | 9431 | 0 | 0 | 11.6030637 | 0.02662825 |
| 4 | 06-01-2002 09:48 | 1010310480 | 1521 | 9432 | -11.1745218 | -0.03030495 | -2.75604734 | -0.00668889 |
| 5 | 06-01-2002 09:48 | 1010310480 | 1521 | 9727 | 0 | 0 | 0 | 0 |
| 6 | 06-01-2002 09:48 | 1010310480 | 1521 | 9728 | 0 | 0 | 0 | 0 |
| 7 | 06-01-2002 09:48 | 1010310480 | 1521 | 9729 | 0 | 0 | 0 | 0 |

The header row contains the fields significance. The first four columns are fixed for all registers: date_time, time (in seconds from January 1, 1970), register step and cell number.

After the fourth column the mean variables appear, also, in a fixed order: Mean U Flow, Mean U Velocity, Mean V Flow and Mean V Velocity.

Only the cells selected in the File Points Dialog are registered.

Log Files

The log files save the communications between classes during the simulation in some specified steps (specified in Logs Menu). Each step is saved in one separated file named "LogfileN", where N is the step number. The extension name reflects the format used.

The "xls" format is obtained with Tab Separated Values and is similar to the "txt" format. The header row describes each column field:

- STEP - time step
- Class_Name - origin class of the communication
- Type - method invoked: Update or Inquiry
- Data_Class - destination class of the communication
- Variable - variable of destination class
- Value - value of the variable
- Box_Number - number of cell

The Type "Update" means that class Class_Name will update the variable Variable in the class Data_Class with the value Value in the cell Box_Number.

The Type "Inquiry" means that class Class_Name asks for the value of the variable Variable in the class Data_Class in the cell Box_Number. The value is returned in Value.

| | A | B | C | D | E | F | G |
|------|------|----------------------|---------|--------------------------|------------------|----------|------------|
| 1 | STEP | Class_Name | Type | Data_Class | Variable | Value | Box_Number |
| 2 | 3 | TBiDimensionalSango3 | Inquiry | TTideWithWanContinuousHa | Tidal height | 2507.522 | 0 |
| 3 | 3 | TBiDimensionalSango3 | Inquiry | TTideWithWanContinuousHa | Tidal height | 2509.432 | 1 |
| 4 | 3 | TBiDimensionalSango3 | Inquiry | TTideWithWanContinuousHa | Tidal height | 2511.325 | 2 |
| 1968 | 3 | TSangoResuspendDep | Inquiry | TBiDimensionalSango3 | Drag coefficient | 0.005187 | 52 |
| 1969 | 3 | TSangoResuspendDep | Update | TSangoNutrients | Ammonia | 0.009328 | 52 |
| 1970 | 3 | TSangoResuspendDep | Inquiry | TSangoPhytoplankton | Phytoplankton b | 1.00499 | 52 |
| 1971 | 3 | TSangoResuspendDep | Inquiry | TSangoPhytoplankton | Chlorophyll to C | 50 | 52 |

The "xml" format is used for more sophisticated pos-processing methods, but the elements of the document follow the headers of the XLS file:

```

1      <?xml version="1.0" encoding="UTF-8"?>
2
3      <LOG>
4      <STEP Number="3">
5      <REGISTRY>
6          <Class_Name>TBiDimensionalSango3</Class_Name>
7          <Type>Inquiry</Type>
8          <Data_Class>TTideWithWanContinuousHarmonics</Data_Class>
9          <Variable>Tidal height</Variable>
10         <Value>2507.521972</Value>
11         <Box_Number>0</Box_Number>
12     </REGISTRY>
13     <REGISTRY>
14         <Class_Name>TBiDimensionalSango3</Class_Name>
15         <Type>Inquiry</Type>
16         <Data_Class>TTideWithWanContinuousHarmonics</Data_Class>
17         <Variable>Tidal height</Variable>
18         <Value>2509.431876</Value>
19         <Box_Number>1</Box_Number>
20     </REGISTRY>
21     <REGISTRY>
22         <Class_Name>TBiDimensionalSango3</Class_Name>
23         <Type>Inquiry</Type>
24         <Data_Class>TTideWithWanContinuousHarmonics</Data_Class>
25         <Variable>Tidal height</Variable>
26         <Value>2511.324807</Value>
27         <Box_Number>2</Box_Number>
28     </REGISTRY>
29 </STEP>
30 </LOG>

```

The root of the XML file is the LOG element that contains one STEP element, with the attribute Number identifying the simulation step number.

This element is composed by multiple REGISTRY elements, each one with six elements that maintain the names and the order of the headers described previously (one registry for each line).

Properties Files

There are two kinds of properties files:

Model Properties file saves the models already created by the EcoDynamo application in the application folder.

EcoDynamo Properties file saves, in the model folder, the last model simulation properties defined with the EcoDynamo application.

Model Properties File

This file is named "Models.properties" and belongs to the EcoDynamo application's folder.

The file saves the models used by EcoDynamo application and is created when the application ends.

Here is an example of model properties file:

```
DefinedModels=3
Model_0=Sango
Path_0=D:\Modelos\Sango
Default_0=false
AutoInit_0=false
Model_1=Alqueva
Path_1=D:\Modelos\Alqueva
Default_1=false
AutoInit_1=false
Model_2=RiaFOcidente2D100mSSLuisCycle
Path_2=D:\Modelos\RiaFOcidente2D100mSSLuisCycle
Default_2=true
AutoInit_2=false
```

The first line contains the number of models already created and opened by EcoDynamo (property name is DefinedModels).

For each created model, four properties are included:

- Model - model name
- Path - location of the model folder
- Default - true if the model is the default model; false otherwise
- AutoInit - true if the model is initialized when opened; false otherwise.

Each model property has the suffix _N where N is the order number of the model in the file.

EcoDynamo Properties Files

Each model created by EcoDynamo will have one file named "EcoDynamo.properties" in its database folder.

The file is created when the model is closed and saves the model properties for the simulation.

Here is an example of one EcoDynamo properties file:

```
PrefixName=RIAF2D
Type=2DH
PathName=D:\Modelos\RiaFOcidente2D100mSSLuisCycle
NrClassesAvailable=21
Available_0=TTide
Available_1=TBiDimensionalRiaFormosa
(...)
Available_20=TFisherMan
NrClassesSelected=11
Selected_0=TBiDimensionalTimeSeriesVariableDT
Selected_1=TLight
(...)
Selected_10=TWindDailyTimeSeries
NrVariablesAvailable=152
AvailableVars_0=U Flow
AvailableVars_1=V Flow
(...)
AvailableVars_151=Daily wind speed
NrVariablesSelectedFileOutput=10
SelectedVarsFileOutput_0=Salinity
SelectedVarsFileOutput_1=Water temperature
(...)
SelectedVarsFileOutput_9=Box depth
NrVariablesSelectedGraphOutput=0
NrVariablesSelectedTableOutput=0
```

```
StartTime=1010286480
FinishTime=1041292800
TimeStep=30.000000
TimeUnit=Seconds
RunMode=Cycle
Integration=Euler
OutputStartTime=1010286600
OutputFinishTime=1041292800
OutputFrequency=3600.000000
OutputTimeUnit=Hours
SimulationRunType=Cycle
NumberOfPeriods=1
StartPeriod_0=1010286480
FinishPeriod_0=1041292800
CyclePeriod_0=1252200
UnitCycle_0=1
FirstFileIndex_0=144
```

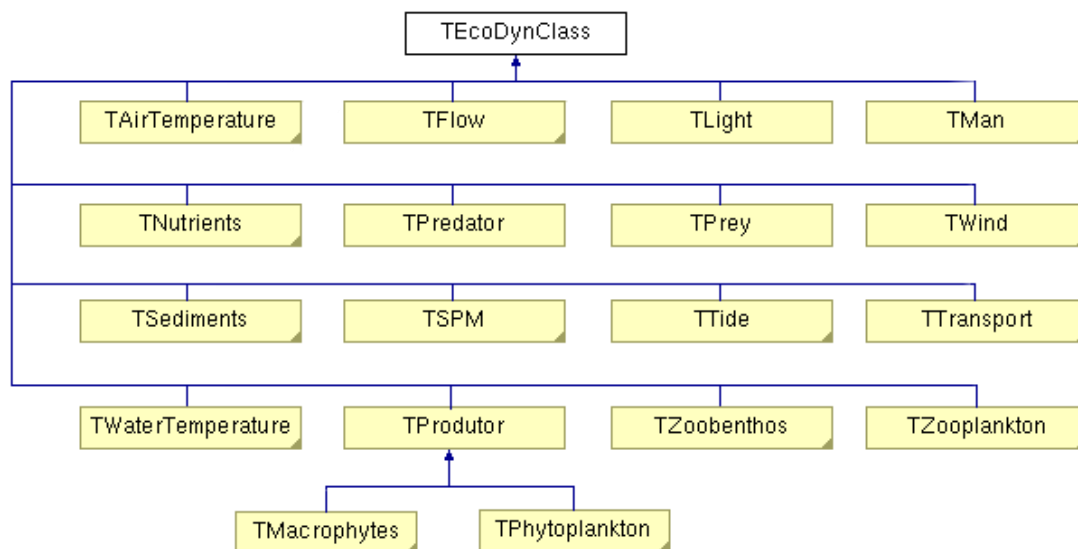
The properties saved in the file are:

