NORTH WEST SHELF
JOINT ENVIRONMENTAL
MANAGEMENT STUDY

Ecosystem model specification with an agent based framework

NWSJEMS TECHNICAL REPORT No. 16

• R. Gray • E. Fulton • R. Little • R. Scott

June 2006

CSIRO

Government of Western Australia
Department of Environment

National Library of Australia Cataloguing-in-Publication data:

Ecosystem model specification within an agent based framework.

Bibliography.
Includes index.
ISBN 1 921061 80 4 (pbk.).

1. Ecosystem management. I. Gray, R. (Randall). II. CSIRO. Marine and Atmospheric Research. North West Shelf Joint Environmental Management Study. III. Western Australia. (Series : Technical report (CSIRO. Marine and Atmospheric Research. North West Shelf Joint Environmental Management Study) ; no. 16).

577.7

Ecosystem model specification within an agent based framework.

Bibliography.
Includes index.
ISBN 1 921061 81 2 (CD-ROM).

1. Ecosystem management. I. Gray, R. (Randall). II. CSIRO. Marine and Atmospheric Research. North West Shelf Joint Environmental Management Study. III. Western Australia. (Series : Technical report (CSIRO. Marine and Atmospheric Research. North West Shelf Joint Environmental Management Study) ; no. 16).

577.7

Ecosystem model specification within an agent based framework.

Bibliography.
Includes index.
ISBN 1 921061 82 0 (pdf).

1. Ecosystem management. I. Gray, R. (Randall). II. CSIRO. Marine and Atmospheric Research. North West Shelf Joint Environmental Management Study. III. Western Australia. (Series : Technical report (CSIRO. Marine and Atmospheric Research. North West Shelf Joint Environmental Management Study) ; no. 16).

577.7

NORTH WEST SHELF JOINT ENVIRONMENTAL MANAGEMENT STUDY

Final report

North West Shelf Joint Environmental Management Study Final Report.

List of technical reports

NWSJEMS Technical Report No. 1
Review of research and data relevant to marine environmental management of Australia's North West Shelf.
A. Heyward, A. Revill and C. Sherwood

NWSJEMS Technical Report No. 2
Bibliography of research and data relevant to marine environmental management of Australia's North West Shelf.
P. Jernakoff, L. Scott, A. Heyward, A. Revill and C. Sherwood

NWSJEMS Technical Report No. 3
Summary of international conventions, Commonwealth and State legislation and other instruments affecting marine resource allocation, use, conservation and environmental protection on the North West Shelf of Australia.
D. Gordon

NWSJEMS Technical Report No. 4
Information access and inquiry.
P. Brodie and M. Fuller

NWSJEMS Technical Report No. 5
Data warehouse and metadata holdings relevant to Australia's North West Shelf.
P. Brodie, M. Fuller, T. Rees and L. Wilkes

NWSJEMS Technical Report No. 6
Modelling circulation and connectivity on Australia's North West Shelf.
S. Condie, J. Andrewartha, J. Mansbridge and J. Waring

NWSJEMS Technical Report No. 7
Modelling suspended sediment transport on Australia's North West Shelf.
N. Margvelashvili, J. Andrewartha, S. Condie, M. Herzfeld, J. Parslow, P. Sakov and J. Waring

NWSJEMS Technical Report No. 8
Biogeochemical modelling on Australia's North West Shelf.
M. Herzfeld, J. Parslow, P. Sakov and J. Andrewartha

NWSJEMS Technical Report No. 9
Trophic webs and modelling of Australia's North West Shelf.
C. Bulman

NWSJEMS Technical Report No. 10
The spatial distribution of commercial fishery production on Australia's North West Shelf.
F. Althaus, K. Woolley, X. He, P. Stephenson and R. Little

NWSJEMS Technical Report No. 11
Benthic habitat dynamics and models on Australia's North West Shelf.
E. Fulton, B. Hatfield, F. Althaus and K. Sainsbury

NWSJEMS Technical Report No. 12
Ecosystem characterisation of Australia's North West Shelf.
V. Lyne, M. Fuller, P. Last, A. Butler, M. Martin and R. Scott

NWSJEMS Technical Report No. 13
Contaminants on Australia's North West Shelf: sources, impacts, pathways and effects.
C. Fandry, A. Revill, K. Wenziker, K. McAlpine, S. Apte, R. Masini and K. Hillman

NWSJEMS Technical Report No. 14
Management strategy evaluation results and discussion for Australia's North West Shelf.
R. Little, E. Fulton, R. Gray, D. Hayes, V. Lyne, R. Scott, K. Sainsbury and D. McDonald

NWSJEMS Technical Report No. 15
Management strategy evaluation specification for Australia's North West Shelf.
E. Fulton, K. Sainsbury, D. Hayes, V. Lyne, R. Little, M. Fuller, S. Condie, R. Gray, R. Scott, H. Webb, B. Hatfield, M. Martin, and D. McDonald

NWSJEMS Technical Report No. 16
Ecosystem model specification within an agent based framework.
R. Gray, E. Fulton, R. Little and R. Scott

NWSJEMS Technical Report No. 17
Management strategy evaluations for multiple use management of Australia's North West Shelf – Visualisation software and user guide.
B. Hatfield, L. Thomas and R. Scott

NWSJEMS Technical Report No. 18
Background quality for coastal marine waters of the North West Shelf, Western Australia.
K. Wenziker, K. McAlpine, S. Apte, R.Masini

CONTENTS

ACRONYMS

ACOM	Australian Community Ocean Model
AFMA	Australian Fisheries Management Authority
AFZ	Australian Fishing Zone
AGSO	Australian Geological Survey Organisation now Geoscience Australia
AHC	Australian Heritage Commission
AIMS	Australian Institute of Marine Science
AMSA	Australian Maritime Safety Authority
ANCA	Australian Nature Conservation Agency
ANZECC	Australian and New Zealand Environment and Conservation Council
ANZLIC	Australian and New Zealand Land Information Council
APPEA	Australian Petroleum, Production and Exploration Association
AQIA	Australian Quarantine Inspection Service
ARMCANZ	Agricultural Resources Management council of Australia and New Zealand
ASIC	Australian Seafood Industry Council
ASDD	Australian Spatial Data Directory
CAAB	Codes for Australian Aquatic Biota
CAES	Catch and Effort Statistics
CALM	Department of Conservation and Land Management (WA Government)
CAMBA	China Australia Migratory Birds Agreement
CDF	Common data format
CITIES	Convention on International Trade in Endangered Species
CTD	conductivity-temperature-depth
CMAR	CSIRO Marine and Atmospheric Research
CMR	CSIRO Marine Research
COAG	Council of Australian Governments
ConnIe	Connectivity Interface
CPUE	Catch per unit effort
CSIRO	Commonwealth Science and Industrial Research Organisation
DCA	detrended correspondence analysis
DIC	Dissolved inorganic carbon
DISR	Department of Industry, Science and Resources (Commonwealth)
DEP	Department of Environmental Protection (WA Government)
DOM	Dissolved organic matter
DPIE	Department of Primary Industries and Energy
DRD	Department of Resources Development (WA Government)
EA	Environment Australia
EEZ	Exclusive Economic Zone
EIA	Environmental Impact Assessment
EPA	Environmental Protection Agency
EPP	Environmental Protection Policy
ENSO	El Nino Southern Oscillation
EQC	Environmental Quality Criteria (Western Australia)
EQO	Environmental Quality Objective (Western Australia)
ESD	Ecologically Sustainable Development
FRDC	Fisheries Research and Development Corporation
FRMA	Fish Resources Management Act
GA	Geoscience Australia formerly AGSO
GESAMP	Joint Group of Experts on Scientific Aspects of Environmental Protection
GIS	Geographic Information System
ICESD	Intergovernmental Committee on Ecologically Sustainable Development
ICS	International Chamber of Shipping
IOC	International Oceanographic Commission
IGAE	Intergovernmental Agreement on the Environment
ICOMOS	International Council for Monuments and Sites

IMO	International Maritime Organisation
IPCC	Intergovernmental Panel on Climate Change
IUNC	International Union for Conservation of Nature and Natural Resources
IWC	International Whaling Commission
JAMBA	Japan Australian Migratory Birds Agreement
LNG	Liquified natural gas
MarLIN	Marine Laboratories Information Network
MARPOL	International Convention for the Prevention of Pollution from Ships
MECO	Model of Estuaries and Coastal Oceans
MOU	Memorandum of Understanding
MPAs	Marine Protected Areas
MEMS	Marine Environmental Management Study
MSE	Management Strategy Evaluation
NCEP - NCAR	National Centre for Environmental Prediction – National Centre for Atmospheric Research
NEPC	National Environmental Protection Council
NEPM	National Environment Protection Measures
NGOs	Non government organisations
NRSMPA	National Representative System of Marine Protected Areas
NWQMS	National Water Quality Management Strategy
NWS	North West Shelf
NWSJEMS	North West Shelf Joint Environmental Management Study
NWSMEMS	North West Shelf Marine Environmental Management Study
ICIMF	Oil Company International Marine Forum
OCS	Offshore Constitutional Settlement
PFW	Produced formation water
P(SL)A	Petroleum (Submerged Lands) Act
PSU	Practical salinity units
SeaWIFS	Sea-viewing Wide Field-of-view Sensor
SOI	Southern Oscillation Index
SMCWS	Southern Metropolitan Coastal Waters Study (Western Australia)
TBT	Tributyl Tin
UNCED	United Nations Conference on Environment and Development
UNCLOS	United Nations Convention of the Law of the Sea
UNEP	United Nations Environment Program
UNESCO	United Nations Environment, Social and Cultural Organisation
UNFCCC	United Nations Framework Convention on Climate Change
WADEP	Western Australian Department of Environmental Protection
WADME	Western Australian Department of Minerals and Energy
WAEPA	Western Australian Environmental Protection Authority
WALIS	Western Australian Land Information System
WAPC	Western Australian Planning Commission
WHC	World Heritage Commission
WOD	World Ocean Database
www	world wide web

TECHNICAL SUMMARY

Agent-based modelling and the management strategy evaluation approach are increasingly popular techniques used in natural resource management. Agent-based modelling is a technique that is used in a variety of fields (spanning the sciences, social services, humanities and economics) while management strategy evaluation (MSE) is a decision support framework that can be extended from its single sector roots into the exploration of more interdisciplinary questions and issues. Tying the two together, by building agent-based models for use in MSE studies has immense potential for increasing the breadth of natural resource management questions that can usefully be considered with any degree of success.

The MSE framework is centred upon an operating model made up of biophysical and resource exploitation sub-models (that attempts to capture the dynamics of the entire system to be managed). Combined with this operating model are sub-models that simulate the management processes (including simulation of observations, assessments, decision making and implementation). Together, these models attempt to represent how the ecosystem is effected, monitored and managed. This simulation approach provides a means for pinpointing key research areas and (potentially more importantly) highlighting performance tradeoffs between alternative management strategies across a range of management objectives in the face of uncertainty (Sainsbury, 1988; Sainsbury, 1991; Sainsbury et al. 1997; Punt et al. 2001; Fulton et al. 2006b).

The track record of management strategy evaluation (which models each part of the resource-management loop) in single sector management is now fairly extensive (Butterworth & Punt, 1999; Sainsbury et al. 1997; Sainsbury et al. 2000; Punt et al. 2001); the record for agent-based resource modelling is not as long, but it is diverse. It is an exceptionally flexible modelling approach that is a useful tool for situations in which small scale or individual level variability may play a significant role in the outcome. This is the case in some natural resource questions and is increasingly the case as management issues have to span more questions (such as the fate of threatened, endangered or protected species).

One disadvantage to the agent-based modelling technique is the computational overheads involved – this can make some large scale questions infeasible if pure decision based agents are used to represent all aspects of a system. New hybrid methods that tie classical dynamic (differential equation) models with decision based agents seem to be the best means of solving these dilemmas. *NWS-InVitro* is an example of just such a hybrid model. It simulates effects of human activity, such as fishing, tourism, and LNG or salt production on the regional ecosystem, and the response of the system to different management regimes. The model does not attempt to represent all ecosystem details, but focuses on the dominant system components, in this case the North West Shelf ecosystem of Australia. These include commercially valuable fish and crustaceans, sharks, turtles, benthic communities, seagrass and mangroves and with the major industries in the area – fisheries, shipping, oil and gas production, salt extraction, coastal development, port maintenance (such as dredging) and recreational activities, such as fishing. The remainder of this document details the formulation of *NWS-InVitro*.

1. INTRODUCTION TO AGENT STRUCTURE

Agent-based modelling is an increasingly popular modelling technique, used in a variety of fields as disparate as sociology (Epstein, 2002), law enforcement (Epstein, 2002), archaeology (Axtell et al. 2002), ecology (Grimm, 1999) and economics (Tesfatsion, 2001). It is an exceptionally useful tool for situations in which the behavioural differences among individuals in a population may play a significant role in the outcome, and have particular application to natural resource modelling which usually deals with a relatively small number of agents, with relatively large variability among individuals – the form of dynamics typically not easily captured by traditional equation-based models. A down-side to the technique is that agent-based models are computationally demanding and as the questions addressed with natural resource models broaden, new hybrid methods need to be considered. *NWS-InVitro* is one example of such a hybrid modelling exercise combining the benefits of individuality in agent-based models and the computationally efficient models based on traditional analytical equations.

The body of this report details the formulation of *NWS-InVitro*. Naming conventions for the parameters and variables used in the equations are given in Appendix A, an outline of the C++ implementation of *NWS-InVitro* is given in Appendix B and a summary of the major assumptions currently in *NWS-InVitro* is given in Appendix C.

1.1 Background to NWS-InVitro

It is easier to understand the detailed content of the *NWS-InVitro* model when one has an appreciation for the context and structure of the overall model. With this in mind, the remainder of this section provides a brief outline of the form of *NWS-InVitro*.

Before beginning the general introduction to *NWS-InVitro*, a note on terminology is necessary. In this document the term “sub-model” will usually refer to either an equation-based representation or a process-based representation of a component of the model. The term “agent” will usually refer to an instance of that sub-model within the overall model. Where there is only one instance of a given sub-model, the terms are used interchangeably.

1.1.1 NWS-InVitro – a Management Strategy Evaluation tool

NWS-InVitro is the basis for an integrated ecosystem-level management strategy evaluation (MSE) on the north-west coast of Australia (figure 1.1.1). The model simulates effects of human activity, such as fishing, tourism, and LNG or salt production on the regional ecosystem, and the response of the system to different management regimes. Even with *NWS-InVitro’s* flexibility, representing the entire ecosystem is beyond the capabilities of the model and so the primary focus was constrained to the dominant system components. The components include commercially valuable fish and crustaceans, sharks, turtles, benthic communities, seagrass and mangroves – and the major industries in the area – fisheries, shipping, oil and gas production, salt extraction, coastal development, port maintenance (such as dredging) and recreational activities, such as fishing. This combination of components allows for an analysis of the effectiveness and robustness of various strategies for managing the main human interactions with the regional ecosystem on the North West Shelf of Australia.

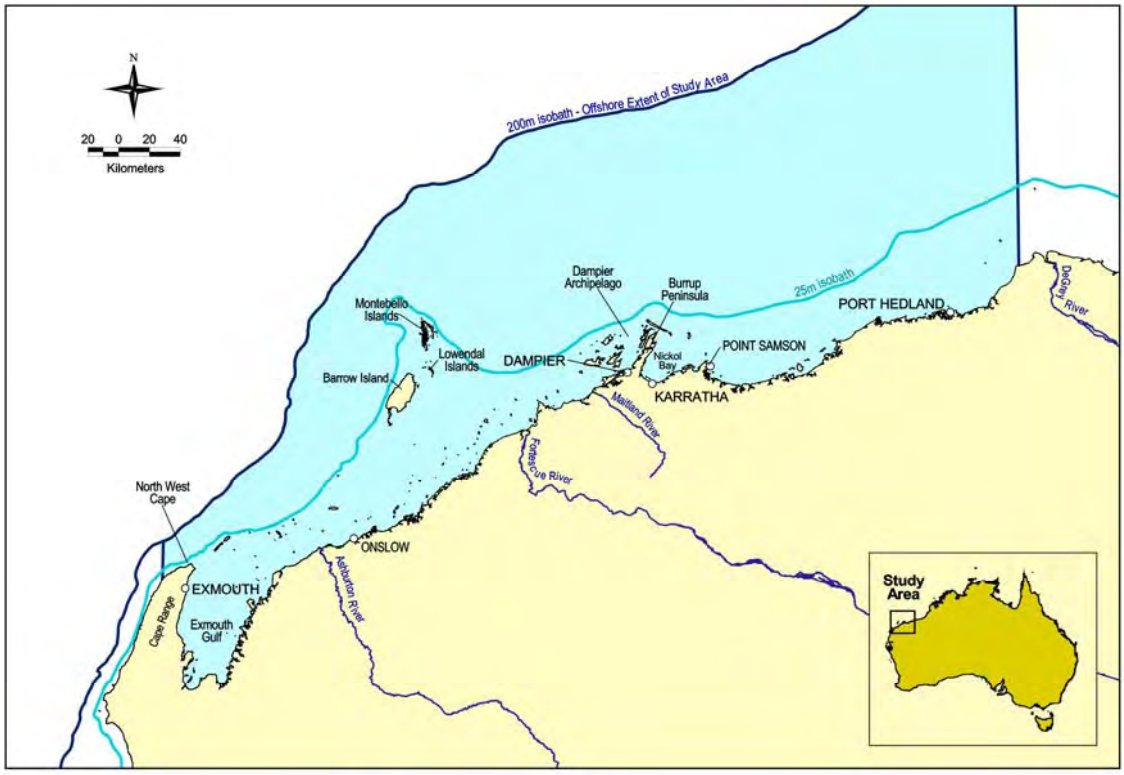

Figure 1.1.1: North West Shelf of Australia – the model domain for NWS-InVitro.

MSE is a simulation approach which has been used with some success within individual sectors (Butterworth & Punt, 1999; Sainsbury et al. 1997; Sainsbury et al. 2000; Punt et al. 2001), although *NWS-InVitro* marks its first use in an integrated multiple use management context. The MSE approach models each part of the resource-management loop (figure 1.1.2) – at the core of which is an operating model that captures critical aspects of the dynamics of natural resources and their exploitation. The other key parts of the MSE approach are the simulated management activities – observation, assessment, decision and implementation. Between them, all of these models represent how the ecosystem is impacted, monitored and managed. By simulating all of these steps explicitly the MSE approach highlights the tradeoffs in the performance of alternative management strategies across a range of management objectives given associated uncertainties in prediction. In turn, this means MSE is a useful tool for identifying key gaps in scientific understanding of system dynamics and for developing adaptive monitoring and management strategies (Sainsbury, 1988; Sainsbury, 1991; Sainsbury et al. 1997; Punt et al. 2001; Fulton et al. 2006b).

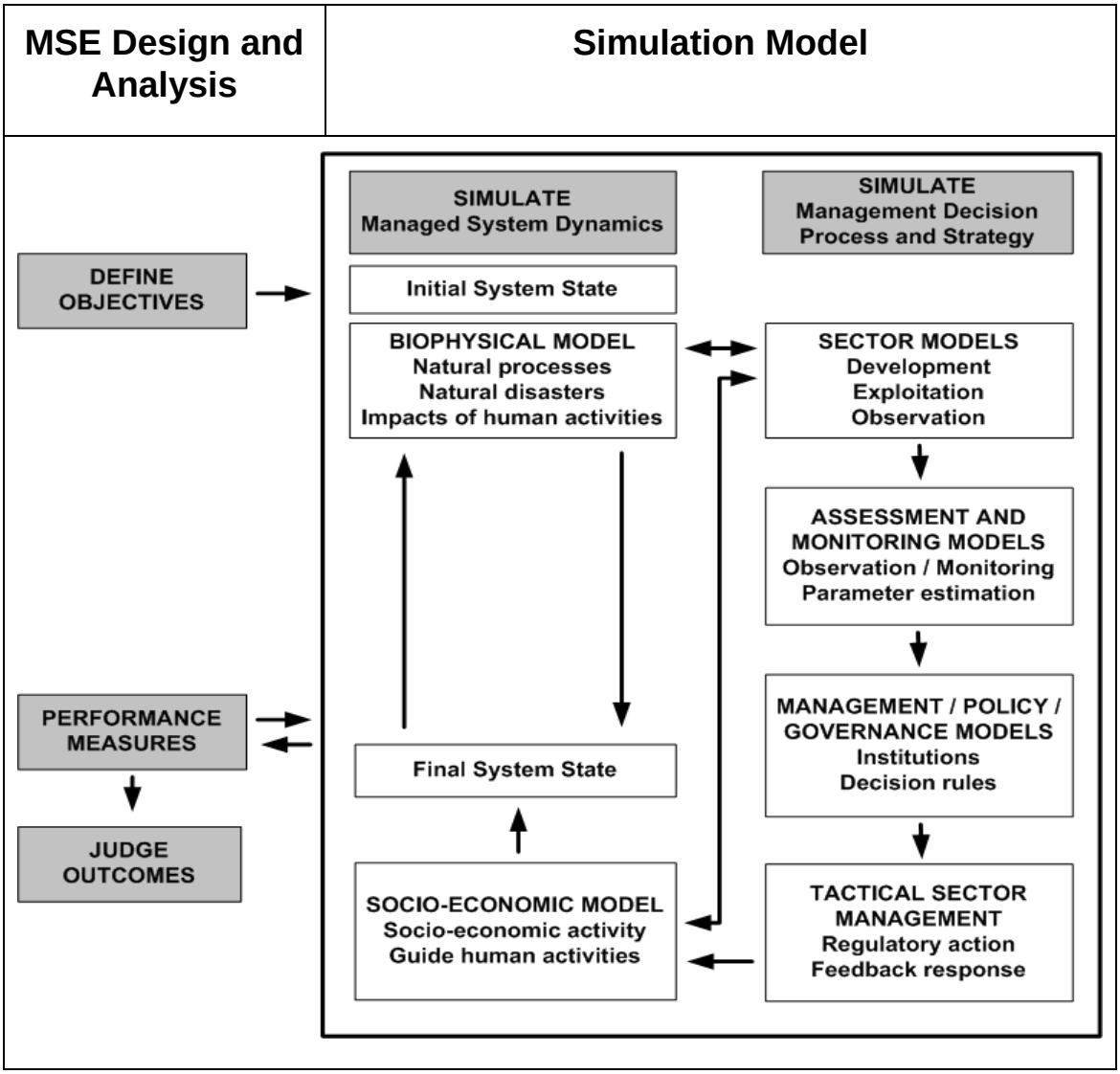

Figure 1.1.2: The MSE Framework. Each of the boxes represent various sub-models. Under the MSE nomenclature the biophysical, sector and socio-economic models can be jointly referred to as the operating model.

The various agent types in *NWS-InVitro* constitute the principal MSE model components:

- the natural biophysical ecosystem
- resource exploitation or modification by each of the sectors (industries)
- monitoring activities
- the management decision and implementation for each sector

Biophysical model

The biophysical ecosystem model at the centre of *NWS-InVitro* makes up the core of this MSE's operating model. It includes: bathymetry, currents, wind, waves, cyclones, seabed types, biogenic habitat groups (reef forming species, seagrass, macroalgae and mangroves), turtles, forage fish, prawn species, the primary target and non-target

finfish, and sharks. While some of these agents are represented at the individual level (turtles) others are aggregates (patches of bottom habitat, sub-populations of finfish, schools of sharks and prawn boils). Functional and physical attributes are detailed for each of these agents and rules are specified for passive and active movement, growth and mortality at the appropriate scales. The agents evaluate the environmental features and community make-up at their location and make the appropriate (temporal and spatial) responses.

Sector models

The sector (industry) models emulate the activities of each major industry in the North West Shelf region. These actions can put pressure on the agents making up the bio-physical model, potentially altering their state. The four sectors included in *NWS-InVitro* are oil and gas, fisheries, conservation and coastal development. The agents used to represent these sectors produce simulated goods and services and are directed by regulations from the relevant simulated management agencies.

The oil and gas sector model emulates the main activities of petroleum companies in the discovery and exploitation of oil and gas reserves. In particular, the release of spilled petroleum products and plumes associated with production such as drilling mud plumes and produced formation water (water which was trapped in the petroleum or surrounding rock).

The fishery sector is represented by models of the recreational line fishery, and commercial finfish trawl, trap-and-line, and prawn trawl fisheries. The recreational fishery is represented at a gross level with effects that are dependent on the distance from access points (e.g. boat ramps) and population centres. The commercial fisheries sub-model is much more elaborate, using methods of Bayesian updating to emulate the decisions and fishing practices of individual fishers.

While the conservation sector is considered in *NWS-InVitro*, it is not modelled explicitly. Conservation instead is largely handled through management regulations applied in the various sectors under the alternative management strategies and scenarios.

The coastal development sector provides a context and facilities for the other sectors and the management agencies. The direct impacts of coastal development on the ecosystem are represented explicitly in the biophysical model as: circulation of discharges (such as industrial waste, sewage, saline bitterns and cooling water) and associated toxicity effects; and habitat modification due to physical disturbance by infrastructure maintenance (such as dredging, construction and production processes).

Monitoring and assessment models

The biophysical agents are observed imperfectly by observation sub-models that feed simulated data to each modelled sector. This simulated data may be used directly by the management model, or input into assessment models that feed in turn into the management decision process. To date, the fisheries assessment models in *NWS-InVitro* are the only elaborated assessment models (based on the assessment model by the Western Australian Department of Fisheries) – all other sectors use simple indices (as assessment models are not yet the norm in these other sectors).

Management models

The management sub-models use the output of the assessment models or other indicators to make decisions about the location and magnitude of the sectors' activities. These regulations constrain the sector activities and resulting impacts, though not necessarily with complete compliance. The decision rules implemented in *NWS-InVitro* are quantitative interpretations of objectives identified in discussions with stakeholders and management bodies in Western Australia. At present, three main management control variables are considered for each sector – spatial zoning structure, the types of activity permitted overall or per zone, and the levels permitted for those activities.

NWS-InVitro as a complete package

NWS-InVitro incorporates all four of the models that make up the core of the MSE simulation. It operates over the scale of an entire regional ecosystem, simulating the regional environment, the exploitation or use of the local resources, the monitoring of the system and the management decisions and implementations. This leads to a highly connected web of interdependent sub-models (figure 1.1.3). By considering the dynamics of this web, the relative strengths and weaknesses of different monitoring and management regimes can be compared across a range of conditions.

Ultimately *NWS-InVitro* is a tool that provides a means of integrating sub-models of different parts of the resource-management loop that operate at different scales and according to different rules. Using *NWS-InVitro* sub-models in simple forms have been cast that focus on the behaviour of the agents rather than the means of integrating them with each other. The shift from a structural to behavioural perspective, in combination with the realisation that there is a spectrum of model types between individual-based and mathematical models, provides immense flexibility and an efficient means of modelling complex structures such as regional marine ecosystems.

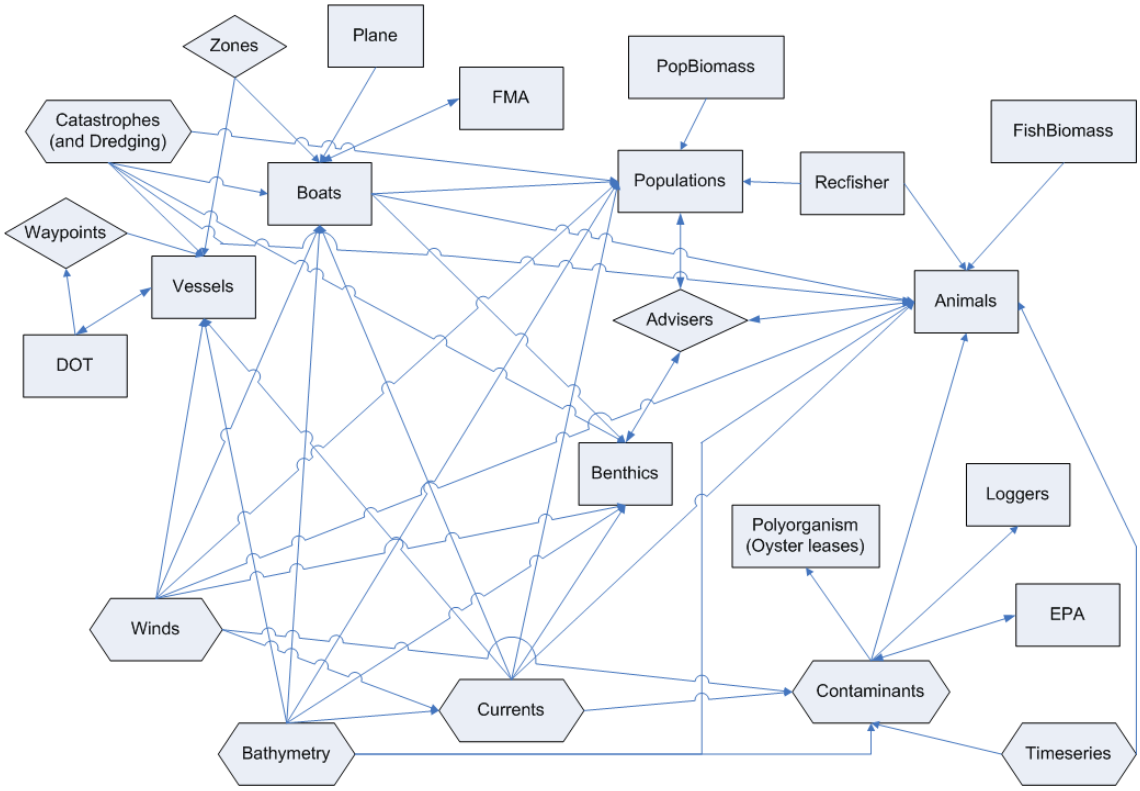

Figure 1.1.3: Schematic diagram of the linkages and dependencies within *NWS-InVitro* (detailed agents are described in the following chapters, while the most basic agent types are described in the remainder of this chapter). The shape of the agent type gives an indication of their role in the network: hexagonal shapes are abiotic data layers or time series (so aren't typically changed dynamically by other agents, but alternative sets can be loaded through the simulation); diamonds indicate monitor agents that only pass information to/between agents, but aren't dynamic in their own right (e.g. "zones" demarcate fishing zones and "advisers" give large scale summaries of the state of the system, which can then be used as a basis for movement decisions); the rectangles represent dynamic agents (the bulk of agents of real interest in the model).

1.2 Space, time and integrating the sub-models

NWS-InVitro ties together a number of disparate models, each with their own optimal temporal and spatial scales. Coercing all these models to some common scales in time and space is inefficient both from the point of view of computational efficiency and of suitability. As an example, while one may wish to model the predatory interactions of a fishing vessel and schools of fish at quite a short time scale, that scale is wholly inappropriate for modelling things like sponges. Similarly, the spatial scale at which the models operate must be appropriate to the model itself: a model of plume dynamics operates at a different spatial scale to a model of stock dynamics.

These differences would typically make the exchange of information between models awkward (in the case of the spatial scale) or make the interaction impossible (in the case of the temporal scale). This conundrum is dealt with in a fairly straightforward fashion: all models are embedded in a continuous three dimensional space, and each model is required to be able to run for an arbitrary amount of time from any point in time.

For most classical models the process of embedding them in this *model space* requires some reformulation, mainly with respect to the ability to update the model state for arbitrary time-steps. The mapping between the spatial component *data space* and the equivalent in model space is typically much more straightforward, and *NWS-InVitro* does this using routines from the PROJ4 library. The projections used map into a models space which uses meters as its natural unit.

1.2.1 Spatial interactions

Locations within the model are three-dimensional floating point vectors, usually corresponding to some sort of east-west ordinate, a north-south ordinate, and a depth. The resolution of these numbers is constrained by the natural representation of "doubles" in C using the GCC compiler (the programming language and compiler used in the development of *NWS-InVitro*): in the case of the *NWS-InVitro* model, this in the case of the *NWS-InVitro* model, this means the model can theoretically span distances from femtometres (10^{-15} m) to many many times the diameter of the universe – in practice however, only distances on the order of 1 cm to a 1000 km were necessary. The degree of "aliasing" due to quantisation errors is thus extremely small in the individual-based sub-models. While some of the models implemented were gridded and were subject to this sort of aliasing, they could have been filtered or interpolated to provide smooth transitions between grid cells if the perceived benefit had justified the computational cost – which it did not in this instance.

Generally an agent will perform most of its operations in its native ordinate space (the data space). This is usually done for reasons of computational efficiency or clarity. Whenever an agent seeks information from another agent about a specific location, it must first map its own representation of the location (in its data space) to a location in the common model space. This is then passed to the agent servicing the request, who must first convert the location from the model space to its own data space, then generate the answer required, and finally return it.

1.2.2 Temporal interactions

NWS-InVitro manages a number of heterogeneous sub-models running in a largely asynchronous fashion. This requires careful attention to the management of the order in which sub-models are given time-steps and the amount of time they are given. While the basic temporal quantum is one second, sub-models would rarely actually use a time-step of a second. Typically a model will have its own notion of an "optimal" time-step, and where ever possible it will use that time-step. Occasionally sub-models will have to interact to exchange information. Some of these exchanges are not dependent on time (such as the location of a pipeline or rocky outcrop) and these exchanges may occur without difficulty. If, however, the information is dependent on the subjective time of one or the other of the agents, then these agents must become synchronous before the interaction can occur. This will typically cause one or the other of the agents to schedule a short time-step that brings them into synchrony.

NWS-InVitro uses a set of priority queues to handle the careful interleaving of sub-models. Nevertheless, some attention must be given within the sub-models to differentiate between those interactions that are, and those that are not, dependent on time. For instance, there are fine scale interactions (like those between predator and prey) that are on fine time scales and need very careful handling; but there are other interactions (e.g. fish searching through a sponge bed) an agent can make that are with agents whose state changes so slowly relative to the natural time-step of the first agent that the interactions can be treated as being independent of time.

1.3 Agents and model taxonomy

NWS-InVitro is an agent-based model and as such the core of the model is the Agent. This agent class provides the core for all the other components of *NWS-InVitro*. A hierarchy of agent types exists (described below), which fleshes out the various sub-models forming the body of *NWS-InVitro*. These sub-models do not function in isolation however. They are linked by the backbone of the *InVitro* modelling framework, the Scheduler (and related queues) and the Neighbourhood grid. An outline of this basic structure is given in the following sections of this chapter. A brief outline of the highest level agents is also given here, but the bulk of the details can be found in the descriptions of the derived (lower-level) agent types that are given in the remainder of this report.

The basis for the *NWS-InVitro* framework is the realisation that an equation-based model is nothing more than an agent-based model with a single agent. With that concept in place the construction of a full ecosystem model like *NWS-InVitro* becomes an exercise in structuring the content of the agents (individual-based or equation-based) and their interactions so that an optimal representation is produced while avoiding temporal and spatial anomalies. The next fundamental concept encountered in the development of the *InVitro* framework is the realisation that alternative sub-model forms are necessary for the optimal representation of the various parts of complex dynamic systems such as ecosystems. Unlike other researchers (e.g. Van Dyke Parunak et al. 1998) who have divided their agents based on where they sit on the "analytical" verses "individual" dichotomy, the authors of this report have purposefully put the focus in *NWS-InVitro* on the role an agent plays in a simulation and have treated all agents with similar roles in largely the same way. This facilitates the development and implementation of many alternative sub-models for the same process, each with its own strengths and weaknesses, which in turn allows for selection of the best representation for the case in hand. By identifying the common features (roles) for the system components a minimum set of high level agent types (classes) were identified (Thing, Environment and Monitor agents) and all other agent types were subordinate forms of these types.

Within all three of the core agent types used in *NWS-InVitro*, the sub-models may sit at any point along the spectrum from individual-based to classical mathematical model formulations. In some cases multiple sub-models sitting at different points on this spectrum were implemented and the final form used was chosen based on computational efficiency and (more importantly) how well it captured the system dynamics at the scale of interest.

1.3.1 Things

The Thing agent type is one of the most widely used agent types in *NWS-InVitro*. The hierarchy of sub-models of the Thing type is given in figure 1.3.1.

Sub-models that are associated with discrete locations are typically derived from the Thing agent type. Thing agents (like Environments described below) use the asynchronous handling of time that is standard in *NWS-InVitro* (discussed in detail in section 1.4 below). Thing agents can interact with all other agent types and are used to represent highly mobile entities such as animals, boats, and drift buoys. Details regarding the agent types derived from Thing can be found in the chapters dealing with Animal (chapter 2), Population (chapter 3), Blastula (chapter 6), Larva (chapter 7), Vessel (chapter 12), Boat (chapter 14) and Port and Fixture (chapter 13) agents.

1.3.2 Environments

The Environment agent type deals with those system components that have a distinct areal coverage or a global influence. For instance, bathymetry, currents, biogenic habitats (such as sponge beds), fisheries management zones and road networks are all represented using Environment agents of one form or another.

The two main forms of Environment agent are time series and data layers (which may be read in from files or calculated dynamically). A hierarchy of Environment agent types is given in figure 1.3.2.