- PrefixName - prefix that heads each mandatory file
- Type - type of model (0D, 1DH, 1DV, 2DH, 2DV, 3D)
- PathName - location of the model folder
- NrClassesAvailable - number of classes available in classes file
- Available_N - list of available classes [N is the class order number - Classes dialog]
- NrClassesSelected - number of classes selected
- Selected_N - list of selected classes [N is the class order number - Classes dialog]
- NrVariablesAvailable - number of variables available from classes selected
- AvailableVars_N - list of available variables [N is the variable order number - Variables dialog]
- NrVariablesSelectedFileOutput - number of variables selected for file output [Variables dialog]
- SelectedVarsFileOutput_N - list of selected variables to file output [N is the variable order number - Variables dialog]
- NrVariablesSelectedGraphOutput - number of variables selected for graph output
- SelectedVarsGraphOutput_N - list of selected variables to graph output [N is the variable order number - Variables dialog]
- NrVariablesSelectedTableOutput - number of variables selected for table output
- SelectedVarsTableOutput_N - list of selected variables to table output [N is the variable order number - Variables dialog]

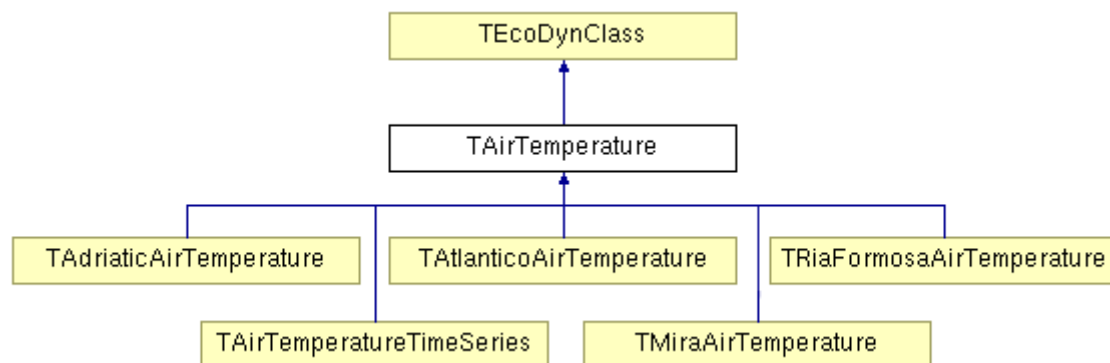
- StartTime - simulation start time (seconds from January 1, 1970) [Time specs dialog (Simulation tab)]
- FinishTime - simulation finish time (seconds from January 1, 1970) [Time specs dialog (Simulation tab)]
- TimeStep - simulation time step (in seconds) [Time specs dialog (Simulation tab)]
- TimeUnit - simulation time step unit [Time specs dialog (Simulation tab)]
- RunMode - simulation run mode [Time specs dialog (Simulation tab)]
- Integration - simulation time integration [Time specs dialog (Integration tab)]
- OutputStartTime - output register start time (seconds from January 1, 1970) [Time specs dialog (Output Register tab)]
- OutputFinishTime - output register finish time (seconds from January 1, 1970) [Time specs dialog (Output Register tab)]
- OutputFrequency - output frequency time (in seconds) [Time specs dialog (Output Register tab)]
- OutputTimeUnit - output frequency time unit [Time specs dialog (Output Register tab)]
- SimulationRunType - type of simulation (Cyclic, Continuous) [Time specs dialog (Cyclic simulation tab)]
- NumberOfPeriods - number of periods in cyclic simulation [Time specs dialog (Cyclic simulation tab)]
- StartPeriod_N - start period time [N is the period order number - Time specs dialog (Periodic simulations definitions tab)]
- FinishPeriod_N - finish period time [N is the period order number - Time specs dialog (Periodic simulations definitions tab)]
- CyclePeriod_N - cycle duration time [N is the period order number - Time specs dialog (Periodic simulations definitions tab)]
- UnitPeriod_N - cycle period time unit [N is the period order number - Time specs dialog (Periodic simulations definitions tab)]
- FirstFileIndex - index of the first mean time values file [Time specs dialog (Periodic simulations definitions tab)]

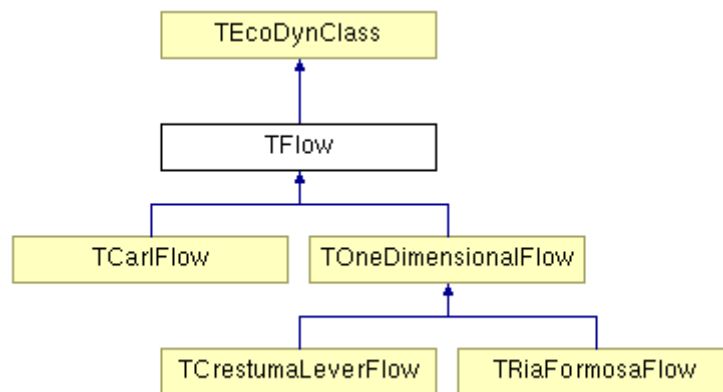
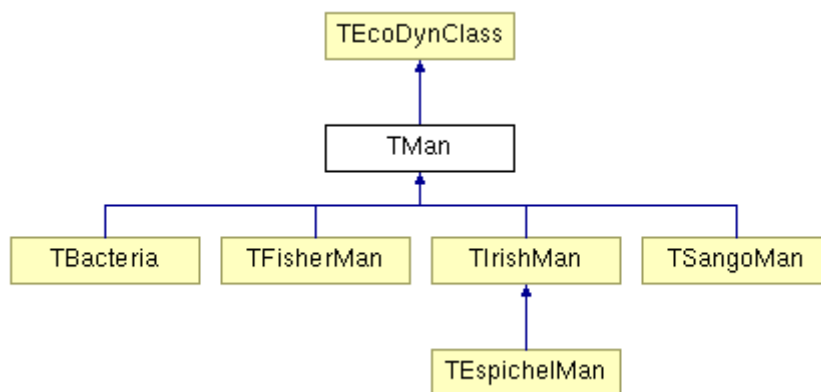
Annex 2 – EcoDynamo Classes Diagram

EcoDynClass Hierarchy Diagram

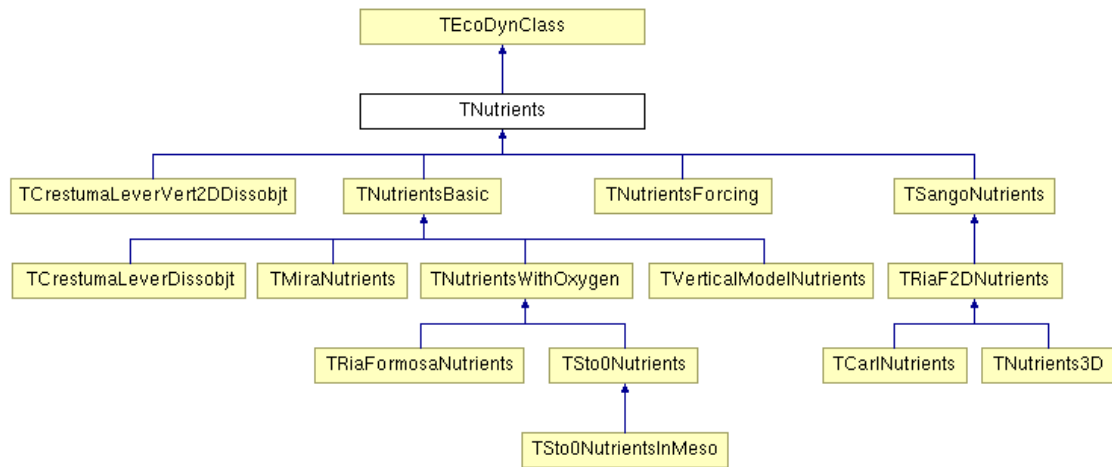


AirTemperature Hierarchy Diagram

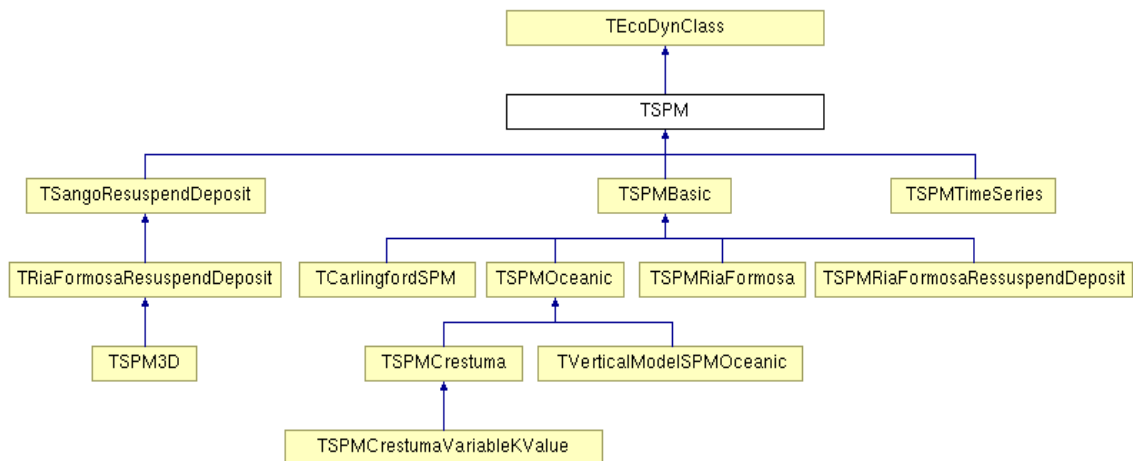


Flow Hierarchy Diagram***Man Hierarchy Diagram***

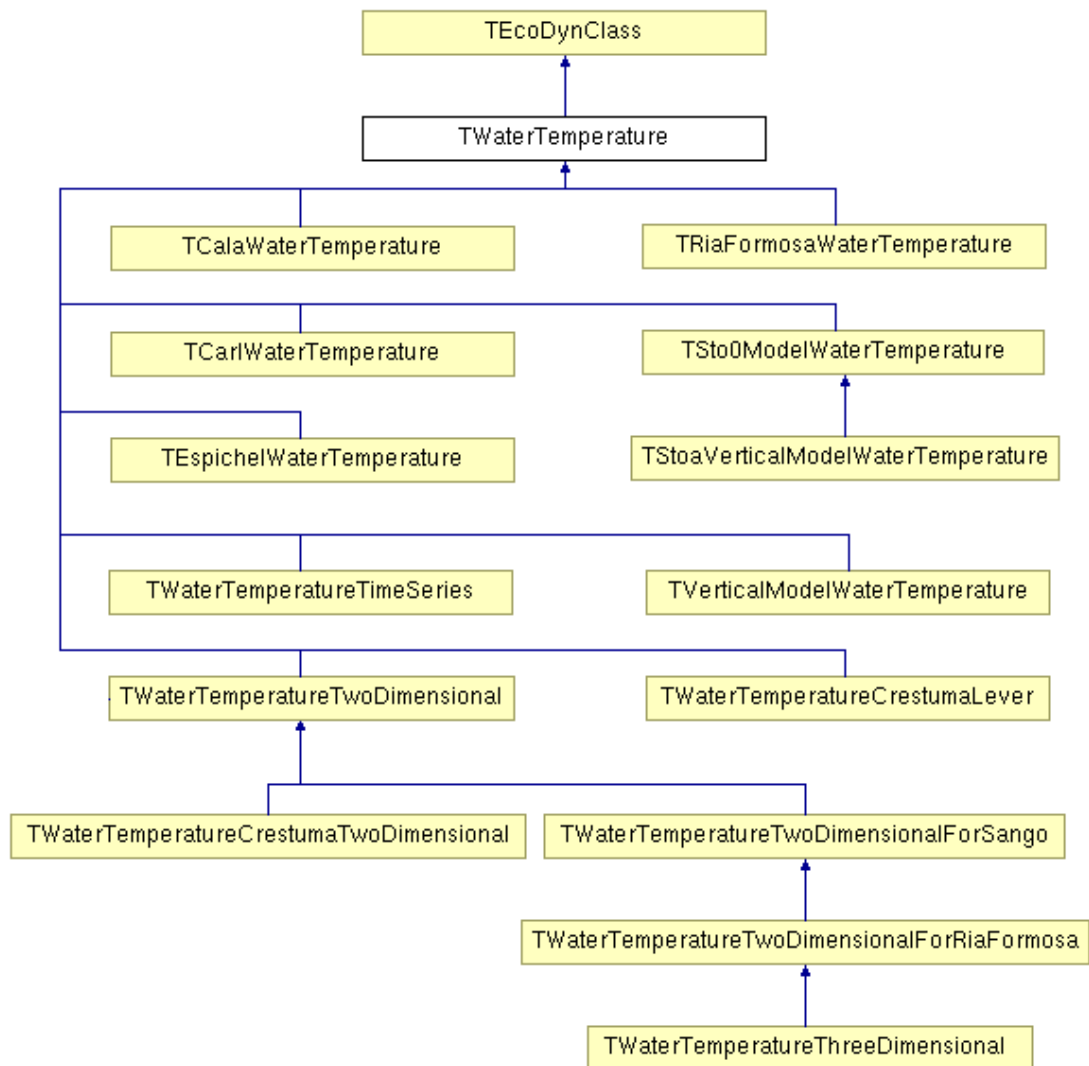
Nutrients Hierarchy Diagram



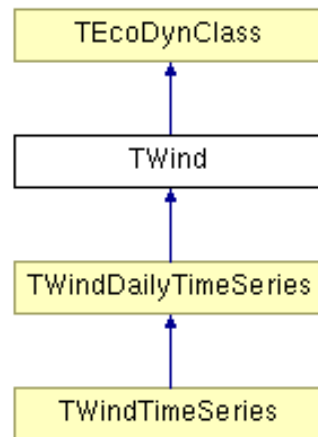
SuspendedMatter Hierarchy Diagram



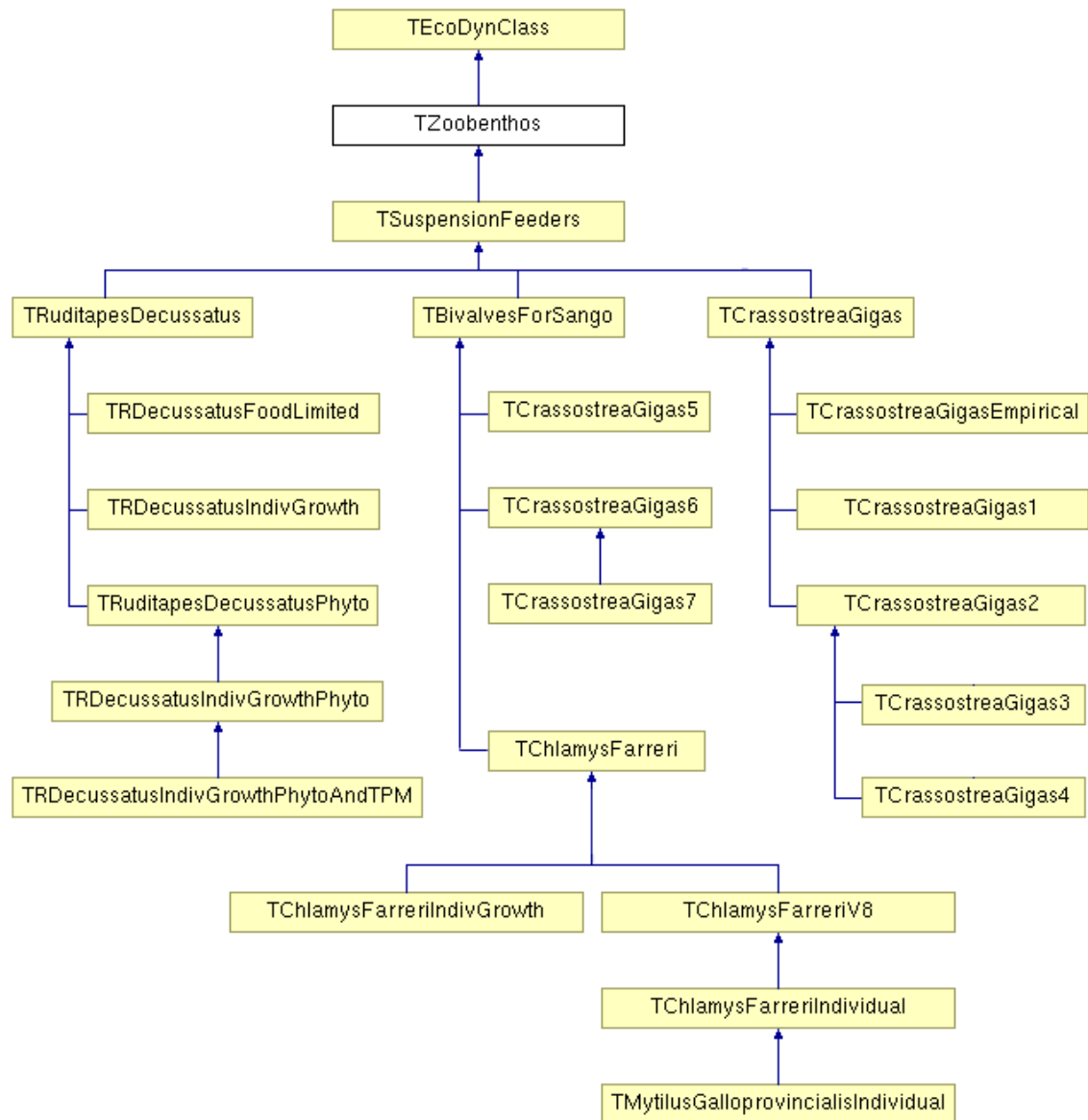
WaterTemperature Hierarchy Diagram



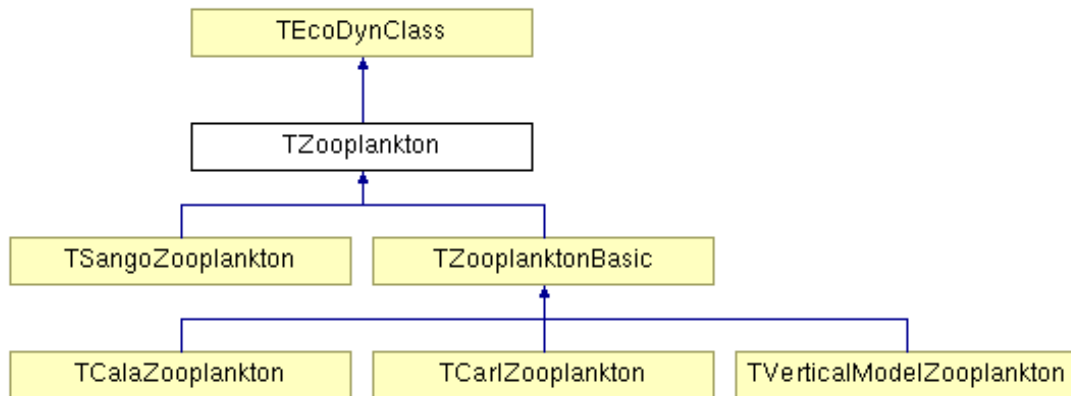
Wind Hierarchy Diagram



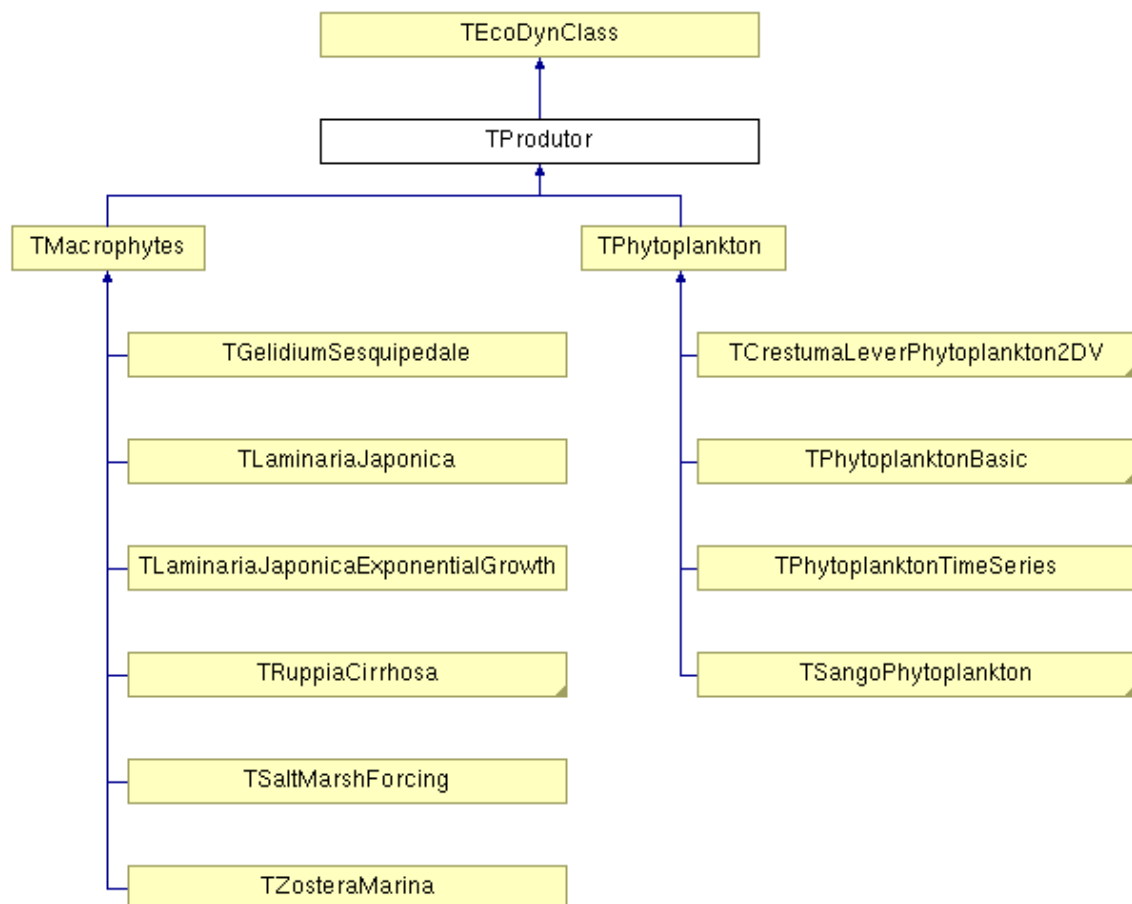
Zoobenthos Hierarchy Diagram



Zooplankton Hierarchy Diagram



Macrophytes and Phytoplankton Hierarchy Diagram



Annex 3 – ECOLANG Specification

One complete description of the ECOLANG communications language can be found in (Pereira, 2008) and an updated version of the document is available in [<http://paginas.fe.up.pt/~amcp/index.html>].

A subset of the document is presented in this section.

Introduction

ECOLANG is the communication language used in the EcoSimNet framework, a multi-agent systems environment for ecological simulations, rooted by the EcoDynamo simulator (Pereira and Duarte, 2005), linked with several intelligent agents and visualisation applications. This document extends the initial definition of the language (Pereira et al., 2005).

The agents' actions and perceptions are translated into messages exchanged between the simulator application and the agents.

The concepts' definitions used follow the BNF notation (Backus et al., 1960) and they're inspired in the COACH UNILANG language (Reis and Lau, 2002).

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

{ } – curly braces used for repetitive items (one or more times);

[] – square braces enclose types of values;

Terminal symbols use bold face letters.

1 MESSAGE DEFINITION

The base syntax of each message is as follows:

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

<ID> is the message identifier sent by the initiator – it is a sequential integer number controlled by each sender (initial value is 1).

<SENDER> is the name of the message initiator application (source).

<RECEIVER> is the name of the message destination application.

<MSG_CONTENT> is the content of the message.