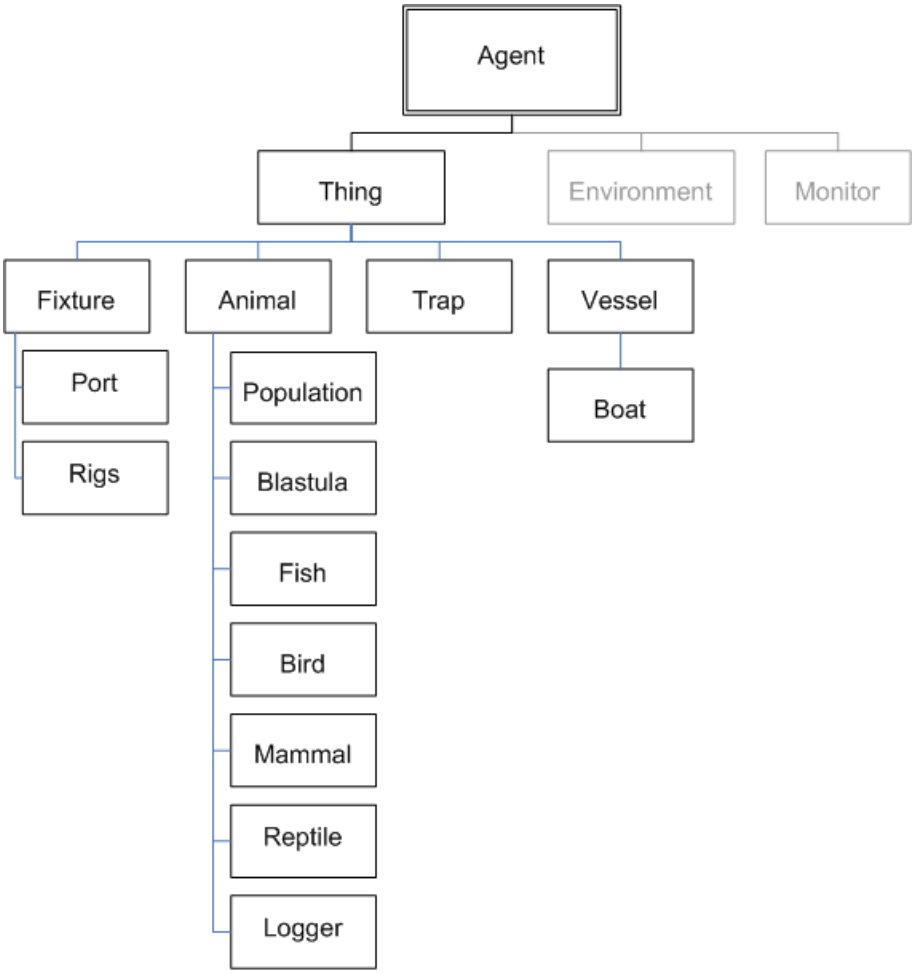

Figure 1.3.1: Hierarchy of Thing agents.

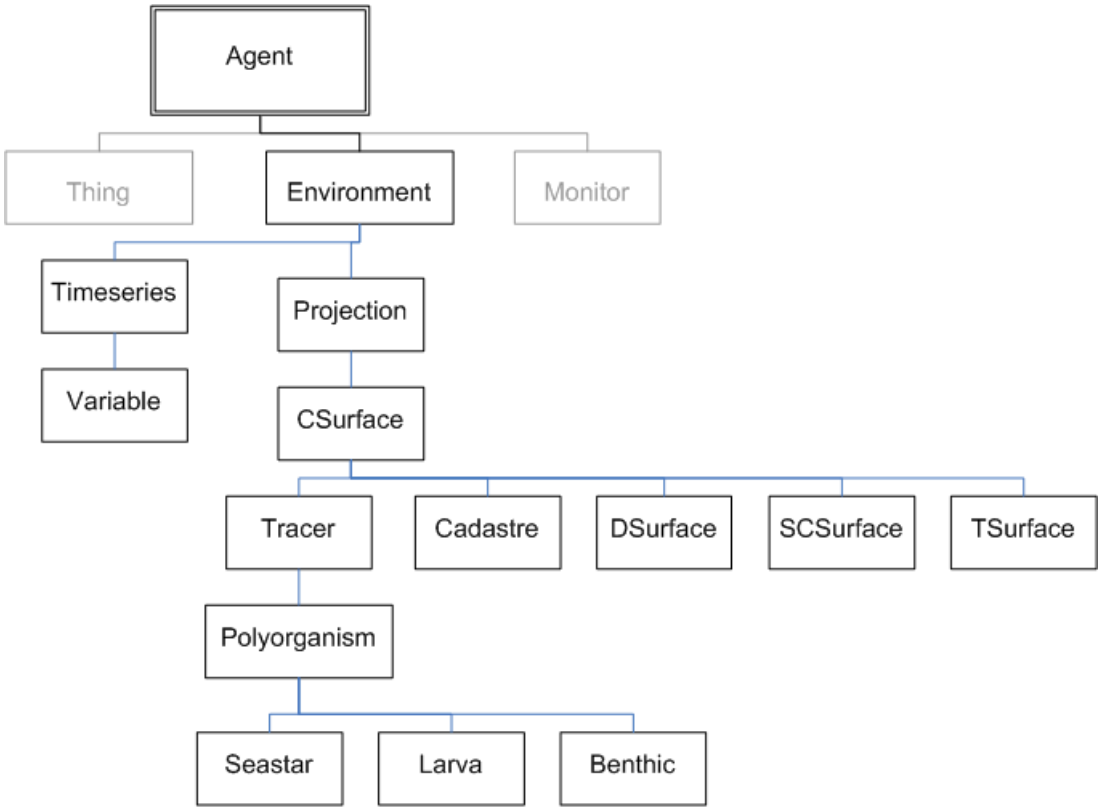

Figure 1.3.2: Hierarchy of Environment agents.

Time series agents

Time series (or Variable) agents can exist as stand-alone agents or as state variables within other classes, specifically in Contaminants and Fixture agents. Time series agents can be considered as a thin layer between time-indexed arrays of *n*-tuples (ordered sets of *n* elements) and the agents that interrogate them. Aspects of the behaviour of many sub-models in *NWS-InVitro* (e.g. discharge from an outfall) are driven by time series data. The behaviour of these sub-models may depend on the status of a times eries agent in two ways – either (i) the current value from the series or (ii) the sequence of times represented in the time series.

Time series agents are usually queried for values appropriate for some nominated time, t, which would usually be the current time-step. The values returned for t (in the half open interval (T_i, T_{i+1}), where time series entries occur at T_i and T_{i+1}) are those associated with T_i.

CSurface agents

CSurface agents are the basic form of agents with spatial extents (e.g. geo-referenced data layers such as bathymetry). The main role of these high level type of environment agents is to seamlessly convert grid-based data into values that smoothly vary across the model domain so that other agents can act accordingly. The values in CSurface agents can either remain static during an entire run or they can vary through time (e.g. for winds and currents) – if they are used to store time-indexed grids of vectors. CSurface agents can change according to the state of another agent, though this requires a comparison (and potentially a synchronisation) of the subjective time of the two agents.

CSurface agents can also be derived from the states of other agents. This is a useful way of representing environmental attributes, such as realised ocean currents that depend on winds, tidal state and underlying currents. In this way broader sets of conditions can be generated that remain consistent with the rest of the prevailing environmental conditions.

Tracer and Cadastre agents

Tracer and Cadastre agents are derived from the CSurface agent type. These polygon and point (vertex) based agents need not be represented as regular grids. They are used to represent anthropogenic features, such as contaminant plumes and pipelines.

Tracer agents can be more elaborate than Cadastre agents. Agent types derived from Tracer agents are used to represent patches of biological entities (like biogenic habitats). Tracer agents may be advected and diffused, and may also exhibit self motility. They are most usefully used to represent dynamic patches or distributions that evolve actively through time and interact extensively with other model components. While pure CSurface agents can change through time they are typically data layers rather than dynamic formulations, whereas Tracer agents are used for the latter. In contrast, Cadastres are usually used to represent things that do not change appreciably or do so in a scripted way, such as the boundaries of fishing zones or areas closed to vessel traffic.

Cadastre and Tracer agents are vertex based, Environment agents, which means they may be advected and diffused, and may exhibit self motility. This is a fundamental property of every class derived from the Tracer class, which includes our representations for pipelines, benthic organisms, larvae and bycatch. Tracers may be used to represent both contaminant plumes and, via derived classes, most of the Environment-represented biological entities.

The inclusion of the animate models in this class hierarchy may seem odd, but the reason is that in the open water, clouds of larvae (for example) behave in much the same way as plumes of contaminants. Since the physics of advection and diffusion operate on the base class, Tracer, its child classes are automatically subject to the same processes. This is particularly important when trying to simulate the organisms suspended in the water column associated with an outflow from some stationary source.

The basic diffusive behaviour of the Tracer agents will be outlined here. The diffusion of Tracer agents is dealt with by considering radial and areal diffusion separately. Radial diffusion is associated with dispersive plumes, the edges of which move outward

at a more or less constant rate so that the area of the plume grows geometrically. Areal diffusion is associated with plumes whose area increases linearly with time. The following formulation is used to find the realised value of diffusion scalar $\mathbf{R}_t$

$$\mathbf{R}_t = \rho_{rad} \cdot \phi_{diffR} + (1 - \rho_{rad}) \cdot \sqrt{\frac{\phi_{diffA}}{\pi} + \frac{A_{t-dt}}{\pi}} - \sqrt{\frac{A_{t-dt}}{\pi}} \quad (1.1)$$

where ρ_{rad} is the radial proportion; A_t is the area of the polygon at time t; ϕ_{diffA} is the diffusion areal constant; and ϕ_{diffR} is the diffusion radial constant. The final distance advected is determined by moving the vertices of the Tracer agent away from its centroid by $\mathbf{R}_t \cdot dt$ units (where dt is the time-step). *NWS-InVitro* uses this simple quantised advection scheme rather than a more sophisticated formulation to aid with to computational efficiency. The errors introduced using the quantised method are not of a magnitude to warrant the use of a more accurate scheme (such as integrating through the current field) which would be prohibitively expensive computationally.

Ocean currents: DSurfaces, CSurfaces and SCSurfaces

Handling ocean currents proved to be one of the significant obstacles in the NWSJEMS project, due to the vast amount of data the system had to step through. The handling of currents went through three distinct phases each proving valuable throughout the development cycle.

In NWSJEMS the interface between the other agent types and the current CSurfaces was consistent for all sub-model types. This means that alternative sub-model variants could be arbitrarily substituted for each other without necessitating changes to the formulation of the other agents in the system. This feature of the *InVitro* framework is common for nearly all agent types, enabling us to compare the dynamics and performance of different representations of a system.

DSurfaces

Initially a proxy for real modelled currents was created using a simple model that generated an oscillating (tidal) current, which was affected by bathymetry in a simplistic manner. It was a reasonably successful proxy that required no significant overhead and allowed development to focus on other models while the time series of ocean currents was being generated and processed. This model is present as the DSurface class, and provides a template for any environmental layer that is directly derived from another layer. In principle this could be used as a starting point for a model which derives a proxy for other broad scale fields. For example it could be used to provide a proxy for chlorophyll from temperature, nutrients and other forcing data, or used to generate some habitat index based on the state of other environmental agents.

CSurfaces

Another form of current modelling was covered in the CSurface agents. These were used to introduce sets of hydrodynamically sensible currents that were taken from the output of another model. These pre-generated currents were snapshots of the predicted current patterns given by the large scale circulation model, MECO (Condie et al. 2006). When using CSurfaces to supply the currents, the time-series of current images was put

in a directory as a set of files whose names corresponded to the model time at which they were to be swapped in to the model. The CSurface agent which handled currents used the time-stamped swapping mechanism, which already existed in *InVitro*, to automatically replace the current field at the right time. While an effective means of introducing realistic currents (much speedier than directly incorporating a finescale hydrodynamics model) there was a large overhead associated with reading in the large data sets (given the regional scale of the model domain). As the development of *NWS-InVitro* progressed, and our simulation became more complex, this overhead became untenable and another way of dealing with the currents was sought. In other circumstances, with smaller data sets, the use if CSurface agents is an extremely straightforward way to introduce data which is unaffected by anthropogenic or biological interaction; unfortunately, for larger data sets, such as the currents in the study region over a number of years, this method can be as impracticable as incorporating the fine scale hydrodynamics model itself.

SCSurfaces

The potential for prohibitive overheads when using CSurface agents in regional scale model implementations lead to the development of the SCSurface agent type – which allows for increased model flexibility and capability across scales. In the SCSurface agent type, a grid is laid over the model domain and a set of third degree polynomial models is specified for the currents in each grid cell. These models took as their variables the tide, a lagged tide, and the northerly and easterly components of the wind field. For *NWS-InVitro* the coefficients for the polynomials were estimated by statistically fitting the polynomials to the output of a MECO model applied on the same grid. The SCSurface model used in *InVitro* was able to reproduce the MECO output to an accuracy of 80% beyond the immediate coastal areas.

As currently implemented, SCSurfaces are general enough to be used as the basis for a variety of models comprising a set of "spatially explicit local models" (they are not constrained to the three dimensional polynomials used to represent the MECO currents within *NWS-InVitro*). If the SCSurface current model is desired then it requires two sets of data: a time series with entries which correspond to the time, tide, lagged tide, and the u and v components of the wind; a set of geographically located coefficients for the polynomial model of the current in the four variables from the previous data set. For efficiency, these geographic references are already in model space, but it is entirely feasible (though slower) to supply them in some other ordinate system. SCSurfaces do not require either dense or regular data: in principle they could be fed quite an irregular coverage of local models. This would correspond well to data that are evaluated on a triangulated network, for example. The performance of the model in terms of "lookup time" tends to increase by "O(log(n))", which is a fairly modest cost for spatial queries. It would be quite straightforward to interpolate between the centroids of local models to yield a continuous field.

On model execution, data from the time series are loaded into a simple table and are stepped through based on the subjective time of the agent providing the currents. The coefficient data are loaded into an indexed table, and the georeferencing data are loaded into a sorted array which is indexed by the x or y ordinate (respectively). The cell values of this array correspond to either an "unresolved" flag, or to the index of the local model that is closest to the cell. Cells are not populated with indices until other

model agents make requests for the current at locations within the cells. By delaying the resolution of these indices until they are needed, the computational overhead is reduced significantly (as it omits the resolution for a large number of cells which are never queried). When an agent requests a current at some location, the appropriate table entry of coefficients is found using a binary search on each of the two geographic ordinates. The polynomials corresponding to the current at the depth required (in our case either the surface or a depth averaged current) are then evaluated using the tidal and wind data from the time series.

1.3.3 Monitors

The Monitor agent type is used to represent all sub-models which must potentially interact synchronously with all (or at least a large percentage) of the other agent types in *NWS-InVitro*. Monitors typically include observational models, management models, and severe storm or cyclone events. These agent types are guaranteed to be synchronous at the point of interaction, which is essential for polling (or adjusting) the state of the appropriate subsystems. A hierarchy of Monitor agent types is given in figure 1.3.3. A few of the high level Monitor agents will be discussed below, but the bulk will be described in later chapters towards the end of the report.

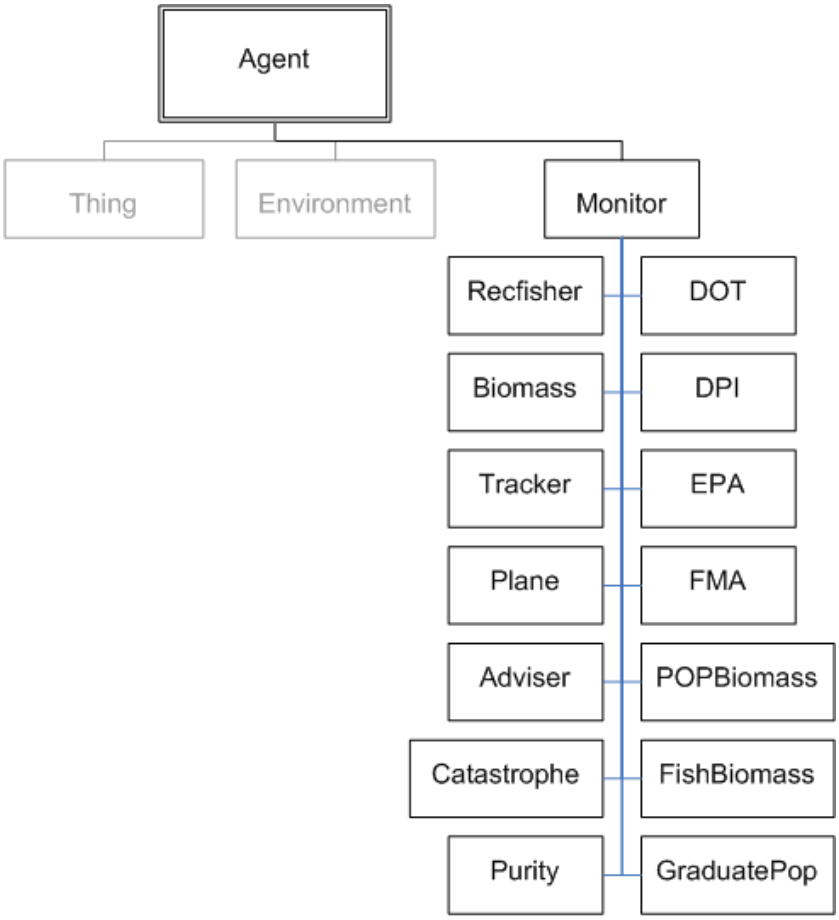

Figure 1.3.3: Hierarchy of Monitor agents.

1.3.4 Biomass and Tracker agents

Biomass and Tracker agents are forms of Monitor agents that keep records on the state of the model (i.e. generate output for later visualisation). The Biomass agents keep a track of overall biomass and abundance in an area (along with associated summary statistics) for any specified species (taxon or agent type); while Tracker agents record the location of individual agents of the specified species (taxon or agent type).

1.4 Agent sequencing

As mentioned in an earlier section, each instance of a sub-model in *NWS-InVitro* has a preferred time-step due to a combination of computation efficiency and modelling assumptions to do with the resolution and type of processes being represented. Mismatches between the time-steps and between the phases of agents can give rise to temporal anomalies and result in poor model performance. Thus, if such mismatches can be avoided then there is a distinct advantage to using asynchronous processing since the agents will use their optimal time-step as much as possible. The most common means of avoiding temporal mismatches while maximising computational efficiency are the judicious use of variable speed splitting and adaptive time-steps across independent processes (Lyne et al. 1994; Ebenhöh et al. 1997). By using these strategies, simulations can be structured to proceed through a series of synchronised "timeslabs" within which agents run purely asynchronously at whatever scale best matches their numeric needs.

Despite its widespread use, this approach to adaptive time-steps was not used in its purest or most common form in *NWS-InVitro*. The complexity of *NWS-InVitro* demanded a higher degree of computational efficiency and so *NWS-InVitro* was written so that it can run purely asynchronously, but can dynamically shift into adaptive synchrony when agents intend to interact. There is an important corollary to this: the models must smoothly scale themselves to time-steps that are shorter than their natural optimum.

The use of asynchronous and adaptive time-steps in *NWS-InVitro* and the spatially explicit and heterogenous nature of the model mean the handling of time and the interaction of agents must be dealt with carefully. The Scheduler is the means by which the two things (the progression of time and sequencing of interactions) are handled and it sequences the sub-models much like a multi-tasking operating system apportions time between competing programs (Lyne et al. 1994; Barabanov, 1997; Waszniowski & Hanzalek, 2004). The *NWS-InVitro* scheduler is built on the maintenance and manipulation of entries in a triad of priority queues: it uses the first queue (the Run Queue) to determine which agent to run next; the second (the "times to run" queue which is local to each agent) to determine how long the agent should run; and the third (the Standing Queue) to determine which other agents must be contemporaneous with the next agent to run. By separating the queues, agents can have adaptive time-steps that allow them to respond to changes in their environment (other agents) as needed.

Time-steps can be asynchronous, but time is still monotonic, so one of the chief jobs of the Scheduler is to ensure that an agent in the Standing Queue that has a subjective time earlier than given by $t+dt$ gets to act first and that the time-step size dt is adjusted accordingly within that agent. A significant consequence of this treatment of time is that all sub-models (regardless of form – whether mathematical or individual-based) must be cast so they are able to run for an arbitrary amount of time from the start of any given time-step. This constraint is absolutely essential, as interactions between sub-models are

usually required to be synchronous. This is the only constraint on the sub-model formulations though, as the use of the Scheduler to handle time and sequence of execution means that the sub-models can be developed without the need to explicitly deal with the interleaving of agent types within the sub-model definitions.

1.4.1 The run queue and standing queue

With each pass through the main loop in *NWS-InVitro*, the Scheduler can process agents from the Run Queue and the Standing Queue – though at any given time, an agent is represented only once across the entire set of agents drawn from the Run and Standing Queues together. All agents in the Standing Queue are required to be contemporaneous with the agent at the top of the Run Queue. These agents must not be dependent on the state of any other agent and are processed in the order of insertion into the queue. In contrast, the agents in the Run Queue can be dependent upon the state of other agents and are sorted by the time at which they wish to run and a priority; agents within any given time-priority block are (optionally) randomised to ensure no systematic preference based on queue order. This separation of the future (desired) execution schedule and the current schedule is critical in maintaining flexibility in dealing with adaptive time-steps in a lightweight fashion. Attempting to incorporate the "time to run" queue into the Runqueue effectively forces a search of the Runqueue and an adjustment of its entries every time a time-step is shortened when agents need to become synchronous. Keeping the queues separate means that an agent can determine its time-step merely by looking at the top of its own "time to run" queue, and adjustments are achieved merely by inserting a request in that queue.

In addition to these Scheduler controlled queues, each agent also maintains its own queue of times at which it would like to begin a time-step. As each agent in the Run Queue is dealt with the appropriate time-step (*dt*) is determined by the agent's preferred step length, the future times it wishes to run and whether it must adjust its *dt* to ensure that it is contemporaneous with any other agents so that interactions may occur.

Scheduler execution of the queues

At the beginning of each pass through the main loop of *NWS-InVitro*, every entry in the Standing Queue is checked against the subjective time of the head of the Run Queue. Any agent in the Standing Queue that is lagging behind is run to bring it into synchrony. Once the Standing Queue and the head of the Run Queue are synchronous then the head agent is run for one time-step. It is left up to each agent to reintroduce itself into the priority queue once it has been processed.

This method of processing the agents means that time-steps used for the various types of agent are optimal for the formulation used in the agents. This is of benefit both for the effectiveness of the model but also for computation efficiency – time-steps can be short when agents are interacting and synchrony is required and they can be extended when agents are acting independently. Potentially interacting agents signal each other and adjust their time-steps accordingly to bring them into synchrony so they can resolve the interaction. Note that agents can not gain or lose time overall. If an agent's subjective time is behind the system's, it acts to catch up (with constraints to do with potential interactions with other agents so that they do not relive elapsed time); and if an agent is ahead of the system's, it either removes itself from the queue without

executing a time-step and re-introduces itself further down the Run Queue, or it only runs for the balance of its time-step.

While the majority of the temporal handling in the model is dealt with in seconds elapsed, special attention has been paid to ensure that we are also able to schedule events which occur periodically based on calendar dates. This is accomplished by parsing a string which encodes the required periodicity (such as "@07/01") and systematically inserting entries in the "times to run" queue of the sub-model in question.

1.5 Dealing with space

Each sub-model in *NWS-InVitro* has a spatial scale that it either tries to maximise computational efficiency or best matches the assumptions used in their derivation. These spatial scales can be quite different across sub-models, which may have important implications for model behaviour. For example, model anomalies or degradation may occur when gridded sub-models do not share a common grid and unresolved mismatches between the grids occur. As indicated earlier, *NWS-InVitro* avoids potential spatial anomalies by maintaining a common model space in a three dimensional "continuous" floating point system and wrapping each sub-model within a conversion layer that translates between the common model space and the spatial scales explicit to that sub-model. Because an agent's responses to queries are all required to be embedded in a "continuous" floating point space, they are inherently less dependent on knowledge of the natural spatial scale of other agents. This also facilitates the interaction of different sub-models (agents), which is the core of the agent-based approach.

To interact, agents need to know about the status of their environment (including the identity and state of surrounding agents). Searching through and checking against lists of agents can be computationally expensive. To mitigate this issue two mechanisms are employed in *NWS-InVitro* to try and reduce the amount of searching each agent must do: a "neighbourhood" grid and Adviser agents (a type of Monitor agent that maintains and periodically refreshes a spatial array of summary statistics describing the state of the system). The nature of the information an agent requires determines which of the two methods is used: when an agent wishes aggregated data (e.g. indication of the state of the local environment) it calls upon the Adviser agent (see chapter 8) covering the area of interest; conversely, when an agent requires information on an individual agent in its local vicinity it uses a "neighbourhood" grid. Neighbourhoods are essentially grids containing pointers to agents found in those grids. As an agent traverses an area the grid is automatically updated. Requests for information on an agent's neighbours can then be limited to searches within the "neighbourhood" grid rather than the entire list of agents.

While the Neighbourhoods and Advisers are gridded, the movement they track within the model, such as advection, diffusion and autonomous motion, occurs in the three dimensional, floating point model space. This approach avoids the aliasing which occurs with gridded movement schemes, and minimises the spatial effects of any quantisation arising from the preferred time-steps of the agents.

1.5.1 Thing agents and navigation

One aspect of interrogating the local environment or "neighbourhoods" is that searches and navigation must be considered along with the other spatial issues. Thing agents and those agent types derived from Things, exhibit forms of search behaviour – mainly to do with movement – either in searching for desirable habitat, evading threats or hunting specific prey (which includes anthropogenic activities such as fishing). This issue has been explored extensively in the fields of foraging theory (Stephens and Kreb, 1986) and computer game design (Yap, 2002). Workers in both fields have found neural networks and genetic algorithms to be effective means of representing the behaviour. The algorithm used by Thing agents in *NWS-InVitro* is some what like the A* algorithm, a form of path search heuristic. This algorithm is essentially a directed search for a low cost path. It was chosen for use in *NWS-InVitro* in preference to other approaches since it fits well with our understanding of the decision process used by ecological and human agents and we can refine it by incorporating heuristics. At present this algorithm is only used for navigation, but it could be used to solve other problems, provided they can be mapped onto some sort of "cost" surface.

2. ANIMAL AGENTS

The Animal agent type is one of the predominant agent types used in *NWS-InVitro*. It (or its subordinates) is used to represent all mobile animals (e.g. prawns, finfish, sharks and turtles). This high level structure draws heavily from the traditional Individual Based Model structure (DeAngelis & Gross, 1992). The bulk of the formulation of this agent type was inherited from earlier versions of *InVitro* known as the PMEZ-Model (Lyne et al. 1994).

The Animal agent type deals with the movement, mortality and basic behaviour of animal life. Derived agents deal with variants or elaborations of this structure (e.g. the fish subclass elaborates the spawning behaviour). The derived agent types also deal with specific life history characteristics, such as air breathing, flight, and spawning of larval stages. The following description will deal with the general Animal agent formulation and any elaborations for specific subclasses will be described in the parts of the document dealing with the behaviour of those agents.

2.1 Basic behaviour tree for Animal agents

The clearest way of describing the implementation of this class of agent is to present its overall behaviour tree and then to elaborate on the specific formulations used in each step. The basic behaviour tree (figure 2.1.1) includes all the major activities a mobile agent is capable of performing at some point in its life cycle. Each Animal agent consults the tree on each time-step and acts accordingly.

2.1.1 Floating and decomposition

While live Animal agents can move independently (section 2.1.2), dead animals can only float and be advected by the currents using:

$$L(x_{t+dt}, y_{t+dt}, z_{t+dt}) = L(x_t, y_t, z_t) + (\omega_w \cdot \mathbf{W}_t + \omega_c \cdot \mathbf{C}_t) \cdot dt \tag{2.1}$$

where dt is the time-step; $L(x_t,y_t,z_t)$ is the location at time t, x is longitude, y is latitude, and z is elevation relative to sea-level (which we usually refer to as depth, in this document); $\mathbf{W}_t$ is the wind vector at time t; $\mathbf{C}_t$ is the current vector at time t; ω_w is the weighting coefficient for wind and ω_c is the weighting coefficient for currents.

During the current time-step the biomass of the dead agents is assumed to decompose following exponential decay such that:

$$w_{t+dt} = \begin{cases} w_t \cdot \left(1 - \dfrac{0.0001 \cdot dt}{600}\right) & , \text{ if } w_t > 0.01 \\ 0.0 & , \text{otherwise} \end{cases} \tag{2.2}$$

Here we take it to decay at an arbitrary 0.01% of its mass every ten minutes, and we note that the standard measure of mass in *NWS-InVitro* is a kilogram.

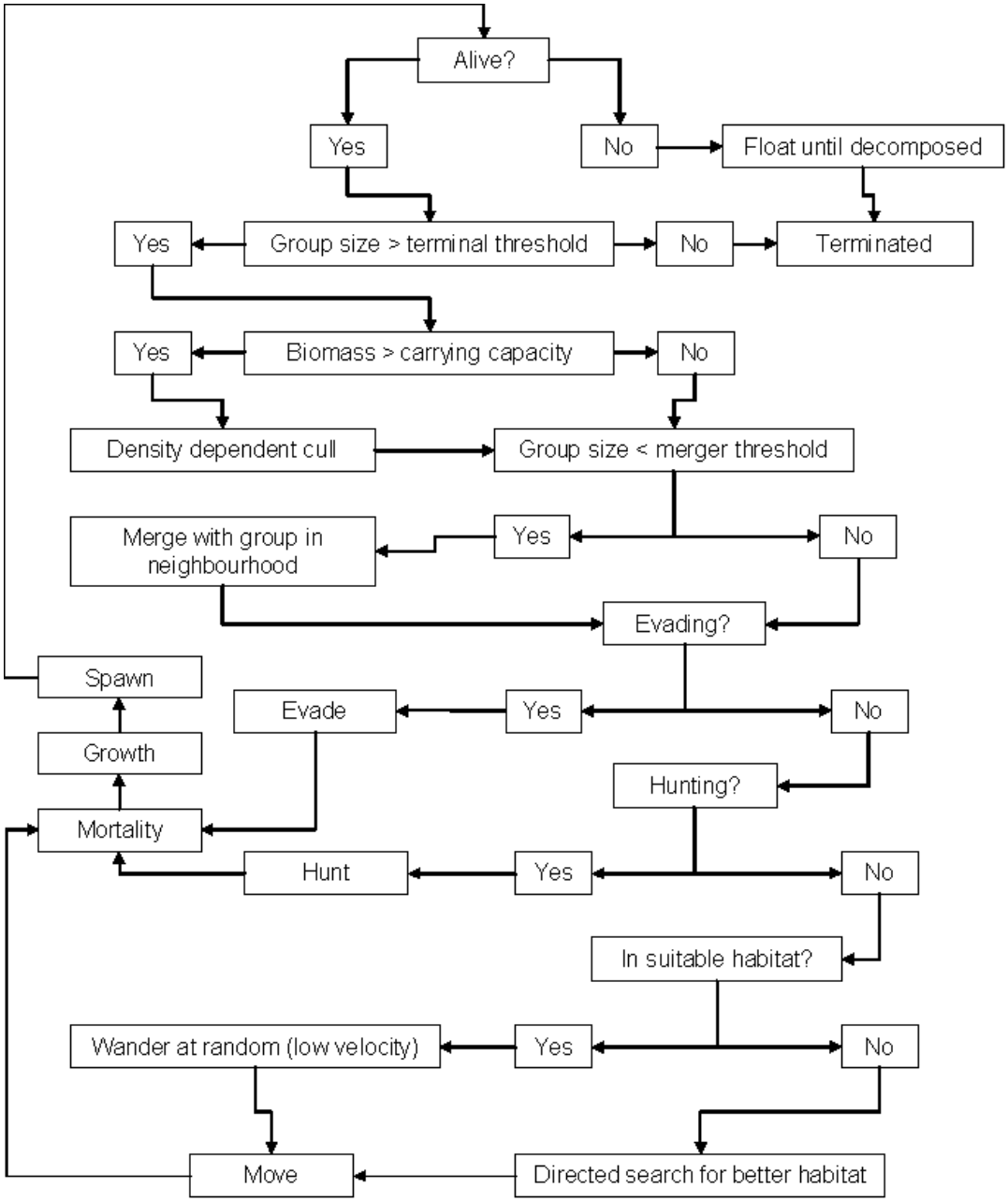

Figure 2.1.1: Behaviour tree for Animal agents, this tree is traversed on each time-step and determines the actions taken by the agent at that time.

2.1.2 Movement

The equation used for movement is very similar to equation (2.1) for floating, such that:

$$L(x_{t+dt}, y_{t+dt}, z_{t+dt}) = L(x_t, y_t, z_t) + (\mathbf{v}_t \cdot s_t + \omega_w \cdot \mathbf{W}_t + \omega_c \cdot \mathbf{C}_t) \cdot dt \quad (2.3)$$

where $\mathbf{v}_t$ is the direction vector of the agent (in three dimensional space); s_t is the speed of the agent; and the other terms are as of (2.1).

For computational efficiency, the implementation of targeted movement in *NWS-InVitro* is handled by providing the agents a list of waypoints that mark the route between the agent's departure and destination points. Agents move sequentially from waypoint to waypoint. If they end up within 10 cm of their current target waypoint they are assumed to have reached it and adopt the appropriate behaviour – either continuing on to the next waypoint using (2.3), finding a new destination point via a gradient search or wandering in the local area (see following sections for details on both of these approaches). Recall that the *NWS-InVitro* model's representation of locations in physical space are as continuous as our floating point representation allows, namely sixteen significant digits in each ordinate, so a radius of 10 cm around waypoint gives us a generous buffer to avoid errors arising from inadequate precision.

Habitat gradient search

The steps given below are applied whenever an Animal agent has to find a suitable habitat in which to relocate. The one constraint on this is that the animal can not move out of the defined model domain (a reflective boundary is assumed in that case). The form of time management used in *NWS-InVitro* (see the section in chapter 1 on the Scheduler) also means that the size of *dt* used when applying these equations differs depending on how far from previously identified suitable habitat the agent currently sits (i.e. if it is close than the *dt* used is small).

Site assessment

The first step of the habitat gradient search is to assess the habitat suitability of potential sites. The habitat assessments are based around calculating an overall habitat suitability rating (η_{tot}) using the following equations of agent habitat suitability (the convention adopted is that negative values indicate unsuitable habitat and high values are favourable habitats):

$$\eta_{tot} = \omega_\eta \left(\min\left(0, \frac{d - d_{\max}}{2}\right) + \min\left(0, \frac{d_{\min} - d}{2}\right) - \frac{\omega_d \cdot \left| d - d^A_{opt} \right|}{100} + \min\left(0, \frac{d^B_{opt} - d}{2}\right) + \sum_i \eta_i \right) \quad (2.4)$$

where ω_η is a comfort scalar (a global parameter (i.e. applying to all taxa) that is typically set to 100, in combination with hunger and terror scalars it weights an animal's reaction to its immediate needs and surroundings); d is current depth of the agent; d_{max} is the maximum depth that taxon can be found at; d_{min} is the minimum depth that taxon can be found at; d^A_{opt} is the best total sea depth for that taxon (to differentiate shelf from slope from open ocean taxa); d^B_{opt} is the optimal depth for that taxa; and η_i is

the habitat rating for specific (usually) biogenic habitat types (e.g. seagrass, mangroves, reefs), which is calculated using:

$$\eta_i = \frac{\eta_{pref,i} \cdot \left(B_i - \eta_{thresh,i}\right)}{\left|\eta_{thresh,i}\right| + 1} \tag{2.5}$$

where $\eta_{pref,i}$ is a taxon specific coefficient of habitat preference; $\eta_{thresh,i}$ is a taxon specific habitat quality threshold; and B_i is the local value (usually biomass of the taxon) of the habitat at the location of interest. This general equation form is also applied if the agent has particular temperature, salinity or conspecific density requirements (e.g. if aggregate to spawn and currently in spawning condition).

Site selection and vector determination

When moving to a new location (or patch of habitat). The first step is to perform habitat assessments, of the form given above, for a short list of potential sites. For ease of computation, this search is done in a spiral – so as to cover as much area as possible as quickly as possible – figure 2.1.2). When a suitable site has been found (one with a non-negative and non-zero suitability rating), or if no suitable sites are found then the best of the list tested is chosen, the agent moves toward the selected location.

The equation used to determine the locations to test is derived from the movement equations (as the agent has to move to reach them), so that the new test location ($L_{t'}$) is given by:

$$L\left(x_{t'}, y_{t'}, z_{t'}\right) = L\left(x_t, y_t, z_t\right) + \left(\mathbf{v}_{dir} \cdot s_{dir} + \omega_w \cdot \mathbf{W}_t + \omega_c \cdot \mathbf{C}_t\right) \cdot dt \tag{2.6}$$

where $\mathbf{v}_{dir}$ is a random vector with each coordinate ~U(0,1); and s_{dir} is the radius of the search from the current location to the one to test, which is calculated using:

$$s_{dir} = \left(1 + \left(\left[\frac{i}{4}\right]\right)^{\Theta}\right) \cdot \frac{-1 + \sqrt{1 + 4 \cdot \left(\omega_v\right)^2 \cdot s_t \cdot dt}}{2 \cdot \left(\omega_v\right)^2} \tag{2.7}$$

with i the index (from zero) of this test location in the list (assuming 4 test locations per "loop of the spiral"); Θ is the exponent indicating the "looseness" of the spiral; ω_v is a directional variability coefficient (the degree of turning of current heading allowed per "loop of the spiral", see figure 2.1.2). Note the square brackets [] indicates only the integer value of the internal fraction is used.

Once the site has been selected then the new direction vector ($\mathbf{v}_{t+dt}$) of the agent is given by:

$$\mathbf{v}_{t+dt} = \frac{\mathbf{v}_{t''}}{\sqrt{x_{t''}^2 + y_{t''}^2 + z_{t''}^2}} \tag{2.8}$$

where

$$\mathbf{v}_{t''} = \frac{\frac{2 \cdot L_I \cdot \mathbf{v}_{shift}}{\sqrt{x_{shift}^2 + y_{shift}^2 + z_{shift}^2}} + \mathbf{v}_t}{L_I + 1} \tag{2.9}$$

with L_1 the a weighting of the optimal direction associated with reaching this location (in this case set to two as a default for animal movement in order to decrease the computational cost of the migration relative to usual travel); and $\mathbf{v}_{shift}$ the vector subtraction of the current location L_t from the new location to move to $L_{t'}$.