Each message will be represented by a numeric reference to facilitate its identification.

2 MESSAGE TYPES

Message exchanged by the applications may belong to five types: connection, definitions, actions, perceptions, and coordination.

The connection messages establish the communication sessions between the agents / applications and specify the computer where agent belongs and the port number in which the agent accepts connections and listens the messages.

Though there could be actions and perceptions dedicated to some type of the agents belonging to the system, there are no restrictions to the messages that applications can use.

```
<MSG_CONTENT> ::= <CONNECTION_MSG> | <DEFINITION_MSG> | <ACTION_MSG> |  
                <PERCEPTION_MSG> | <COORDINATION_MSG>
```

2.1 CONNECTION MESSAGES

Connection messages delimits the sessions between applications. In this group there are also messages to ask the agents known by the other partner of the session. This allows the establishment of links between multiple applications, facilitating the expansion of the communications and knowledge network.

```
<CONNECTION_MSG> ::= <CONNECT> | <DISCONNECT> | <ACCEPT> | <ASK_AGENTS> |  
                    <KNOWN_AGENTS>
```

The session begins with the *connect* message and finishes with the *disconnect* message. The *accept* message answers any of those messages:

```
<CONNECT> ::= connect <HOST_NAME> <HOST_ADDR> <SERVER_PORT> [1.1]  
<DISCONNECT> ::= disconnect [1.2]  
<ACCEPT> ::= accept (<ACTION_ID> <ACTION_RESULT>) [1.3]  
  
<HOST_NAME> ::= [string]  
<HOST_ADDR> ::= [string]  
<SERVER_PORT> ::= [integer]  
<ACTION_ID> ::= <ID>  
<ACTION_RESULT> ::= ok | failed
```

To ask the other partner for the known agents:

```
<ASK_AGENTS> ::= agents [1.4]
```

```
<KNOWN_AGENTS> ::= known_agents (<ACTION_ID> {<AGENT>}) [1.5]
```

```
<AGENT> ::= (<AGENT_NAME> <HOST_NAME> <HOST_ADDR> <SERVER_PORT> <CONNECTED>)
```

```
<AGENT_NAME> ::= [string]
```

```
<CONNECTED> ::= connected | disconnected
```

<HOST_NAME> and <HOST_ADDR> identify the computer name and IP address where the application runs, <SERVER_PORT> identifies the port number where the application expects connections.