In addition, the new speed (s_{t+dt}) of the agent is as follows:

$$s_{t+dt} = \min\left(\frac{\sqrt{(x_t - x_{t'})^2 + (y_t - y_{t'})^2 + (z_t - z_{t'})^2}}{dt}, s_{cruise}\right) \tag{2.10}$$

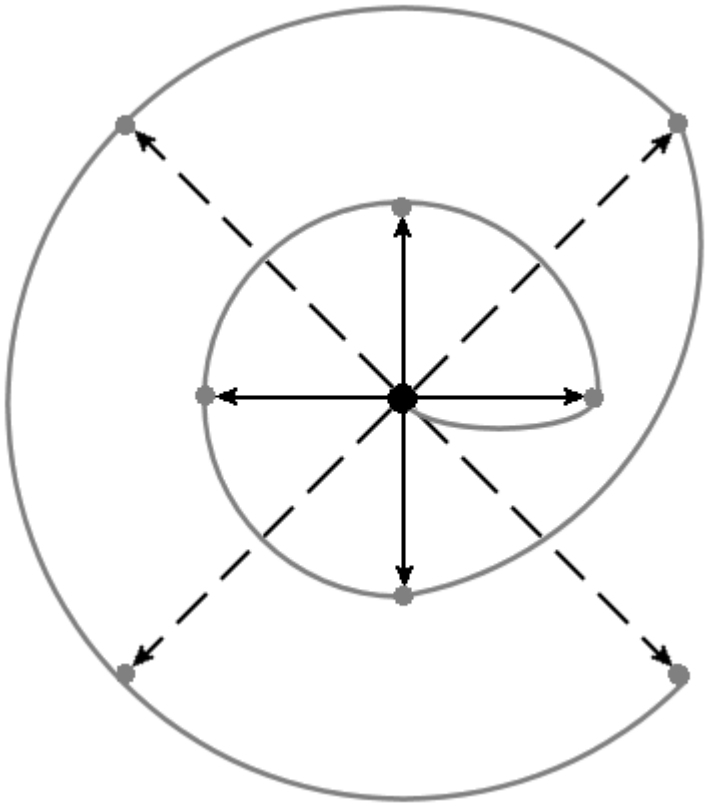

Figure 2.1.2: Example of a search spiral.

Local random wandering

If an animal is in suitable habitat it does not search for new patches, but wanders randomly in the local area. The direction of this movement ($\mathbf{v}_{t+1}$) is given by equation (2.8) where:

$$\begin{aligned} x_{t''} &= x_t \cdot \cos(\sigma_v \cdot \pi \cdot \omega_v) - y_t \cdot \sin(\sigma_v \cdot \pi \cdot \omega_v) \\ y_{t''} &= y_t \cdot \cos(\sigma_v \cdot \pi \cdot \omega_v) + x_t \cdot \sin(\sigma_v \cdot \pi \cdot \omega_v) \\ z_{t''} &= z_t \end{aligned} \tag{2.11}$$

and $\sigma_v \sim N(0,1)$. The new site this movement would take the agent to is determined using equation (2.3), but replacing $\mathbf{v}_t$ with $\mathbf{v}_{t+1}$. If this new site is not within suitable

habitat then the process is tried again (up to a maximum of 20 attempts). Also before actually moving the agent their realised speed (s_{t+dt}) is varied slightly using:

$$s_{t+dt} = \min\left(s_{\max}, \max\left(0.001, s_t \cdot \left(\sigma_s \cdot \omega_s + 1\right)\right)\right) \tag{2.12}$$

$\sigma_s \sim N(0,1)$. This shift in speed attempts to represent small scale events that would impact the agent's speed along the path (e.g. slowing to observe for predators or speeding across open space in between microhabitat features) and also prevents agents becoming locked into fixed steps around their habitat. Furthermore it also allows for a more diverse range of habitat exploration behaviours and movement trajectories.

Merging schools

When an agent represents a school or group rather than individuals, for computation efficiency (and in some cases for biological plausibility) it is necessary to merge significantly depleted agents (this would represent something like schools merging). Searching for agents with which to merge is done using equations (2.6) to (2.10), but with the agent looking for conspecifics rather than suitable habitat and the first bracketed term in (2.7) replaced by a merge radius. If the agent can move within that merge radius of a conspecific agent then it merges with that other agent. Of the two original (unmerged) agents, the smaller one is taken as the subsumed agent and is nested within the larger conspecific agent. The movement and other aggregate behaviours of the merged agent is dictated by the controlling (originally larger) agent, but all other characteristics of the (nested and controlling) parts of the new merged agent are kept from the original agents they merged agent was formed from. This avoids the need to aggregate agent attributes that are a combination of the attributes of the two original agents.

The ability to merge agents in this way can play a tremendous role in keeping the computational overhead to a minimum. As mortality occurs within the agents, the number of entities represented drops, but the computational load associated with the agent remains constant. If we consider this in the context of a population in an environment with a finite carrying capacity, we find that the number of agents will tend to increase significantly as the simulation progresses. A significant contribution to the overhead is made up by the various searching behaviours associated with the agents. By merging conspecifics with low memberships, the numbers of agents that are overtly active are reduced. The single controlling agent in a set of merged agents is responsible for searching and assessing the environment. This agent controls all of the basic behaviour of the merged group, and so reduces the load associated with the behaviour of the animals to that of the controlling agent. Ultimately merging achieves the best compromise between computational efficiency and the agent-specific "experience history" that makes ABMs so attractive. That is, given that the agent is representing an aggregate entity, it makes little sense to persist in tracking them separately if they have few individuals remaining and a combined entity does not lose any of the specific details that are the key reason for using an agent-based approach in the first place.

2.1.3 Hunting and evading

Behavioural algorithms for hunting and evasion are included in *InVitro*, both as behaviours available to animals, but in a modified form they also become the basis for

the fishing boats. For increased flexibility in model deployment, additional options were included in *InVitro* so that the explicit use of this behaviour is not necessary if assumptions relating habitat quality to food availability and overall trophic web health are brought into play. This optional representation implicitly assumes that the hunting and evading is going on rather than stepping through it explicitly. To ensure continuity in the representations, the hunting and evading behaviour was explored during the module developmental and calibration stages to see if the final more abstract forms still captured the desired end-point properties of the finer scale behaviours. The option using the implicit representation of hunting was the one used in the NWS-MSE analysis (Fulton et al. 2006b). Nevertheless, given that the hunt-evade behaviour was used to condition the broad-scale habitat-based option and that it underlies the boat behaviour, a summary of the hunting and evasion algorithms are presented below.

Hunting (searching)

The first step in hunting is to locate prey. To do this, the animal moves as discussed in section 2.1.2, but evaluates anything it perceives in its local neighbourhood (that is any agents within its perception range, which is a user defined parameter). If the animal is already hunting a prey item identified earlier it ignores any new potential prey items. If however, the animal is actively searching for prey and is yet to locate any then it assesses each potential prey item using the following criteria and weighting. Note that a prey item need not represent an individual. Rather it is whatever is appropriate for that predator type, so for a tuna it may be a single pilchard form a larger school (with stalking at the level of the school and mortality assigned at an individual level since the attack is successful); while for a filter feeding organism it would be a patch of plankton rather than a single plankter.

The size criteria ($\delta_{desire,size}$) determines if the prey is of a size which can be consumed:

$$\delta_{desire,size} = \begin{cases} 1 & , \; b_{p,t} < b_{h,t} \cdot \kappa_{h,gape} \text{ AND } w_{p,t} < w_{h,nom,t} \cdot \kappa_{h,gut} - g_{h,t} \\ 0 & , \text{ otherwise} \end{cases} \tag{2.13}$$

where $b_{p,t}$ is the width of individual prey at time t (similarly for predator width $b_{h,t}$); $\kappa_{h,gape}$ is the proportional gape size of the predator; $w_{p,t}$ is the mass of individual prey; $w_{h,nom,t}$ is the current nominal mass of healthy individuals of that species of predator at its current age; $\kappa_{h,gut}$ is the predators gut capacity parameter (proportional to body size); and $g_{h,t}$ is the current gut content of the predator. Only prey where $\delta_{desire,size}$ is equal to one are pursued, this means that any prey that are too large to enter the gut or fit into the mouth are rejected out of hand. Some species consume their prey in bites, rather than whole – in these cases $\kappa_{h,gape}$ would be larger than one, and similarly for $\kappa_{h,gut.}$

The weighting factor weighs the prey based on diet preferences and hunger level of the predator. If presented with multiple prey agents in the same search the predator will hunt the most heavily weighted (most desirable) prey. The prey weighting ($\delta_{desire,diet}$) is given by:

$$\delta_{desire,diet} = \frac{w_{p,t} \cdot \iota_{p,h} \cdot \left(1 - \dfrac{g_{h,t}}{w_{h,nom,t} \cdot k_{h,gut}}\right) \cdot \omega_{diet} \cdot \omega_{hunger}}{\max\left(1, d(h,p)\right)} \tag{2.14}$$

with $d(h,p)$ the distance between the prey p and predator h; $\iota_{p,h}$ the preference of predator h for prey p; and the hunger scalar (ω_{hunger}) is a global parameter (applying to all taxa) that is typically set to 1 000. The main purpose of ω_{hunger} is in establishing the relative importance between different modes of behaviour such as feeding, fleeing or reproduction, and while it is currently a global constant it can be made specific to the various taxa. The diet scalar (ω_{diet}) is calculated as:

$$\omega_{diet} = \begin{cases} \left(1 + w_{h,nom,t} - w_{h,t}\right) \cdot 100 & , \; w_{h,t} < w_{h,nom,t} \\ 1 & , \text{ otherwise} \end{cases} \tag{2.15}$$

Attack

When an animal identifies a prey item (by searching its local environment) it registers interest in that agent, which forces the time-step of the two agents to synchronise. The predator then tries to cover the distance to the prey (who may try to evade) and get close enough to actually catch and consume the prey. To do this the following inequality is checked:

$$d(h,p) > \kappa_{attack} \cdot s_{\max} \tag{2.16}$$

where K_{attack} is the predator's allowed temporal separation from the prey (equivalent to the time the predator will commit to a final lunge in an attack); and s_{max} is the predator's maximum possible speed. If the inequality is true then the predator uses the movement algorithm 2.9 to keeping chasing the prey. If the equality is false (and a final attack is possible) the prey is attacked. It is typically assumed that in any one attack only one prey individual in the prey agent is attacked per individual predator represented by the predator agent (e.g. if an agent representing a school of 1 000 pilchards was attacked by an agent representing 10 sharks then only 10 pilchards could be eaten in that one attack). Organisms which consume more than one animal at a time (such as filter feeders) can be readily accommodated as the need arises, however by making the prey items consumed per attack a taxon specific parameter, or by comparing the volume of the prey agent with the volume of the predator's mouth (as is the case for trawling). The following equation is used to update the membership ($N_{p,t}$) of the prey agent at time t:

$$N_{p,t} = \begin{cases} N_{p,t} & , \; b_{p,t} > b_{h,t} \cdot \kappa_{h,gape} \text{ or } g'_{h,t} > w_{h,nom,t} \cdot \kappa_{h,gut} \\ \max\left(0, N_{p,t-dt} - \alpha N_{h,t}\right) & , \text{otherwise} \end{cases} \tag{2.17}$$

where $N_{h,t}$ is the membership of the predator agent at time t; α is the quantity each predator consumes and $g'_{h,t}$ is the new gut content of individual predators if they were to consume the prey, which is given by:

$$g'_{h,t} = \begin{cases} g_{h,t} + w_{p,t} & , \; N_{h,t} > N_{p,t-1} \\ g_{h,t} + \dfrac{N_{p,t-dt} \cdot w_{p,t}}{N_{h,t}} & , \text{otherwise} \end{cases} \tag{2.18}$$

with $w_{p,t}$ the weight of individual prey. Note that the initial state of the gut is entered as part of the agent's configuration, so at run start the agents have a user specified gut fullness and any newborn agents created during the model run are assumed to be created hungry.

Once a prey item has been consumed the hunt is considered over and the time-step used by the predator reverts to its "normal" length (as defined in its parameter file). At this point the prey is also identified as eaten (which means it can not interact with anything external to the predator agent and has effectively been removed from the model domain).

Evasion

Animal agents can attempt to evade predators and unattractive (e.g. contaminated) habitats within their taxon specific perception range. In the same way as the local neighbourhood is assessed for potential prey, the neighbourhood is also assessed for potential threats. The threat assessment is performed using the following criteria and weighting (which are similar in principle to (2.14) and (2.15) used in finding in prey).

The size criteria ($\delta_{fear,size}$) determines if the threat is of a size that is threatening:

$$\delta_{fear,size} = \begin{cases} 1 & , \; w_{e,t} < w_{f,t} \cdot 0.77 \;\; \text{AND} \; w_{f,t} \cdot N_{f,t} > w_{e,t} \cdot 1.3 \\ 0 & , \; \text{otherwise} \end{cases} \tag{2.19}$$

where $w_{e,t}$ is the mass of the animal assessing the threat; $w_{f,t}$ is the mass and $N_{f,t}$ the membership of the agent who is potentially a threat. Only potential threats with $\delta_{fear,size}$ equal to one are evaded, this means that any individual smaller than the agent making the threat assessment that doesn't also hunt in packs are not considered fearsome. A more sophisticated assessment which dealt with smaller predators which "bite" would be a straightforward extension.

The weighting factor weighs the potential threats based on fear ratings and terror level of the Agent performing the assessment. The threat weighting ($\delta_{fear,scale}$) is given by:

$$\delta_{fear,scale} = \frac{\iota_{e,f} \cdot \omega_{immed} \cdot \omega_{terror}}{\omega_{feardist}} \tag{2.20}$$

with $\iota_{e,f}$ the threat rating of threat f by taxon e; the terror scalar (ω_{terror}) is a global parameter (applying to all taxa) that is typically set to 2 000 and is used to balance the imperative to flee against other activities; $\omega_{feardist}$ is the fear distance factor and ω_{immed} is a threat immediacy factor. The fear distance factor is given by:

$$\omega_{feardist} = \begin{cases} d_{range} + \left(d_{safe} - d_{range}\right)^2 & , \iota_{e,f} > 1 \;\; \text{AND} \; d_{range} > 0 \\ 1 & , \text{otherwise} \end{cases} \tag{2.21}$$

where d_{safe} is the taxon specific "safe distance" and d_{range} is the range to the threat and is given by:

$$d_{range} = \sqrt{\left(x_{e,t} - x_{f,t}\right)^2 + \left(y_{e,t} - y_{f,t}\right)^2 + \left(z_{e,t} - z_{f,t}\right)^2} \tag{2.22}$$

with ($x_{e,t}$, $y_{e,t}$, $z_{e,t}$) the assessing agent's coordinates and ($x_{f,t}$, $y_{f,t}$, $z_{f,t}$) the threat's coordinates. The immediacy factor is given by:

$$\omega_{immed} = \begin{cases} w_{f,t} \cdot N_{f,t} & , \quad d_{range} > d_{safe} \\ 0 & , \text{ otherwise} \end{cases} \tag{2.23}$$

The most fearsome threat recognised is evaded. Evasion consists of movement at maximum speed using the methods detailed in section 2.1.2. If an animal successfully escapes a predator by exceeding the "safe distance" then the animal's time-step is reset to synchronise with the next expected general time-step of the model (this acts to expand the animal's time-step back up to larger spans without immediately jumping to "unstressed" levels).

2.1.4 Mortality

Natural mortality

A number of different mortality options were used depending on the exact form of the Animal agent implemented for any one species. These mortality options need not be mutually exclusive and one or all of these options may operate at any one time on an Animal agent. Each of the mortality options is outlined below.

Density-dependent mortality

If density dependent mortality is implemented as a control on population size (used when specific feeding equations are omitted and sufficient food supply assumed to hold if suitable habitat is available) then the following mortality term (M_D the number of individuals lost to this form of mortality) is applied at the beginning of each time-step:

$$M_D = \begin{cases} \left(\left(\sum_i B_{i,t}\right) - K_{tot}\right) \cdot \left(1 - \exp\left(-\left(\frac{\log(0.125)}{\kappa_t}\right) \cdot dt\right)\right) & , \text{ if } \sum_i B_{i,t} > K_{tot} \\ 0 & , \text{otherwise} \end{cases} \tag{2.24}$$

where K_{tot} is the species (taxon) specific total carrying capacity for the entire area of the model domain; $B_{i,t}$ is the biomass of agent i (of the specific taxon or species under consideration) at time t; κ_t is the period over which density dependent mortality occurs (a cull period, so to speak); and dt is the time-step. Animals are removed from the population through this mechanism only when the biomass exceeds carrying capacity. Density dependent mortality is usually applied as a way to control the number of agents of a species. In the formulation we use "log(0.125)" rather than the more usual half-life to introduce a systematically more aggressive mortality (four times more aggressive) across the cull period. Within an agent other types of mortality are applied to the members that comprise the agent.

Individual-based random mortality

For taxa where each agent represents less than 100 individuals the most appropriate form of mortality is likely to be randomly applied mortality. This form of mortality (M_I) is given by:

$$M_I = \max(j \cdot N_t, 1) \tag{2.25}$$

where N_t is the number of individuals represented by the agent:

$$j = \begin{cases} 0.001 \cdot j_R & \text{, if } a_t < a_{\max} \text{ for the taxon and } j_{\mathrm{M}} < \kappa_M \\ 1 & \text{, otherwise} \end{cases} \tag{2.26}$$

a_t is the current age (in years) of the agent; a_{max} is the longevity (in years) of the taxon; and $j_R \sim \mathrm{U}(0,1)$; $j_M \sim \mathrm{U}(0,1)$ and κ_M is the mortality rate calculated as (note that the minimum and maximum functions are used to constrain the possible values returned by the equation to realistic ranges):

$$\kappa_M = \min\left(1, \max\left(0, \left(1 - \sqrt{\left(1-\kappa_M^o\right)^2 \cdot \left(1 - \frac{\left(a_t - a_{\mathrm{mod}}\right)^2}{\left(\frac{a_{\max}+t_0}{2}\right)^2}\right)}\right) \cdot \frac{dt}{s_{yr}}\right)\right) \tag{2.27}$$

with κ^o_M is the base mortality rate; t_0 is the assumed age of recruitment (in years); s_{yr} is the number of seconds in a year and:

$$a_{\mathrm{mod}} = \frac{a_{\max}+t_0}{2} - t_0 \tag{2.28}$$

Age independent mortality

This form of mortality is similar in execution to the random mortality discussed above, but does not include age dependence. Instead it uses simple uniform mortality irrespective of age (a form of mortality often used in Individual-Based Models). This form of mortality is given by equations (2.25) and (2.26) except that κ^o_M is used in place of κ_M in (2.26).