<ACTION_ID> of the answer messages must be the <ID> of the corresponding connection message.

2.2 DEFINITIONS

From version 1.3 of the protocol, these messages include the definition of regions and information about the type of model loaded in the simulator, its dimensions, its morphology and species of shellfish and molluscs prevailing in it.

```
<DEFINITION_MSG> ::= <REGIONS_MSG> | <MODEL_DEFINITIONS>
```

Each region is referred to by name, will be of one type and will cover a certain area. It may also be defined at the expense of other regions already defined.

```
<REGIONS_MSG> ::= <DEFINE_ACTION> | <DELETE_ACTION> | <GET_REGIONS> |  
    <GET_REGION> | <DEFINE_RESULT> | <DELETE_RESULT> |  
    <GET_REGIONS_RESULT> | <GET_REGION_RESULT>
```

```
<DEFINE_ACTION> ::= define (<REG_NAME> <REGION>) [2.1]
```

```
<DELETE_ACTION> ::= delete {<REG_NAME>} [2.2]
```

```
<REG_NAME> ::= [string]
```

```
<REGION> ::= <REGION_TYPE> <REGION_AREA> | {<REG_NAME>}
```

Each region is of the type land or water and, in the later, is characterized by its quality as well as the type and quality of its sediment.

```
<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 a set of one or more simple regions, each one defined by one point or a simple polygon (rectangle, square, circle or circle 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 [integer] [integer])
```

Each of the previous messages (**define** and **delete**) should be answered by the application that receive the message:

```
<DEFINE_RESULT> ::= define_result (<ACTION_ID> <ACTION_RESULT>) [2.3]  
<DELETE_RESULT> ::= delete_result (<ACTION_ID> <ACTION_RESULT>) [2.4]
```

At any time it is possible to know which are the defined regions and their characteristics:

```
<GET_REGIONS> ::= get_region_names [2.5]  
<GET_REGION> ::= get_region <REG_NAME> [2.6]
```

The previous messages should obtain as answers, respectively

```
<GET_REGIONS_RESULT> ::= region_names (<ACTION_ID> {<REG_NAME>}) [2.7]  
<GET_REGION_RESULT> ::= region (<ACTION_ID> <REG_NAME> <REGION>) [2.8]
```

There are also messages to get information about the model loaded in the simulator – they can obtain the size, type, morphology and species of molluscs and shellfish prevailing in it:

```
<MODEL_DEFINITIONS> ::= <GET_DIMENSIONS> | <GET_MORPHOLOGY> | <GET_SPECIES>  
    | <DIMS_RESULT> | <MORPHOLOGY_RESULTS> | <SPECIES_RESULTS>  
  
<GET_DIMENSIONS> ::= model_dimensions [2.9]  
<GET_MORPHOLOGY> ::= model_morphology [2.10]  
<GET_SPECIES> ::= model_species [2.11]
```

The previous messages should obtain as answers, respectively:

```
<DIMS_RESULT> ::= dimensions (<ACTION_ID> <LINES> <COLUMNS> <LAYERS>  
    <MOD_TYPE>) [2.12]  
  
<MORPHOLOGY_RESULTS> ::= <MORPHOLOGY_RESULT> | <MORPHOLOGY_END>  
<MORPHOLOGY_RESULT> ::= morphology (<ACTION_ID> > {( <CELL> [real] )}) [2.13]  
<MORPHOLOGY_END> ::= morphology_end [2.14]  
<SPECIES_RESULTS> ::= <SPECIES_RESULT> | <SPECIES_END>  
<SPECIES_RESULT> ::= benthic_species (<ACTION_ID> {(<SPECIES_NAME>  
    <BOXES>)}) [2.15]
```

```

<SPECIES_END> ::= benthic_species_end [2.16]

<LINES> ::= [integer]
<COLUMNS> ::= [integer]
<LAYERS> ::= [integer]
<MOD_TYPE> ::= 0D | 1DH | 1DV | 2DH | 2DV | 3D
<CELL> ::= [integer]
<SPECIES_NAME> ::= ([string])
<BOXES> → defined in message [3.13] (<GET_VAR_VALUE>)

```

2.3 ACTIONS

The actions' messages are closely linked to each type of agent involved in the system.

An agent / application that has an interest in the production of shellfish has actions of deposit, inspect and collect species of molluscs.

```

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

```

To deposit (seed), the agent indicates the region, the time, the characteristics of the species of molluscs to deposit and the total weight seeded. The two real values indicated in the message may have different meanings, depending on molluscs in question. By example, for the oysters and scallops, the first value indicates the individual weight of the shell and the second indicates the individual weight of meat; for clams, the first value indicates the individual dry weight, and the second indicates the individual weight.

To inspect, the agent indicates the region and the time of the inspection.

To collect (harvest), it must indicate, beyond the region, the characteristics of shellfish to collect and time of collection.

```

<SEED_ACTION> ::= seed (<REG_NAME> <TIME> <BIVALVE_S> <DENSITY>) [3.1]

```

```

<INSPECT_ACTION> ::= inspect (<REG_NAME> <TIME>) [3.2]

```

```

<HARVEST_ACTION> ::= harvest (<REG_NAME> <TIME> <BIVALVE>) [3.3]

```

```

<BIVALVE_S> ::= <BTYP> ([real] [real])

```

```

<BIVALVE> ::= <BTYP> <SHELL_LENGTH>

```

```

<BTYP> ::= scallop | kelp | oyster | mussel | clam

```

```

<SHELL_LENGTH> ::= (length [real])

```

```

<DENSITY> ::= (density [real])

```

Reference to the action moment can be as quickly as possible (*now*) or one value that indicates the number of seconds from 1970 January, 1 00:00:00.

```

<TIME> ::= now | [integer]

```

Any agent / application can act over the simulator choosing the model it wants to simulate, controlling the parameterization of the model - gathering / changing parameters of the simulated classes and collecting / recording the results of the simulation. Messages can be divided into four different types:

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

Actions to choose the model to simulate - open or close model, survey the model in simulation

```
<MODEL_ACTION> ::= <OPEN_MODEL> | <CLOSE_MODEL> | <GET_MODEL> | <SAVE_CONF>
```

```
<OPEN_MODEL> ::= open_model <MODEL_NAME> [3.4]
```

```
<CLOSE_MODEL> ::= close_model [3.5]
```

```
<GET_MODEL> ::= model_name [3.6]
```

```
<SAVE_CONF> ::= save_configuration [3.7]
```

```
<MODEL_NAME> ::= [string]
```

Actions over the simulation execution - monitor and influence the simulations - initialising, running, stopping, restarting and completing the simulations:

```
<EXEC_ACTION> ::= initialise | run | stop | pause | <STEP_CMD> [3.8]
```

```
<STEP_CMD> ::= step [integer] [3.9]
```

Actions over the model in simulation – to choose / survey classes, choose / inquire initial values for variables and parameters, choose / survey frequency and range of simulation, choose / survey simulation sub-domain:

```
<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 [3.10]
```

```
<SELECT_CLASSES> ::= select_classes {<CLASS_NAME>} [3.11]
```

```
<SP_VARS> ::= <GET_CLASS_VARS> | <GET_VAR_VALUE> | <SET_VAR_VALUE>
```

```
<GET_CLASS_VARS> ::= get_variables <CLASS_NAME> [3.12]
```

```
<GET_VAR_VALUE> ::= get_variable_value <CLASS_NAME> <VAR_NAME> <BOXES> [3.13]
```

```
<SET_VAR_VALUE> ::= set_variable_value <CLASS_NAME> {(<VAR_NAME> <BOXES>  
    [real])} [3.14]
```

```
<SP_PARMS> ::= <GET_PARMS> | <SET_PARMS>
```

```
<GET_PARMS> ::= get_parameters <CLASS_NAME> [3.15]
```

```
<SET_PARMS> ::= set_parameters <CLASS_NAME> {(<PARAM_NAME> [real])} [3.16]
```

```
<SP_TIME> ::= <GET_TIME> | <SET_TIME>
```

```
<GET_TIME> ::= get_time_spec [3.17]
```

```
<SET_TIME> ::= set_time_spec <STEP> <START_TIME> <FINISH_TIME> [3.18]
```

```
<SUB_DOMAIN> ::= subdomain <DOMAIN> [3.19]
```

```

<CLASS_NAME> ::= ([string])
<VAR_NAME> ::= ([string])
<BOXES> ::= (<SUB_DOMAIN> | ({<CELL>}))
<PARM_NAME> ::= ([string])
<STEP> ::= [integer]
<START_TIME> ::= [integer]
<FINISH_TIME> ::= [integer]
<DOMAIN> ::= all | ({<REG_NAME>})

```

Actions to record the results - choose variables to register, frequency, range and type of recording, choose sub-domain to register or activate the monitoring mode (trace):

```

<REG_ACTION> ::= <REG_FILE> | <REG_VARS> | <REG_LOG> | <REG_TIME> |
<REG_TRACE>
<REG_FILE> ::= output_file <FILE_NAME> [3.20]

```

```

<REG_VARS> ::= <GET_VARS> | <SELECT_VARS> | <UNSELECT_VARS>
<GET_VARS> ::= get_available_variables [3.21]

```

```

<SELECT_VARS> ::= select_variables <OUTPUT_TYPE> ({<VAR_NAME>}) <BOXES> [3.22]

```

```

<UNSELECT_VARS> ::= unselect_variables <OUTPUT_TYPE> {<VAR_NAME>} [3.23]

```

```

<REG_LOG> ::= log <LOG_TYPE> ({<LOG_STEP>}) [3.24]

```

```

<REG_TIME> ::= <GET_REG_TIME> | <SET_REG_TIME>
<GET_REG_TIME> ::= get_output_time [3.25]

```

```

<SET_REG_TIME> ::= set_output_time <STEP> <START_TIME> <FINISH_TIME> [3.26]

```

```

<REG_TRACE> ::= trace [3.27]

```

```

<FILE_NAME> ::= ([string])
<OUTPUT_TYPE> ::= file | graph | table | remote
<LOG_TYPE> ::= xml | xls | txt | remote
<LOG_STEP> ::= [integer]

```

2.4 PERCEPTIONS

The perceptions' messages are closely related with the type of agent involved in the system and also to the actions performed by each over the simulator.

An agent with an interest in the production of shellfish, seed molluscs and its perceptions are the result of the actions taken.

The response to the seed action of the agent may be positive or negative (in the case such action is denied). In response to the inspection action the agent receives a message with the bivalve's characteristics in the region. The resulting harvest is negative or positive, and in this case, it is indicated the total weight harvested.

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

```
<SEED_RESULT> ::= seed_result (<ACTION_ID> <ACTION_RESULT>) [4.1]
```

```
<INSPECT_RESULT> ::= inspect_result (<ACTION_ID> {<BIVALVE>}) [4.2]
```

```
<HARVEST_RESULT> ::= harvest_result (<ACTION_ID> <ACTION_RESULT> <WEIGHT>)  
[4.3]
```

```
<WEIGHT> ::= [real]
```

The <ACTION_ID> of the perception message identifies the <ID> of the corresponding action message.

The agents' / applications' perceptions are both messages with the result of the actions initiated by the agent / application or messages spontaneously sent by the simulator. They may be from 5 types:

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

Answers to the actions over the model:

```
<MODEL_RESULT> ::= <OPEN_RESULT> | <CLOSE_RESULT> | <GET_RESULT> |  
    <SAVE_RESULT>
```

```
<OPEN_RESULT> ::= open_result (<ACTION_ID> <ACTION_RESULT>) [4.4]
```

```
<CLOSE_RESULT> ::= close_result (<ACTION_ID> <ACTION_RESULT>) [4.5]
```

```
<GET_RESULT> ::= model (<ACTION_ID> <MODEL_NAME>) [4.6]
```

```
<SAVE_RESULT> ::= save_result (<ACTION_ID> <ACTION_RESULT>) [4.7]
```

Answers to the actions over the simulation execution:

```
<EXEC_RESULT> ::= exec_result (<ACTION_ID> <ACTION_RESULT>) [4.8]
```

Answers to the actions over the model configuration:

```
<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>}) [4.9]
```

```
<CLASSES_SELECTED> ::= classes_selected (<ACTION_ID> {<CLASS_NAME>}) [4.10]
```

```
<VARS_RESULT> ::= <CLASS_VARS> | <VAR_VALUE> | <VAR_SET>
```

```
<CLASS_VARS> ::= variables (<ACTION_ID> {<VAR_NAME>}) [4.11]
```

```
<VAR_VALUE> ::= variable_value (<ACTION_ID> {(<CELL> [real])}) [4.12]
```

```
<VAR_SET> ::= variable_set_result (<ACTION_ID> <ACTION_RESULT>) [4.13]
```

```
<PARMS_RESULT> ::= <CLASS_PARMS> | <PARMS_SET>
```

```
<CLASS_PARMS> ::= parameters_values (<ACTION_ID> {(<PARAM_NAME> [real])})  
[4.14]
```

```
<PARMS_SET> ::= parameters_set_result (<ACTION_ID> <ACTION_RESULT>) [4.15]
```



```
<TIME_RESULT> ::= time_spec (<ACTION_ID> <STEP> <START_TIME> <FINISH_TIME>)
[4.16]
```

```
<SUB_DOMAIN_RESULT> ::= subdomain_result (<ACTION_ID> <ACTION_RESULT>) [4.17]
```

Answers to the actions over the register of the results:

```
<REG_RESULT> ::= <FILE_RESULT> | <REG_VARS_RESULT> | <LOG_RESULT> |
<REG_TIME_RESULT> | <TRACE_RESULT>
```

```
<FILE_RESULT> ::= output_file_result (<ACTION_ID> <ACTION_RESULT>) [4.18]
```

```
<REG_VARS_RESULT> ::= <GET_VARS_RESULT> | <SELECT_VARS_RESULT> |
<UNSELECT_VARS_RESULT>
```

```
<GET_VARS_RESULT> ::= variables_available (<ACTION_ID> {<VAR_NAME>}) [4.19]
```

```
<SELECT_VARS_RESULT> ::= select_variables_result (<ACTION_ID>
<ACTION_RESULT>) [4.20]
```

```
<UNSELECT_VARS_RESULT> ::= unselect_variables_result (<ACTION_ID>
<ACTION_RESULT>) [4.21]
```

```
<LOG_RESULT> ::= log_result (<ACTION_ID> <ACTION_RESULT>) [4.22]
```

```
<REG_TIME_RESULT> ::= output_time (<ACTION_ID> <STEP> <START_TIME>
<FINISH_TIME>) [4.23]
```

```
<TRACE_RESULT> ::= trace_result (<ACTION_ID> <TRACE_STATUS>) [4.24]
```

```
<TRACE_STATUS> ::= on | off
```

Spontaneous messages sent by the simulator:

```
<EVENT_MSG> ::= <REG_MSG> | <LOG_MSG> | <END_MSG>
```

```
<REG_MSG> ::= register (<REG_INDEX> <REG_TIME> <VAR_NAME> {(<CELL> [real])})
[4.25]
```

```
<LOG_MSG> ::= logger (<STEP> {(<CLASS_NAME> <FUNC_TYPE> <DATA_CLASS>
<VAR_NAME> <CELL> [real])}) [4.26]
```

```
<END_MSG> ::= end_simulation | end_step | running | stopped | paused [4.27]
```

```
<REG_INDEX> ::= [integer]
```

```
<REG_TIME> ::= [integer]
```

```
<FUNC_TYPE> ::= Inquiry | Update
```

```
<DATA_CLASS> ::= <CLASS_NAME>
```

2.5 COORDINATION

The coordination messages, defined from version 1.4 of the protocol, are used between agents and can be grouped by type. The first type includes messages for optimization and for parallel simulated annealing (PSA) coordination.

The PSA algorithm defines one coordinator agent and several workers, each one responsible for a group of simulators. The messages are divided into four groups: to define agents' roles, to

configure the simulations and the desired optimization, and to exchange agents' optimization data.

```
<COORDINATION_MSG> ::= <FUNCTION_MSG> | <CONFIGURATION_MSG> | <PROCESS_MSG>
| <OPTIMIZATION_MSG>
```

Each <COORDINATION_MSG> must be answered by one <EXEC_RESULT> [4.8] perception.