Large-scale mortality (exponential decay)

For those taxa with agents containing large numbers (e.g. those representing entire sub-populations or large schools) exponential decay is a more effective way of representing mortality (random mortality tending to lead to anomalous mortality patterns). This form of mortality (M_L) is given by:

$$M_L = \begin{cases} N_t - N_t \cdot \exp\left(\frac{-\kappa_M^o}{s_{yr}} \cdot dt\right) & \text{, if } a_t < a_{\max} \text{ for the taxon} \\ N_t & \text{, otherwise} \end{cases} \tag{2.29}$$

Starvation mortality

If an animal's condition deteriorates to the point where it is less than two thirds of the nominal mass then it dies from starvation.

Human-induced and catastrophic mortality

The formulations used for human-induced mortality (e.g. fishing mortality) is described in the agent types responsible for the human threat to other agents (fishing is discussed in chapters 3 and 14, and poisoning in chapter 10). Similarly, catastrophic mortality for animals is described in the chapter on catastrophes (chapter 9).

2.1.5 Growth

Two forms of growth are implemented in *InVitro*. The first assumes that the agents actively feed (i.e. use the hunt and evasion behaviours in section 2.1.3). The other (discussed here under the heading "false metabolism") is used in the *NWS-InVitro* option where animals are assumed to find sufficient food so long as suitable habitat can be found.

Metabolism and growth

With explicit feeding enabled the growth of the individuals represented by an Animal agent is based on gut contents and metabolic demands such that the new mass (w_t) after growth is given by:

$$w_t = \begin{cases} w_{t-dt} + \left| w_{req} \cdot \mathbf{W_m} \right| & , \mathrm{w_{req}} > 0 \\ w_{t-dt} + w_{req} & , \text{otherwise} \end{cases} \tag{2.30}$$

where the growth ogive $\mathbf{W_m}$ = [0.70, 0.70, 0.70, 0.65, 0.60, 0.55, 0.50, 0.40, 0.30, 0.20, 0.15, 0.10, 0.07, 0.05, 0.03, 0.02, 0.01, 0.0]; and the mass left after metabolic requirements have been dealt with (w_{req}) is given by:

$$w_{req} = \begin{cases} w_{t-dt} \cdot \omega_{met} \cdot dt - g_t & , w_{t-1} \cdot \kappa_{met,basic} \cdot dt > g_t \\ w_{t-dt} \cdot \kappa_{peakmet} \cdot dt & , \text{otherwise} \end{cases} \tag{2.31}$$

with $\kappa_{peakmet}$ the peak metabolic rate; g_t the current gut content; and the maintenance rate (w_{met}) is given by:

$$\omega_{met} = \kappa_{peakmet} \cdot \left(1 - \mathbf{W_m}\right) \tag{2.32}$$

and the (zero based) index of the cell used from the ogive (i) is:

$$i = \left[17 \cdot \min\left(1, \left(\frac{w_{t-dt}}{w_{nom}} \right) \right) \right] \tag{2.33}$$

Note that the square brackets [] means only use the integer part of the internal value.

After growth the new gut contents (g'_t) is given by:

$$g'_t = \max\left(0, g_t - w_{req}\right) \tag{2.34}$$

and the new nominal mass ($w_{nom,t}$) is recalculated as:

$$w_{nom,t} = \begin{cases} w_{nom,t-dt} + w_{req} \cdot \mathbf{W}_m & , \dfrac{w_t}{10000 \cdot l_t^3} > 1.1 \\ w_{nom,t-dt} & , \text{otherwise} \end{cases} \tag{2.35}$$

noting that the condition on the use of the first equality in (2.35) indicating it will be used when realised mass is greater than 110% of nominal mass; and with l_t the current length of the animal given by:

$$l_t = w_t^{\left(\log(\lambda_l) / \log(\lambda_w)\right)} \tag{2.36}$$

where λ_l is the taxon specific length coefficient and λ_w is the taxon specific weight coefficient of the length-weight. conversion

False metabolism

This representation attempts to represent average growth in the presence of sufficient food to cover maintenance costs and typical levels of growth. This representation is required when an organism's supporting trophic web is not explicitly represented in the configuration. Under this formulation the new mass (w_t) is given by:

$$w_t = w_{\max} - \exp\left(-\frac{\lambda_{mass}}{365.25 \cdot 86400} \cdot dt \right) \cdot \left(\frac{2 \cdot w_{\max} - w_{t-dt} - w_{nom,t-dt}}{2} \right) \tag{2.37}$$

where w_{max} is the weight of individuals that have reached the von Bertalanffy maximum length (l_∞); and λ_{mass} is the taxon specific (annual) growth correction coefficient.

When the formulation (2.37) is used for growth, the new value for nominal mass ($w_{nom,t}$) is given by:

$$w_{nom,t} = w_{\max} - \exp\left(-\frac{\lambda_{mass}}{365.25 \cdot 86400} \cdot dt \right) \cdot \left(w_{\max} - w_{nom,t-dt} \right) \tag{2.38}$$

2.1.6 Reproduction (spawning)

The reproduction by Animal agents is governed by seasonality. If it is the correct time of the year and the agent is mature then it will spawn. Fish agents, use a slightly more elaborated set of reproduction steps. These steps will be detailed here along with the animal reproduction routines.

Reproduction by animal (fish) agents is dealt with using a multi-step process. First the time of the year and (if a fish agent) environmental conditions are checked (see sections below) and then the state (age and condition) of the agent is checked, before the final calculation of realised fecundity (see description later in this section).

The maturity and breeding condition of an Animal agent is checked using the following algorithms. The value of $\delta_{spawn,cond}$ (the flag indicating if an Animal agent is in spawning condition) is set using:

$$\delta_{spawn,cond} = \min\left(1, \max\left(0, w - 0.9 \cdot w_{nom,t}\right)\right) \tag{2.39}$$

with w the mass of the animal; and $w_{nom,t}$ is the nominal mass of healthy individuals of that species of animal at its current age. The test against 90% of the nominal weight is to prohibit organisms in poor condition from successfully reproducing, particularly when the false metabolism is in use.

The value of $\delta_{spawn,age}$ (the flag indicating whether the animal is sexually mature) is determined using:

$$\delta_{spawn,age} = \min\left(1, \max\left(0, a_t - a_{mat}\right)\right) \tag{2.40}$$

where a_t is the Animal agent's current age (in seconds); and a_{mat} is the age of maturity of that taxon in seconds (converted from days or weeks in the parameter file on model initialisation).

If conditions and the time of year are found to be suitable for spawning (see below), all mature Animal agents are allowed to spawn, providing that:

1. the maximum number of allowed agents has not been exceeded; and
2. that the Animal agent has not already spawned recently (there is a fallow period after each spawning, though multiple spawnings per season are possible depending on the parameterisation used).

The check to see if an animal has cleared its fallow period takes the following form:

$$\delta_{spawn,active} = \min\left(1, \max\left(0, \gamma_{spawn} - \left(t - t_{last\_spawn}\right)\right)\right) \tag{2.41}$$

where γ_{spawn} is the length of the breeding fallow period in seconds (converted from days or weeks in the parameter file on model initialisation); t is the current time (in seconds) since the start of the run; and t_{last_spawn} is the time (in seconds) that the Animal agent last spawned.

Spawning is only undertaken when the final spawning flag (δ_{spawn}) is greater than zero, with:

$$\delta_{spawn} = \delta_{spawn,cond} \cdot \delta_{spawn,age} \cdot \delta_{spawn,active} \tag{2.42}$$

Timing

The cyclical nature of breeding through a year is captured using a sine curve. First the proportion of the Julian calendar year that has expired is calculated as:

$$p_{doy} = \frac{t_{doy}}{t_{yr}} \tag{2.43}$$

where t_{yr} is the total number of days in the current calendar year within the model run (accounting for leap years); and the days elapsed in the current calendar year (t_{doy}) given by:

$$t_{doy} = \sum_{prev\ months} t_{dom,i} + t_d \tag{2.44}$$

with $t_{dom,i}$ the number of days in month i (the extra day in February in leap years is accounted for at this point); and t_d is the number of days that have elapsed in the current month.

The probability that an animal at time t is within the spawning season and ready to spawn ($\delta_{time,t}$) is given by:

$$\delta_{time,t} = \max\left(0, \left((1-\Omega)\cdot\sin\left(2\cdot\pi\cdot\frac{\left(p_{doy}\cdot 86400\cdot 365.25 - t_{startcycle,d}\right)}{t_{peroid,d}}\right)+\Omega\right)\right) \quad (2.45)$$

where $t_{startcycle,d}$ is the day of the year the first breeding period of the year begins; $t_{peroid,d}$ is the time in days for one iteration of the spawning cycle (attaining readiness then going through a fallow period); and Ω is any offset to the curve (noting that Ω is a proportion of a year and that $\Omega \in [0,1]$). The calculation performed in (2.45) is then repeated for the time ($t+dt$) and the final step to see set the flag indicating whether the animal actually attempts to spawn at time t (δ_{time}), is carried out according to:

$$\delta_{time} = \begin{cases} 1 & , k < \delta_{time,t} \text{ OR } \delta_{time,t} > 0, \delta_{time,t+dt} = 0 \\ 0 & , \text{otherwise} \end{cases} \quad (2.46)$$

where k is a random value from U(0,1). The partial dependence of the flag on the probabilites at time t and time $t+dt$ is to ensure the animal attempts to spawn at some point in the breeding season.

Location

Fish agents must also be in suitable spawning habitat before reproducing. The indicator of spawning habitat suitability (δ_{site}) is calculated as:

$$\delta_{site} = \delta_{bed}\cdot\delta_{mate} \quad (2.47)$$

where the indicator of physical habitat suitability (δ_{bed}) is only non-zero if the bed is non-negligible in size (>1e-8m long) and it is within a certain distance of the animal (this spawning radius is entered as a taxon specific model parameter). If the value of this flag is zero then the animal moves to a more suitable habitat. The movement algorithms discussed in section 2.1.2 are used to accomplish this.

The value of the conspecific indicator (δ_{mate}) is set based on the presence of suitable mates in the area. This uses the local neighbourhood array (see the section in chapter 1 on handling space) to search within the local patch of spawning habitat for conspecifics in spawning condition. If insufficient mates are found (the size of spawning aggregations is another taxon specific model parameter) then spawning stops.

Realised fecundity

Once an Animal agent actually spawns (so the indicators in (2.22), (2.25) and (2.26) are non-zero) then the number of offspring produced is calculated using:

$$R_t = 0.5\cdot N_t\left(\frac{\max\left(0, \left(\prod_{external\ forcing}\frac{\varpi_i}{\varpi_{ref,i}}\cdot\left(\mu_f - \mu_r\cdot\left(w_{\max} - w_t\right)\right)\right)\right)}{w_{new}}\right) \quad (2.48)$$

where ϖ_i is the value of some external forcing factor i (say rainfall or river flow) that effects reproductive success; $\varpi_{ref,i}$ is the reference level for external factor i (the level at which fecundity is neither increased or decreased by that factor); μ_f is the base fecundity for that taxon; μ_r is an additional condition specific spawning rate; w_{max} is the weight of individuals that have reached the von Bertalanffy maximum length (l_∞); w_t is the current weight of the spawning animal; w_{new} is the weight of the newly spawned individuals (either newborns, hatchlings or larva); and N_t is the membership of the spawning agent. After spawning the mass of the spawning individuals is reset to 75% of their nominal mass.

In many cases the new animals are created as new instances of the Animal agent and begin their existence at their parent's location. The membership of each new group (i.e. the numbers in the new group) is set using:

$$N_t = \min\left(N_{std}, R_t\right) \tag{2.49}$$

with N_{std} the taxon specific default group size. If $R_t > N_{std}$ then multiple groups are created, such that no one of the new groups has more than N_{std} members. If other agents further down the agent queue also spawn in this local area in this time-step then their offspring will be used to top-up any already existing newly created agents rather than immediately create more new agents (in this way a good number of fully-stocked agents are created rather than ending up with increasing numbers of sparsely populated agents). Note that at the end of this time-step if any of the new agents are still sparsely populated (i.e. do not contain the minimum number of members for a group of that species) then they will try to merge with other groups of their own age, as discussed in 2.1.2.

This method of handling reproduction is not used for turtles or any species (such as fish) which have a distinct juvenile or larval phase that carries them away from the adult population. The alternative methods of handling this distinct juvenile phase are discussed in chapters 6 and 7.

2.2 Mammal, bird and reptile agents

Mammal, bird and reptile agents are subclasses of Animal agents that are elaborated to a small extent in order to capture specific details of those life histories. Specifically:

- *Mammals* must surface to breathe if stepping through small time-steps (e.g. during predation or evasion), on larger steps they are assumed to be surfacing to breath as required.
- *Birds* may be in the air or on land rather than just in the water. If they are in the water then there are limitations on the time spent underwater.
- *Reptiles*, as with mammals, these must come to the surface to breathe if using short time-steps (it is assumed they surface as required if using large time-steps). In addition, they may come ashore to breed.

3. POPULATION AGENTS

Population agents are a subordinate form of the thing agent type. Their formulation is a variant of the age-structured models used widely in fisheries (Haddon, 2001). Rather than represent the entire population with a single agent, a number of Population agents are used, each representing a sub-population with its own centre of gravity, movement behaviour and history. This has proved to be a good compromise between the flexibility of individual (or school) based models and the computational efficiency of differential equation models. The following equations detail the specific implementation used in *NWS-InVitro*.

The basic behaviour tree employed by the Population agents (figure 3.1) includes the major activities these kinds of agents undertake during their annual pattern of behaviour. The Population agents consult the tree on each time-step and act accordingly.

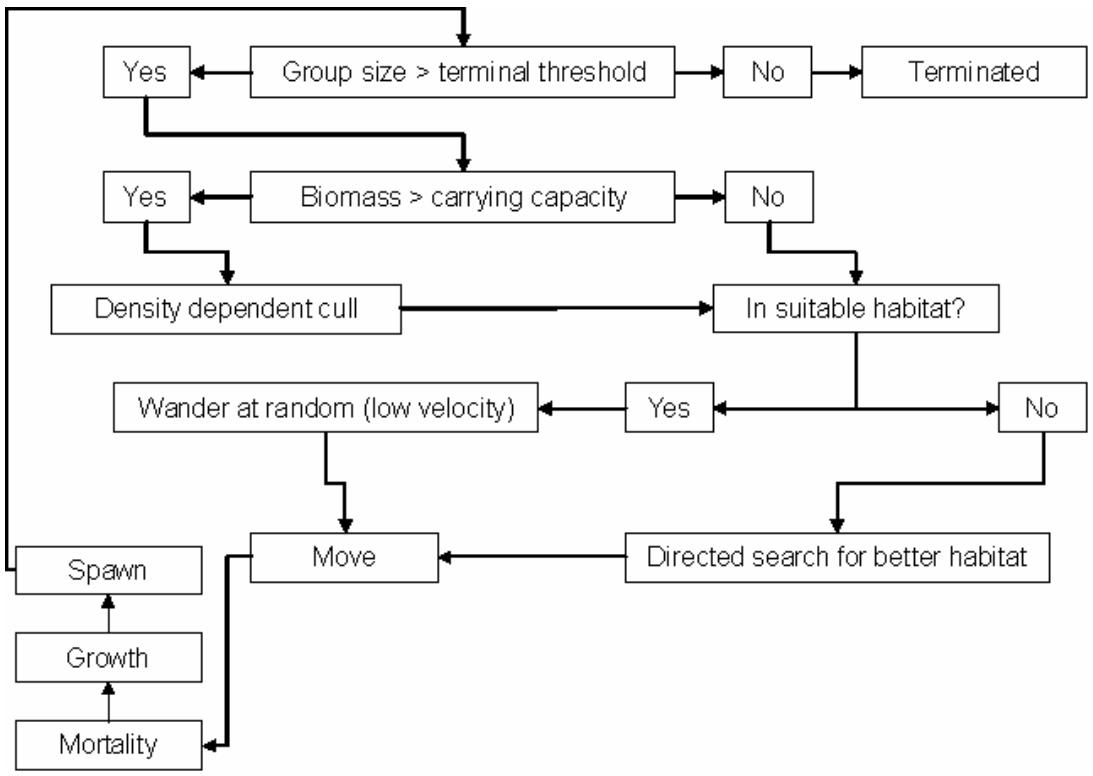

Figure 3.1: Behaviour tree for Population agents, which is traversed on each time-step, and determines the actions taken by the Population agent at that time.

3.1 Numbers in population agents

The number of individuals in age class a+1 of a "population agent" i at the start of the model ($N_{i,a+1,0}$) is given by:

$$N_{i,a+1,0} = \begin{cases} N_{i,0} & \text{for } a = 0 \\ N_{i,a,0} \cdot \exp(-M) & \text{for } 1 \le a \le A-1 \end{cases} \tag{3.1}$$

$$N_{i,0} = R_t \tag{3.2}$$

where R_t is the number of new recruits; M is natural mortality; and A is the maximum number of age-classes in the population. Once running, at any give point of time within the year, t such that $0<t<1$, the number of individuals in age class y+1 of a population i ($N_{i,y+1,t}$) is given by:

$$N_{a,y+1,t} = N_{a,y,t\_dt} \cdot \exp(-M \cdot dt) \tag{3.3}$$

During the historical fishing period at the end of each year the mortality caused by fishing is imposed on population i as:

$$N_{i,a,y,t} = N_{i,a,y,t-dt} \cdot \exp(-V_a \cdot F_{i,y,t}) \tag{3.4}$$

where $F_{i,t}$ is the fishing mortality imposed on age class y of population i at time t and:

$$V_a = \left(1 + \exp\left(-\ln(19) \cdot \left(\frac{a - \upsilon_{0.5}}{\upsilon_{0.95} - \upsilon_{0.5}}\right)\right)\right)^{-1} \tag{3.5}$$

with $\upsilon_{0.5}$ age of 50% selectivity and $\upsilon_{0.95}$ age of 95% selectivity.

Population agents are also subject to density dependent mortality. The formulation of this is of the same form as used for Animal agents (see equation (2.24) in section 2.1.4).

3.1.1 Initialisation of population age structure

The initial stock structure for each Population agent was determined using the Beverton and Holt stock-recruit parameters (as these specify the initial number of larvae and hence the remaining age structure of each fish population). The following equations based on equations (3.3) and (3.10) (see following sections) were run iteratively until a stable stock structure was achieved and then this structure was assigned to the Population agent as its initial conditions:

$$B_{larva,t} = N_{i,0,t} = \frac{S_{i,t} \cdot \alpha}{\beta + S_{i,t}} \tag{3.6}$$

$$N_{i,a,0} = N_{i,a-1,0} \exp(-M) \tag{3.7}$$

where S_i is the spawning biomass (equivalent to 0.5 of the total biomass) and M is annual natural mortality rate.