- Actions to configure the role of each agent:

```
<FUNCTION_MSG> ::= <COORDINATOR_MSG> | <WORKER_MSG> | <SIMULATORS_MSG>
```

```
<COORDINATOR_MSG> ::= psa_coordinator [5.1]
```

```
<WORKER_MSG> ::= psa_worker <COORDINATOR_NAME> [5.2]
```

```
<SIMULATORS_MSG> ::= psa_simulator ({<SIMULATOR_NAME>}) [5.3]
```

```
<COORDINATOR_NAME> ::= [string]
```

```
<SIMULATOR_NAME> ::= [string]
```

All the <FUNCTION_MSG> messages are sent from the configuration agent to the agents of the PSA process. The message <COORDINATOR_MSG> is sent to the coordinator agent. The message <WORKER_MSG> is sent to the coordinator and worker agents. The message <SIMULATORS_MSG> is sent to each one of the workers.

- Actions to configure the simulations and the optimization process:

```
<CONFIGURATION_MSG> ::= <TEMPERATURE_MSG> | <ITERATIONS_MSG> | <WORKERS_MSG>
| <SYNCH_POINTS_MSG>
```

```
<TEMPERATURE_MSG> ::= psa_temperature (<INITIAL_T> <FINAL_T> <DRATE_T>) [5.4]
```

```
<ITERATIONS_MSG> ::= psa_iterations [integer] [5.5]
```

```
<WORKERS_MSG> ::= psa_workers [integer] [5.6]
```

```
<SYNCH_POINTS_MSG> ::= psa_synchronization_points [integer] [5.7]
```

```
<INITIAL_T> ::= [real]
```

```
<FINAL_T> ::= [real]
```

```
<DRATE_T> ::= [real]
```

All the <CONFIGURATION_MSG> messages are sent from the configuration agent to the agents of the PSA process. The messages <TEMPERATURE_MSG> and the <ITERATIONS_MSG> are sent to the coordinator and worker agents. The messages <WORKERS_MSG> and the <SYNCH_POINTS_MSG> are sent to the coordinator agent.

- Actions to start optimization and to exchange agents' results:

```
<PROCESS_MSG> ::= <PUBLISH_MSG> | <RESULTS_MSG> | <RUN_MSG>
```

```
<PUBLISH_MSG> ::= <PUBLISH_BEST_RESULT> | <PUBLISH_BEST_SOLUTION>
```

```
<PUBLISH_BEST_RESULT> ::= psa_publish_best_result [real] [5.8]
```

```
<PUBLISH_BEST_SOLUTION> ::= psa_publish_best_solution ({<SOLUTION>}) [5.9]
```

```
<RESULTS_MSG> ::= <BEST_RESULT> | <BEST_SOLUTION>
```

```
<BEST_RESULT> ::= psa_best_result [real] [5.10]
```

```
<BEST_SOLUTION> ::= psa_best_solution ({<SOLUTION>}) [5.11]
```

```
<RUN_MSG> ::= psa_run [5.12]
```

```
<SOLUTION> ::= (<CELL> [real])
```

```
<FINAL_T> ::= [real]
```

```
<DRATE_T> ::= [real]
```

```
<CELL> → defined in message [2.13] (<MORPHOLOGY_RESULT>)
```

The message <RUN_MSG> is initially sent from the configuration agent to the coordinator. The <RESULTS_MSG> messages are used by the coordinator and the workers agents to exchange the best results, and the messages <PUBLISH_MSG> are sent from the coordinator agent to all the others participants, including the configuration agent, in the end of the optimization process.

- Actions to configure the optimization:

```
<OPTIMIZATION_MSG> ::= <SIMULATION_MSG> | <SPECIES_CONFIGURATION_MSG>
```

```
<SIMULATION_MSG> ::= simulation_steps [integer] [5.13]
```

```
<SPECIES_CONFIGURATION_MSG> ::= species_optimization ({<BIVALVE_S> <DENSITY>  
  <NR_CELLS> <ECONOMICAL_WEIGHT> <REG_NAME> <OPT_TYPE>}) [5.14]
```

```
<BIVALVE_S> → defined in message [3.1] (<SEED_ACTION>)
```

```
<DENSITY> → defined in message [3.1] (<SEED_ACTION>)
```

```
<NR_CELLS> ::= [integer]
```

```
<ECONOMICAL_WEIGHT> ::= [real]
```

```
<REG_NAME> → defined in message [2.1] (<DEFINE_ACTION>)
```

```
<OPT_TYPE> ::= areas | density
```

All the <OPTIMIZATION_MSG> messages are sent from the configuration agent to the coordinator and worker agents of the PSA process.

3 COMMUNICATIONS PROTOCOL

The communication between the simulator (EcoDynamo application) and the other actors present in the simulation system is usually of the type handshake - a message-type action expects to receive an answer from the destination application; that response comes in the form of a perception type message.

Only the spontaneous messages and the logging results sent by the simulator don't require feedback.

The first message of each agent for the simulator must be connected (connect). The reception of a positive acceptance message (to accept ok result) indicates that the agent was registered in the simulator as an agent interested in obtaining results from the simulations. When the agent leaves the system it must send the message to disconnect from the simulator.

The simulator, before leaving the system, must send to all registered active agents the disconnect message.

The communications' protocol establishes the answers expected for each action, according to the following table:

| [msg] | Message | Expected answer | [msg] |
|--------|--------------------|----------------------------------|--------|
| [1.1] | connect | accept | [1.3] |
| [1.2] | disconnect | accept | [1.3] |
| [1.4] | agents | known_agents | [1.5] |
| [2.1] | define | define_result | [2.3] |
| [2.2] | delete | delete_result | [2.4] |
| [2.5] | get_region_names | region_names | [2.7] |
| [2.6] | get_region | region | [2.8] |
| [2.9] | model_dimensions | dimensions | [2.12] |
| [2.10] | model_morphology | morphology ⁵ | [2.13] |
| [2.10] | model_morphology | morphology_end ⁶ | [2.14] |
| [2.11] | model_species | benthic_species ⁷ | [2.15] |
| [2.11] | model_species | benthic_species_end ⁸ | [2.16] |
| [3.1] | seed | seed_result | [4.1] |
| [3.2] | inspect | inspect_result | [4.2] |
| [3.3] | harvest | harvest_result | [4.3] |
| [3.4] | open_model | open_result | [4.4] |
| [3.5] | close_model | close_result | [4.5] |
| [3.6] | model_name | model | [4.6] |
| [3.7] | save_configuration | save_result | [4.7] |

⁵ This is the answer while there were messages to send from morphology: morphology of each message has, at most, 750 elements.

⁶ This is the answer indicating end of morphology messages.

⁷ This is the answer while there were messages to send from benthic species: each benthic species message has, at most, 150 elements.

⁸ This is the answer indicating end of benthic species messages.

| | | | |
|--------|-------------------------|---------------------------|--------|
| [3.8] | initialise | exec_result | [4.8] |
| [3.8] | run | exec_result | [4.8] |
| [3.8] | stop | exec_result | [4.8] |
| [3.8] | pause | exec_result | [4.8] |
| [3.9] | step | exec_result | [4.8] |
| [3.10] | get_available_classes | classes_available | [4.9] |
| [3.10] | get_selected_classes | classes_selected | [4.10] |
| [3.11] | select_classes | classes_selected | [4.10] |
| [3.12] | get_variables | variables | [4.11] |
| [3.13] | get_variable_value | variable_value | [4.12] |
| [3.14] | set_variable_value | variable_set_result | [4.13] |
| [3.15] | get_parameters | parameters_values | [4.14] |
| [3.16] | set_parameters | parameters_set_result | [4.15] |
| [3.17] | get_time_spec | time_spec | [4.16] |
| [3.18] | set_time_spec | time_spec | [4.16] |
| [3.19] | subdomain | subdomain_result | [4.17] |
| [3.20] | output_file | output_file_result | [4.18] |
| [3.21] | get_available_variables | variables_available | [4.19] |
| [3.22] | select_variables | select_variables_result | [4.20] |
| [3.23] | unselect_variables | unselect_variables_result | [4.21] |
| [3.24] | log | log_result | [4.22] |
| [3.25] | get_output_time | output_time | [4.23] |
| [3.26] | set_output_time | output_time | [4.23] |
| [3.27] | trace | trace_result | [4.24] |
| [4.25] | register | -- | |
| [4.26] | logger | -- | |
| [4.27] | end_simulation | -- | |
| [4.27] | end_step | -- | |
| [4.27] | running | -- | |
| [4.27] | stopped | -- | |
| [4.27] | paused | -- | |
| [5.1] | psa_coordinator | exec_result | [4.8] |
| [5.2] | psa_worker | exec_result | [4.8] |
| [5.3] | psa_simulator | exec_result | [4.8] |
| [5.4] | psa_temperature | exec_result | [4.8] |
| [5.5] | psa_iterations | exec_result | [4.8] |
| [5.6] | psa_workers | exec_result | [4.8] |

| | | | |
|--------|----------------------------|-------------|-------|
| [5.7] | psa_synchronization_points | exec_result | [4.8] |
| [5.8] | psa_publish_best_result | exec_result | [4.8] |
| [5.9] | psa_publish_best_solution | exec_result | [4.8] |
| [5.10] | psa_best_result | exec_result | [4.8] |
| [5.11] | psa_best_solution | exec_result | [4.8] |
| [5.12] | psa_run | exec_result | [4.8] |
| [5.13] | simulation_steps | exec_result | [4.8] |
| [5.14] | species_optimization | exec_result | [4.8] |

Annex 4 – Farmer Agent Configuration

Configuration Files

Farmer Agent (FA) is configured by the user to define the goals, tactics and constraints that will initialize the optimization processes. Several types of files are considered by FA.