3.2 Size of population agent members

Individual growth for an animal in age class a is assumed to be governed by a von Bertalanffy growth equation, where length (l_a) is given by:

$$l_a = l_\infty(1 - \exp(-\kappa(a - t_0)) \tag{3.8}$$

where l is length; l_∞ is the von Bertalanffy maximum length; κ is the von Bertalanffy steepness parameter; and t_o is age of recruitment.

The associated individual mass is determined from a standard allometric equation:

$$w_a = \lambda_1 L_a^{\lambda_2} \tag{3.9}$$

with w is weight; and λ_1 and λ_2 the allometric length-weight relationship coefficients.

3.3 Reproduction in population agents

Population growth due to spawning and recruitment is determined using a modified Beverton-Holt recruitment function. Before a cohort is recruited to a population as $N_{i,0,y}$, the new recruits must pass through larval (and potentially juvenile) stages as a "larval agent"(Note that "larval agent" refers to this non-adult, non-recruited stages of the life history. Within the model it can be actually represent either "Larva" or "Blastula" agent types as described in Chapters 7 and 6). Larva agents are created when a Population agent successfully spawns. There are three stages in larval development: a free-floating stage, a settled stage, and a recruited stage, each with an associated growth and mortality. The biomass of a larval agent ($B_{larva,t}$) initialised at time t is determined from:

$$B_{larva,t} = \frac{S_{i,t} \cdot \alpha}{\beta + S_{i,t}} \tag{3.10}$$

$$S_{i,t} = 0.5 \cdot \sum_{a_{mat}} w_a \cdot N_{i,a,t} \tag{3.11}$$

where S_i is the spawning biomass (equivalent to 0.5 of the total biomass) and a_{mat} is the age of maturity.

To capture important ontogenetic shifts a species may go through before being recruited to the adult population, biomass passed to the larval agents passes through three stages before being given back to the adult population. The stages of larval (and juvenile) development are:

Stage 1 – larval: free floating cloud of larvae advect and disperse with the currents, grow using von Bertalanffy growth, and die given a specific larval mortality rate. In the default form of this model used in the North West Shelf implementation (see chapter 6, Blastula Agents) this stage is skipped, advection is assumed to occur and the distribution of larvae into juvenile (settled) habitat is based on distance (using the same form of exponential representation given below for the distribution of recruits to adult populations).

Stage 2 – settling: if the larvae is the correct age, over the correct substrate type (correct habitat defining groups, in the correct depth zone and is not already at

carrying capacity) then it settles. Growth and mortality is as for the larval stage.

Stage 3 – established: grows (based on the quality of the habitat it is in, this habitat can be degraded by anthropogenic activities such as trawling) until maturity when it recruits to an adult population. Natural and contaminant induced mortality is handled in the same way as for the other stages.

Recruits for a population come from all larvae of the same species that are of recruitment age. The contribution can be made dependent on the distance between a larval agent and population agent. In that case the recruitment to a population i ($R_{i,t}$) comes from each larval agent j of the same species as:

$$R_{i,t} = \sum_j R_{j,r} \cdot \kappa_{l,i,j} \cdot \exp(-\beta_j \cdot d_{i,j}) \tag{3.12}$$

where $d_{i,j}$ is the distance between population i and larva/blastula j; β_j is the spatial recruitment coefficient; $R_{j,r}$ is the number of larvae of recruitment age in larval agent j; and $\kappa_{l,i,j}$ is the scaling factor for larva j among the populations i such that $\sum_i \kappa_{l,i,j} = 1$

3.4 Movement of population agents

Population agents use the same movement algorithms as Animal agents (section 2.1.2). This includes habitat gradient searches and local random wandering. Population agents however do not merge as they are depleted because their reproductive potential allows the numbers within a Population agent to recover from low levels (in contrast to the more individual-based structure of the Animal agents).

3.5 Fishing mortality

During the historical period for which we have catch and effort data, fishing mortality is calculated by the Population Biomass (PopBiomass) agent (see chapter 11 for all relevant equations). The imposition of fishing mortality in the projection period of a simulation is handled by the Boat agents (chapter 14).

3.6 Ageing

The handing of population aging through the age classes is a sensitive issue in a model with asynchronous time-steps, such as *NWS-InVitro*. To ensure the timing of this was regular, but did not impact deleteriously on the actions of other agents, or the population itself (through anomalous interactions with the timing of spawning or the application of fishing mortality) a dedicated Monitor agent (GraduatePop) was created to handle the task. This agent type makes use of the Monitor characteristics to handle the timing and synchronisation issues and its only elaboration is its role in annually activating and moving the members of each age class in each Population on, by one age class.

4. POLYORGAMISM AGENTS

The Polyorganism agent type is used in *NWS-InVitro* to represent flora and fauna best modelled as patches (2D or 3D) rather than individuals or schools (e.g. patches of larvae or plankton, mangrove forests, seagrass meadows, stands of macroalgae and reefs). The decision structure behind this agent type is similar to that for the mobile agents (chapter 2), but incorporates more features typical of standard metapopulation models.

These agents are called polyorganisms as the most effective means of representing patches/forests/meadows/reefs is via polygons. These polygons may be discrete with a single polygon representing a single instance of the agent (with its own unique history and characteristics and with the ability to change its vertices through time as it grows, disperses or dies), or the polygons maybe a linked set, where a number of polygons (e.g. a regular grid) are used to represent the entire coverage of the agent in the modelled area. Individual polygons within this set will have their own attribute values and represent local dynamics of a wider population.

4.1 Basic behaviour tree

As for the Animal agents, the clearest way of describing this class of agent is to present their overall behaviour tree and then to elaborate on the specific formulations. The basic behaviour tree, (figure 4.1.1) includes all the major activities a "patchy" agent may need to be able to perform. Each Polyorganism agent consults the tree on each time-step and acts accordingly.

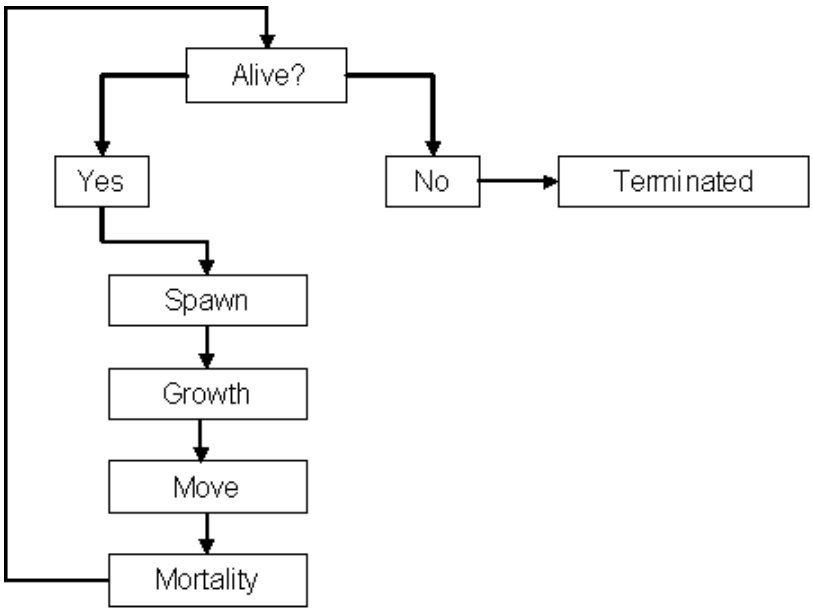

Figure 4.1.1: Behaviour tree for Polyorganism agents. This tree is traversed on each time-step and determines the actions taken by the agent at that time.

4.2 Movement

The movement of polyorganisms is via simple advection-diffusion algorithms. These algorithms either: advect the centroid of the polygon and then adjust the vertices of the polygon accordingly (i.e. translate the polygon); or they advect each vertex individually. If the first option is used then the basic advection-diffusion algorithm used matches that given in (2.1). If the second option is used then the following algorithm is used:

$$L(x_{t+dt}, y_{t+dt}, z_{t+dt}) = L(x_t, y_t, z_t) + \left(\delta_{diff} \cdot \mathbf{R}_t + \delta_{advect} \cdot \left(\omega_w \cdot \mathbf{W}_t + \omega_c \cdot \mathbf{C}_t\right)\right) \cdot dt \tag{4.1}$$

where $L(x_t,y_t,z_t)$ is the location at time t; $\mathbf{W}_t$ is the wind vector at time t; $\mathbf{C}_t$ is the current vector at time t; ω_w is the weighting coefficient for wind and ω_c is the weighting coefficient for currents; δ_{advect} is the flag indicating whether advection is being used; δ_{diff} is the flag indicating whether diffusion is used; and $\mathbf{R}_t$ is a diffusion scalar given by:

$$\mathbf{R}_t = \frac{\left(1-\rho_{rad}\right) \cdot \left(\sqrt{\frac{\phi_{diffA}}{\pi} + \frac{A_{t-dt}}{\pi}} - \sqrt{\frac{A_{t-dt}}{\pi}}\right) + \rho_{rad} \cdot \phi_{diffR}}{\sqrt{\left(x_t - x_c\right)^2 + \left(y_t - y_c\right)^2 + \left(z_t - z_c\right)^2}} \tag{4.2}$$

with $L(x_c,y_c,z_c)$ is the location of the centroid at time t; ρ_{rad} is the radial proportion; A_t is the area of the polygon at time t; ϕ_{diffA} is the diffusion areal constant; and ϕ_{diffR} is the diffusion radial constant.

The concentration (C_t) in this newly advected polyorganism is such that:

$$C_t = \frac{A_{t-dt} \cdot C_{t-dt}}{A_t} \tag{4.3}$$

4.3 Reproduction and growth

The increase of a polyorganism occurs via expansion of the existing polygon(s) or by the creation of new polygon(s). The expansion of the existing polygons (growth) is given by relocating the vertices using:

$$\begin{aligned} x_t &= x_c + \left(x_c - x_t\right) \cdot \rho_{growth} \\ y_t &= y_c + \left(y_c - y_t\right) \cdot \rho_{growth} \end{aligned} \tag{4.4}$$

where x_t is the new vertex coordinate; x_c is the centroid coordinate; and ρ_{growth} is the spatial growth proportion. In contrast, the creation of new polygons is either by duplicating the parental polygon and then letting it drift off, or by budding off a small dodecahedron (with a one metre radius) and letting advection and diffusion reshape it as it ages.

4.4 Mortality

Mortality for polyorganisms represents loss of the entire polygon, not reshaping (death of small parts) which was dealt with implicitly in the movement and "growth" routines. As a polyorganism can be made up of multiple organisms, the entire polyorganism is not terminated until the last polygon dies. The death of individual polygons is governed

by comparing a uniform random number (U(0,1)) with the probability of mortality (calculated using (4.5)) – if the random number is less than the probability then the polygon is killed. The probability of mortality is given by:

$$p_{mort} = \min\left(1, \max\left(0, m_t\right)\right) \tag{4.5}$$

with the effective mortality rate (m_t) as follows:

$$m_t = 1 - \sqrt{\left(1 - \kappa_m^o\right)^2 \cdot \left(1 - \left(\frac{\left(\left(a_t - \frac{a_{\max} + t_0}{2} + t_0\right)^2\right)}{\left(\frac{a_{\max} + t_0}{2}\right)^2}\right)\right)} \tag{4.6}$$

and a_t is the current age (in years) of the agent; a_{max} is the longevity (in years) of the taxon; κ^o_M is the base mortality rate; and t_0 is is the assumed age of recruitment (in years). This representation was used so that the age-specific mortality represents the non-linear shape typical for most species (high when at either extreme of life).

5. BENTHIC AGENTS

Benthic agents are subclasses of Polyorganism agents. The Benthic agents were designed to represent all of the habitat defining groups found on the North West Shelf of Australia. These habitat defining groups – seagrasses, macroalgae, mangroves and reef builders (corals and sponges, but primarily sponges in this instance) – are represented using a metapopulation model framework that tracks the evolution of percentage cover through time. This approach is adapted from previous habitat and metapopulation modelling work (Levins, 1969; Sainsbury, 1991; Tilman & Kareiva, 1997). Each of the habitat groups (agents) is represented by a series of habitat polygons that cover a specified area. In this case the sets of polygons used are regular grids. The reef habitat grids are 10 by 10 minute across the entire shelf and inshore areas; the seagrass and macroalgae grids are 12 by 12 minute and restricted to depths <50 m; and the mangrove grid is 3 by 3 minute and restricted to the coastline. While regional populations are considered for each benthic group in this North West Shelf implementation, the model formulation allows consideration of model areas of a wide range of sizes, from highly restricted (if small scale effects are under consideration) to broadscale (e.g. the entire matrix of reef habitat for the North West Shelf region). Regardless of the spatial scale chosen, within each polygon of a habitat agent the percentage cover, average height and biomass is tracked. These statistics are then used as indices for diversity (empirical observations indicate that there is a direct relationship between diversity and the average height of organisms in biogenic habitats such as sponge beds).

Two formulations are used to represent the different forms of habitat defining groups on the North West Shelf:

1. Benthos (small and large): Mangroves and reef builders are represented using an age-structured (benthos) model. The lifecycle of reef components is assumed to begin with recruitment into the youngest ageclass (smallest size class) of the small benthos and then with time the organism ages (grows) through the small benthos classes and may eventually transition into the large benthos class. The percentage cover of the small and large benthos classes is tracked separately. Note that patches of these two different size classes may overlap. Therefore, while the percent cover of small benthos is ≤100% (similarly for large benthos), the sum of the percent cover of small and large benthos ≤200%.
2. Macrophytes: The seagrass and macroalgae are represented by a model without age-structuring (i.e. they recruit straight to their final class and there is no progression between classes), but with light limitation. The general formulation for this variant uses many of the same equations as for the benthos and so will be discussed within the general benthos sections below, with only a few specific details relating to macrophytes given in the final section of this chapter.

5.1 Small benthos – cover

Small specimens of habitat include the “small” stages of species that grow to large size as well as some those species that always remain small (≤25 cm in height for reef builders and ≤100 cm for mangroves – heights taken from data classification scheme used in benthic sampling program (Fulton et al. 2006a)).

The percentage cover of small reef habitat (p_s) per polygon changes using the following age-structured equations of change, so that delays in habitat recovery can be adequately represented:

$$\frac{dp_s}{dt} = \sum_j \left(\begin{array}{l} \dfrac{A_{j,t-dt} \cdot p_{s,t-dt} \cdot \left(1 - p_{s,t-dt}\right) \cdot \mu_s \cdot U \cdot \Psi}{1 + e^{(-\lambda(j-\nu))}} + U \cdot \xi \cdot \Psi \cdot \left(1 - e^{(-p_T)} + \kappa_{rec}\right) \\ - A_{j,t-dt} \cdot p_{s,t-dt} \cdot \left(\dfrac{\kappa_s}{1 + e^{(\theta(j-\varphi))}} + \Phi_j A_{j,t-dt} \cdot D_{s,t} \right) - \dfrac{A_{j,t-dt} \cdot p_{s,t-dt} \cdot \omega}{1 + e^{(\phi \cdot (j-\varepsilon))}} \\ + A_{j-1,t-dt} \cdot p_{s,t-dt} \cdot \omega - A_{j,t-dt} \cdot p_{s,t-dt} \cdot \omega \end{array} \right) \quad (5.1)$$

where $A_{j,t}$ is the proportion of small habitat in age-size group j at time t. The terms of (5.1) correspond to growth, recruitment, mortality and aging and are defined in the text below.

5.1.1 Horizontal growth

Proportional coverage of Benthic agents can increase through horizontal growth or recruitment. The first term in (5.1) represents the horizontal growth of existing individuals. This term is non-zero for all age-size classes and both size categories (small and large). Where: U is light limitation (set to 1.0 for reef builders, while the mangroves use the same formulation as in equation (5.6) for the seagrass and macroalgae); μ_s is the rate of horizontal growth for small habitat; Ψ is the sediment suitability rating for the habitat polygon (proportional presence of gravel and sand (Jones, 1973; McLoughlin & Young, 1985), scaled so that 1.0 is equal to perfect sediment composition); λ is the index of spread for the logistic growth function; and ν is the inflexion point of the logistic growth function.

5.1.2 Recruitment

The second term in (5.1) is recruitment of new individuals (and initiation of new patches), this is the other way the proportional coverage of benthos can increase. Where: ξ is the rate of recruitment of new small habitat specimens (only non-zero for the smallest class); and p_T is the proportion of the North West Shelf region covered with habitat. A constant recruitment term (κ_{rec}) was added to the final formulation used in *NWS-InVitro* as regional self seeding was not sufficient to allow for habitat patch recovery rates of the order seen in reality. This constant can be considered to represent those larvae coming in from outside the modelled area. In addition, it is an abstract mitigation for the failure to capture very fine scale seafloor details in this benthic model (such fine scale features are likely to be the core of reestablishment colonies in the real world, but for reasons of computation constraints they are much too fine to represent in regional scale models).

To allow for more generic representations for recruitment (and growth) than is required for the North West Shelf reef habitat, a depth dependency for the rate parameters used in the small benthos was also developed. This depth related dependency was necessary for macrophytes (due to their light requirements), but is probably not required for benthos in most cases however, as sediment dependency may be sufficient.

5.1.3 Mortality

The third term in (5.1) is the mortality term. The first part is natural background mortality and the second catastrophic mortality due to fishing and cyclones. In this case κ_s is the natural mortality rate of small habitat; θ is the index of spread for the logistic age-structured natural mortality function ; φ is the inflexion point of the age-based natural mortality function; Φ_j ~U(0,1), with the sum of Φ_j over j is equal to one (this is to avoid the assumption of homogeneous distribution of all age-size classes without necessitating subgrid scale spatial monitoring of patch composition); and $D_{s,t}$ is the damage done to small habitat by cyclones, dredging and fishing at time t (a simple percentage overlap of the track of the cyclone or trawl and the polygon is used in a catch equation with vulnerability constants from Hall (1999) to give this damage contribution).

5.1.4 Ageing and vertical growth

The final two terms in (5.1) deal with growth in the vertical plane, that is both growth (and aging) up through the classes of small benthos and the transition from small to large benthos, where: ω is the vertical growth rate of small habitat (equivalent to aging); ϕ is the index of spread for the logistic function for the transition to large habitat; and ε is the inflexion point of the transition function.

5.1.5 Formulation note

Logistic functions were used in this formulation for growth, mortality and transition to large benthos across age classes of small benthos so that age-size dependency was present. Alternative functions (particularly alternative asymptotic functions) could have been used instead. Sensitivity to this formulation assumption has not been considered in depth.