SIMULATED ANNEALING FILE

File with one line defining the parameters to control the Simulated Annealing algorithm. This file usually has the extension “.sa”, only for convenience. Line example:

```
sa IT FT DR NB NI NT CF HC
```

where:

```
sa - indicates SA parameters following
IT - initial temperature
FT - final temperature
DR - multiplicative factor to adjust temperature value in each iteration
NB - neighbourhood breadth distance
NI - maximum number of iterations
NT - number of solutions in the list of the best solutions
CF - use cache files to save results [0 - do not use; 1 - use cache]
HC - hill climbing [0 - simulated annealing; 1 - hill climbing]
```

Example:

```
sa 1.0 0.0001 0.980 2 341 10 1 0
```

TACTICS FILE

Tactics file, also referred as toggles file, define the initial tactic used by FA. For convenience, this file usually uses the extension “.tg/”, but this is not mandatory. The algorithms selected to influence the optimization process are situated between the lines “init” and “eof”. Each line references one algorithm and indicates the moment of actuation or inactivation. The general format of the line is:

```
tog ALG ET ON P1 P2 P3
```

where:

```
tog - indicates algorithm line
```

ALG - algorithm [0- FarmerTabu; 1 - FarmerGA; 2 - FarmerRL]
ET - entry time of the algorithm (percentage of initial temperature)
ON - activate/deactivate [0 - deactivate; 1 - activate]
P1 - first parameter for the algorithm (algorithm dependant)
P2 - second parameter for the algorithm (algorithm dependant)
P3 - third parameter for the algorithm (algorithm dependant) - optional

Example of one toggles file:

```
init
tog 0 0.0 1 0.8 1
tog 1 0.4 1 15 5
tog 2 0.5 3 0.05
eof
```

OPTIMIZATION PROCESS CONFIGURATION FILE

The complete configuration of the optimization process is grouped in a text file (*config.cfg*) that gathers the duration of the simulations (number of steps) and the type of optimization desired to the algorithms previously mentioned.

Example of *config.cfg* file:

```
initSA 1.0 0.0001 0.980 2 341 10 1 0
simulation 86400 0
togglesfile firstTGL.tgl
species oyster 0.0005 0.00002 55.10 30 1.0 inner-outer.sol areas
species scallop 1.192300 0.152070 56.50 30 2.0 inner-outer.sol areas
```

The first line of the file indicates the SA parameters and can be replaced by:

```
initSA <SA filename>
```

The second line (started with *simulation* word) indicates the number of simulation time steps and the intention to repeat simulations of known solutions [0 – do not repeat; 1 – repeat].

The third line (started with *togglesfile* word) indicates the name of the toggles file.

Each line starting with *species* keyword indicates that optimization is related with aquaculture, and have the general format:

```
species SP W1 W2 SD NC EW DF OT
```

where:

SP - species common name
W1 - initial dry weight for species individuals
W2 - initial meat dry weight for species individuals

SD - seed density
NC - number of cells to seed
EW - economical weight of the species
DF - name of the file with the domain areas
OT - optimization type [areas or density]

TASKS FILE

It is possible to pre-program several tasks to FA. Each line of the tasks file (named "*tasklist.tsk*") contains the name of a *config* file, and the optimization process is performed sequentially. The start command to the FA is "Run tasks" instead of "Run".

File System Structure

The file system structure hierarchy used by the Farmer Agent was designed to save the configuration files and the results, providing each optimization performed with a unique identification. This allows the reutilization of previous results for repeated optimization configurations. The Figure A4-1 represents the file system hierarchy.

The base folder for all the files is named "FarmerData". By default, this folder contains all the files that contain the solutions' domains, the configuration of the algorithms, the saved tactics and the predefined tasks used by the FA.

When an optimization runs, all the internal FA data is saved in one folder named "data". Each optimization has one folder with a unique identification that distinguishes the configuration used. The name of the folder must contain the fields:

- The model name;
- The number of simulation steps used;
- The identification of the optimization type and the parameterization of the bivalve species optimized.

All the fields are separated by hyphen and the last one is repeated for each benthic species. This folder contains files named "solutionX.dat", with the optimization results ordered by the best to the worst, where the file "solutions0.dat" contains the description of the 100 best solutions, the file "solution1.dat" the next 100 solutions, and so on.

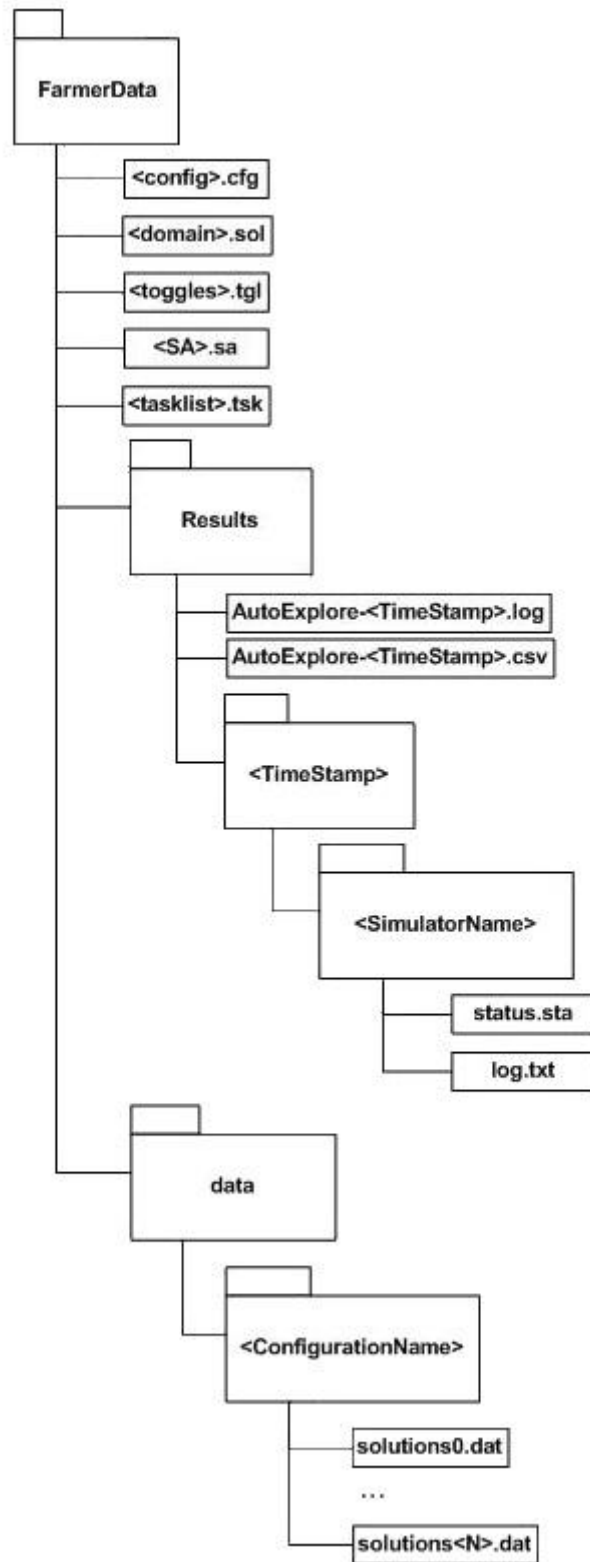


Figure A4-1: Farmer Agent File System Hierarchy

The folder “Results” contains the results of each optimization run, named with its initial time. Record files are named “AutoExplore-<TimeStamp>”:

- AutoExplore-<TimeStamp>.csv – contains the results of each iteration

- AutoExplore-<TimeStamp>.log – contains the log of the optimization process.

<TimeStamp> has the form AAAAMMDDhhmmss, where:

- AAAA – is the four-digit year
- MM – is the two-digit month number (from 01 to 12)
- DD – is the two-digit day of month
- hh – is the two-digit hour of the day (from 00 to 23)
- mm – is the two-digit minute
- ss – is the two-digit seconds.

Each <TimeStamp> folder has folders with the names of the simulators used for the optimization and, within them, two files with information related with each simulator process.