5.2 Small benthos – fragmentation

The rate of change of fragmentation of small benthos per polygon (B_s) is given by:

$$\frac{dB_s}{dt} = \sum_j \left(\Phi_j \cdot A_{j,t-dt} \cdot D_{s,B,t} - \Theta_j \cdot \frac{j^2}{\chi^2} \cdot \left(\begin{array}{c} \dfrac{A_{j,t-dt} \cdot p_{s,t-dt} \cdot \left(1 - p_{s,t-dt}\right) \cdot U \cdot \mu_s \cdot \Psi}{1 + e^{(-\lambda (j - \nu))}} \\ + U \cdot \xi \cdot \Psi \cdot (1 - e^{(-p_T)}) \end{array} \right) \right) \quad (5.2)$$

where Θ_j is the proportion of the edge of unfragmented sections of the habitat of age j in this polygon that have access to fragmented areas; χ is the number of age classes in small benthos; and $D_{s,B,t}$ is the new fragmentation of small habitat in this polygon due to cyclones, dredging and fishing.

5.3 Large benthos – cover

Large habitat refers to the large bodied habitat defining species (e.g. some of the largest corals and sponges). In practice, in this implementation, large specimens are considered to be >25 cm tall for reef builders and >100 cm for mangroves – heights again taken from data classification scheme used in benthic sampling program (Fulton et al. 2006a).

The rate of change in percentage cover of large habitat (p_L) is handled slightly differently as it is not age structured and is given by:

$$\frac{dp_L}{dt} = \left(\begin{array}{l} p_{L,t-dt} \cdot \left(1 - p_{L,t-dt}\right) \cdot U \cdot \mu_L \cdot e^{\left(-\left(\varpi \cdot d_m + \frac{\zeta}{\Psi}\right)\right)} + \sum_j \left(\frac{A_{j,t-dt} \cdot p_{s,t-dt} \cdot \omega}{1 + e^{\left(-\phi \cdot (j - \varepsilon)\right)}} \right) \\ + A_{\chi-1,t-dt} \cdot p_{s,t-dt} \cdot \omega - (\kappa_L + D_{L,t}) \cdot p_{L,t-dt} \end{array} \right) \quad (5.3)$$

The terms of (5.3) correspond to growth, recruitment, mortality and are explained in the following text.

5.3.1 Horizontal growth

The first term deals with horizontal growth, where μ_L is the rate of horizontal growth for large habitat; d_m is the seabed depth in metres; ϖ is the coefficient of the depth effect on horizontal growth of large habitat; and ζ is the coefficient of the sediment effect, Ψ, on the horizontal growth of large habitat.

5.3.2 Ageing and vertical growth

The second and third terms in equation (5.3) is the growth of the cover of large benthos due to the vertical growth (and ageing) of small benthos. With: χ is the number of age-size groups of small habitat (set to 10 here)

5.3.3 Mortality

The final term is the mortality term, where: κ_L is the natural mortality rate of large habitat; and $D_{L,t}$ is the damage done to large habitat by cyclones, dredging and fishing at time t (which is calculated in the same way as for $D_{s,t}$).

5.4 Large benthos – fragmentation

Lastly, the rate of change of fragmentation for large habitat (B_L) is calculated using:

$$\frac{dB_L}{dt} = \left(D_{L,B,t} - \Theta_L \cdot \left(\begin{array}{l} p_{L,t-dt} \cdot \left(1 - p_{L,t-dt}\right) \cdot U \cdot \mu_L \cdot e^{(-(\varpi \cdot m + \frac{\zeta}{\Psi}))} \\ + \sum_j \left(\frac{A_{j,t-dt} \cdot p_{s,t-dt} \cdot \omega}{1 + e^{(-\phi \cdot (j - \varepsilon))}} \right) + A_{\chi-1,t-dt} \cdot p_{s,t-dt} \cdot \omega \end{array} \right) \right) \tag{5.4}$$

where Θ_L is the proportion of the edge of unfragmented sections of the large habitat in this polygon that have access to fragmented areas; and $D_{L,B,t}$ is the new fragmentation of large habitat in this polygon due to cyclones, dredging and fishing.

5.5 Macrophyte – cover and fragmentation

The formulation used for macrophytes (seagrass and macroalgae) is very similar to that for large benthos (equations (5.3) and (5.4)), with percent cover (p_m) given by:

$$\frac{dp_m}{dt} = \left(p_{m,t-dt} \cdot \left(1 - p_{m,t-dt}\right) \cdot U \cdot \mu \cdot e^{(-(\varpi \cdot d_m + \frac{\zeta}{\Psi}))} - (\kappa + D_t) \cdot p_{m,t-dt} \right) \tag{5.5}$$

where

$$U = \min\left(1.0 \ , \ I_{top} \cdot \exp\left(-\gamma \cdot m\right)\right) \tag{5.6}$$

and fragmentation (B) given by:

$$\frac{dB}{dt} = \left(D_{B,t} - \rho \cdot p_{m,t-dt} \cdot \left(1 - p_{m,t-dt}\right) \cdot U \cdot \mu \cdot e^{(-(\varpi \cdot m + \frac{\zeta}{\Psi}))} \right) \tag{5.7}$$

where μ is the rate of horizontal growth; d_m is the seabed depth in metres; ϖ is the coefficient of the depth effect on horizontal growth of large habitat; and ζ is the coefficient of the sediment effect on the horizontal growth of large habitat; κ is the natural mortality rate; D_t is the damage done by cyclones, dredging and fishing at time t; I_{top} is the level of irradiance at the sea surface; γ is the extinction coefficient (there are different onshore and offshore values for the North West Shelf due to the levels of inshore turbidity); ρ is the proportion of the edge of unfragmented sections that have access to fragmented areas and $D_{B,t}$ is the new fragmentation due to cyclones, dredging and fishing.

6. BLASTULA AGENTS

Blastula agents are another subclass of Polyorganism agents. In *NWS-InVitro* Blastula agents are the primary means of representing juveniles that live separately from the adult population. A Blastula agent consists of a queue (list) of entries so that multiple cohorts of different ages can be stored in the one agent-type. In this way a single Blastula agent can represent all of the juveniles of a taxon in a specific general area. No spatially explicit locations are given to these individuals they are just assumed to be at some point within that area.

6.1 Inducting new juveniles

When agents, such as Animal agents, reproduce, the offspring are passed to Blastula agents where they stay until they mature. This induction is done by adding a new entry to the blastula queue – the initial number of offspring in that new entry in the blastula queue and the time at which they will mature.

6.2 Processes while juveniles

6.2.1 Growth

In *NWS-InVitro*, where flora and fauna are assumed to have access to sufficient resources to grow as needed, the growth of individuals within a Blastula agent is done by assuming that growth follows a variant of the von Bertalanffy curve. Periodically (at a user defined interval) the total biomass for the Blastula agent is checked. This biomass is calculated as the product of the blastula's membership and their ideal masses at their current ages. If this total biomass exceeds the carrying capacity for the area (based on habitat quality) then the excess is culled immediately. This cull is carried out using the formulations detailed in section 6.2.2. The relationship used to calculate the carrying capacity of the area covered by the blastula ($K_{t,local}$) is:

$$K_{t,local} = \begin{cases} \max\left(0.000000001, \dfrac{\kappa_{cap,a} \cdot B_{hab,t}}{B_{habarea,t}}\right) & , B_{hab,t} > 0 \text{ AND } B_{habarea,t} > \kappa_{cap,b} \cdot N_t \\ \max\left(0.000000001, \dfrac{B_{habarea,t}}{\kappa_{cap,b}}\right) & , \text{ otherwise} \end{cases} \quad (6.1)$$

where $B_{hab,t}$ is the biomass of suitable habitat under the Blastula agent; $B_{habarea,t}$ is the total area (in square metres) of suitable habitat under the Blastula agent; $\kappa_{cap,a}$ is the taxon specific mass (in kg) of settled juveniles supported per kg/m^2 of habitat; $\kappa_{cap,b}$ is the taxon specific minimum area (in square metres) required to support one kg of settled juveniles; N_t is the total membership of the Blastula agent. If this local carrying capacity exceeds the user defined total carrying capacity for the entire model domain then the local carrying capacity is reduced to match the value of the overall carrying capacity.

Options have been included for the use of the basal or false metabolism representations defined for Animal agents (detailed in section 2.1.5). These can be used with or instead of the carrying capacity check discussed above.

6.2.2 Mortality

Mortality is applied at the level of individual members of the Blastula agent not at the level of the agent itself. A Blastula agent with no members functions by waiting for a contribution. As mentioned above, mortality is based on the total biomass represented by a Blastula agent relative to the local carrying capacity. This is implemented as:

$$M_{t,i} = N_{t,i} \cdot \left(1 - \omega_{M,i}\right) \cdot \exp\left(\frac{\log\left(\omega_{M,i}\right)}{a_{mat}} \cdot dt\right) \tag{6.2}$$

where $M_{t,i}$ is the number of individuals within the blastula at time t in cohort i that are culled; $N_{t,i}$ is the membership of cohort i before mortality is applied; a_{mat} is the age of maturity (converted to a value in seconds when the original parameter file is read in); and $\omega_{M,i}$ is calculated as:

$$\omega_{M,i} = \max\left(0.01, \min\left(\left(1 - \frac{\kappa_m^o}{s_{yr}}\right), \frac{K_{local,t}}{N_{t,i} \cdot w_{t,i}}\right)\right) \tag{6.3}$$

with $w_{t,i}$ the current individual weight of members of cohort i; $K_{local,t}$ is the local carrying capacity; s_{yr} is the number of seconds in a year; and κ^o_m is the base mortality rate.

6.3 Maturing out of blastula

At each time-step, each Blastula agent checks for cohorts that have matured (that is, if date of maturity for the cohort stored in the blastula queue is within a taxon specific user specified number of days of the current time). The members of a mature cohort are allocated to the youngest age classes of any Population agents in range, or new Animal agents are created, depending on the specific configuration being used for that taxon.

If Animal agents are created then the number of agents (N_A) created from the maturity cohort is given by:

$$N_A = \left\lfloor \frac{N_{i,t}}{N_{std} + 1} \right\rfloor - M_{t,i} \tag{6.4}$$

where N_{std} is the standard membership for Animal agents of that taxon; and $N_{i,t}$ is the membership of the maturing cohort. Note the final term in 6.4 indicates that a cull is imposed on the cohort immediately before they are allowed to recruit to the next life history stage. The individual mass at age for the members of the cohort leaving the blastula is taken from a variant of the von Bertalanffy growth curve, such that:

$$w_{t,i} = \left(w_{\max} - w_0\right) \cdot \left(1 - \exp\left(-\frac{\lambda_{mass} \cdot a_{t,i}}{s_{yr}}\right)\right) + w_0 \tag{6.5}$$

with s_{yr} is the number of seconds in a year; $a_{i,t}$ the current age of cohort i (in seconds); λ_{mass} is the taxon specific growth correction coefficient; w_{max} is the maximum individual mass for that taxon and w_0 is the mass of a larva. Before a new agent is created the total number of existing agents is checked (there are upper bounds on what is computationally effective to simulate) and the local neighbourhood is polled to check

for any existing newly settled Animal agents of the same taxon (e.g. if the animals were fish they would join an existing school of the right age range). If no existing suitable agents are found in the local neighbourhood then a new Animal agent is formed.

The maturation of cohorts that enter population agents is slightly different. Firstly the distance to each active population agent is checked and the cohort is distributed evenly amongst all populations within a taxon specific maturation range (so if the population is within range of the blastula it will receive settlers from that blastula). If no populations are found within range the cohort will found a new population at its current location.

Weighting settler allocation by distance has also been explored using:

$$R_{i,t} = \frac{\left(N_{i,t} - M_{i,t}\right) \cdot \exp\left(-\lambda_{juvdist} \cdot \sqrt{\left(x_{i,t} - x_{pop,t}\right)^2 + \left(y_{i,t} - y_{pop,t}\right)^2 + \left(z_{i,t} - z_{pop,t}\right)^2}\right)}{\sum_{popj} \exp\left(-\lambda_{juvdist} \cdot \sqrt{\left(x_{i,t} - x_{popj,t}\right)^2 + \left(y_{i,t} - y_{popj,t}\right)^2 + \left(z_{i,t} - z_{popj,t}\right)^2}\right)} \quad (6.6)$$

where $R_{i,t}$ are the number of settlers from cohort i entering population *pop*; and $\lambda_{juvdist}$ is the coefficient for steepness in the resulting spatial distribution. Given the abstract form of what a population and blastula represents at these regional scales, in the case of *NWS-InVitro* the unweighted allocation produced more reasonable biomass distributions through time and so the weightings based on distance were not used.

Once a cohort has matured out of the blastula (whether to an Animal or Population agent) then that entry is deleted from the blastula's queue.

7. LARVA AGENTS

Another subclass of the Polyorganism agent is the Larva agent. Larva agents are an alternative to Blastula, which explicitly undergo stage specific behaviours such as advection-diffusion, directed movement to settlement sites, juvenile growth, and directed movement to adult sites. While Blastula agents are typically used at the broad scale regional level (e.g. in the NWSJEMS MSE analysis; Fulton et al. 2006b), Larva agents can be used when dispersal or small scale issues are of primary concern. Larva agents were also used to guide the development of the blastula agent (to ensure consistency). Thus for completeness a discussion of their formulation is given here, broken up into the four major developmental stages a larval agent can display/represent.

7.1 General behaviour

Irrespective of developmental stage all Larva agents use the same basic growth forms. In the unsettled (free-floating) phase or if there is no dependency on habitat then growth is given by:

$$B_t = B_{t-dt} \cdot \exp\left(\lambda_{mass} \cdot dt\right) \tag{7.1}$$

where λ_{mass} is the taxon specific growth correction coefficient; and B_t is the biomass of the larval agent. If the larval agent has settled and there is habitat dependency then growth is given by:

$$B_t = \frac{B_{t-dt} \cdot K_t}{\left(K_t - B_t\right) \cdot \exp\left(-\lambda_{mass} \cdot dt\right) + B_t} \tag{7.2}$$

with K_t the carrying capacity of the habitat area the larval agent is settled in, which is given by:

$$K_t = \begin{cases} \dfrac{A_{hab,t}}{K_\beta} & , \; A_{hab,t} > B_{hab,t} \cdot K_\beta \text{ AND } B_{hab,t} > 0 \\ K_\alpha \cdot \dfrac{B_{hab,t}}{A_{hab,t}} & , \text{otherwise} \end{cases} \tag{7.3}$$

where $A_{hab,t}$ is the area of the habitat patch; $B_{hab,t}$ is the biomass of the habitat patch; K_α is the biomass supported by every kgm^{-2} of habitat; and K_β is the area (m^2) required to support one kg of larvae.

7.2 Free floating stage

The free floating stage of a Larva agent performs all the behaviours defined for polyorganisms (see chapter 4), such as advection-diffusion, but they do not possess any other specific behaviour of their own.

7.3 Settlers

7.3.1 Movement

The free floating stage of the larval agents do not show directed movement, but are simply advected and diffused. Once the agent is old enough to settle it tries to find a suitable site (which may be chosen randomly or it may be the closest suitable site). Movement is handled by moving each vertex of the polygon representing the patch the Larva agent occupies, using the movement equation given for Animal agents in (2.1). An option is included for the Larva agent to use diel vertical migration to give them directed movement.

Diel vertical migration and directed movement

When using directed movement, the Larva agent checks its orientation with the coastline by comparing against a baseline coordinate in xyz space (in this case (1,-1,0)). The direction of travel is then determined using the following trigonometric relationship:

$$\cos\theta_{dir} = \frac{x_t \cdot x_{base} + y_t \cdot y_{base}}{\sqrt{2 \cdot \left(x_t^2 + y_t^2\right)}} \tag{7.4}$$

where θ_{dir} is a proxy for the current angle to the coastline; and x_t and y_t the current location of the cental point of the Larva agent. This angle is then used to modify the weighting coefficients for wind and current contributions to movement. If the direction of the water flow is not in the direction the Larva agent needs to go then the coefficients are set to zero and the Larva agent effectively hugs the bottom. If, on the other hand, water flow is in the correct direction then the coefficients are left at the values in the parameter file and equation (2.1) is used unmodified. In the case of the North West Shelf, the critical values of $\cos\theta_{dir}$ were +/- 0.7, and the Larva agents would only move when $\cos\theta_{dir} > 0.7$ (water flowing toward the coast) thus taking them to inshore nursery habitats. Larva agents which have pelagic offshore juvenile habitat only moved when $\cos\theta_{dir} < -0.7$ (water is flowing away from the coast).

If the Larva agents are parameterised to be completely marine and the movement of the agent leaves it washed up on the coast, then the limbs of the polygon that are on land are retracted back towards the centre of the polygon (half of the distance between the original central point and the current vertex point with each step of the contraction). If after five such contractions the limbs of the polygon are still beached then those points are withdrawn to the polygon's original central point. If the centre point of the polygon is beached then the Larva agent is assumed to die off. Obviously any species that needs to reach land to "settle" is not affected in this way and continues on with its life cycle once it has beached.

7.3.2 Settlement

At each new location the Larva agent reaches a check is performed to see if it has reached the habitat patch originally selected. Once inside the patch the Larva agent settles (and moves to the next developmental stage), if no patch is found the Larva agent continues to float until it dies.

7.4 Juveniles

Patches of settled juveniles only perform the behaviours defined for polyorganisms (see chapter 4), such as growth, and do not possess any other specific behaviour of their own.

7.5 Maturing sub-adults

7.5.1 Movement

Larva agents that have settled, entered the juvenile state and survived to the age of recruitment, attempt to move to adult habitat and become mature adults (which may be of a different agent type, such as animal or population). If the Larva agent does not mature before the adult age of reproduction then the patch is assumed to have died without successfully reaching its new habitat and the agent is terminated. The migration of maturing sub-adults is handled similarly to the directed movement of settlers in section 7.3.

While moving, the area covered by the Larva agent (i.e. the actual size of the polygon) contracts, so that by the time it reaches maturity the area of the polygon matches the area of the adult agent type. This re-scaling is done with a simple power function such that:

$$A_{scale} = \kappa_{aggreg}^{dt} \tag{7.5}$$

where A_{scale} is the scaling rate used to contract the larva polygon; and κ_{aggreg} is the taxon specific rate of aggregation specified in the parameter file.

7.5.2 Maturing out of Larva agents

This is handled much as for Blastula agents (see chapter 6). The allocation of larvae (from a Larva agent) to populations is also dealt with in section 3.3 with the number of larvae of recruitment age ($R_{j,r}$) given by:

$$R_{j,t} = \frac{B_t}{\rho_m \cdot w_{t,m} + (1-\rho_m) \cdot w_{t,f}} \tag{7.6}$$

Where B_t is the mass associated with the Larva agent's polygon; $w_{t,i}$ is the taxon specific weight at maturity of sex i –m for males, f for females – (from equations (3.6) and (3.7)); and ρ_m is the proportion of the population made up males.

If the larvae are maturing into Animal agents then the number of recruits ($N_{R,t}$) is given by:

$$N_{R,t} = \frac{B_t}{w_t} \tag{7.7}$$

where B_t is the biomass of the Larva agent polygon; and w_t is the weight of the individual sub-adults calculated as:

$$w_{t,i} = \left(w_{\max} - w_0\right) \cdot \left(1 - \exp\left(-\frac{\lambda_{mass} \cdot a_{t,i}}{s_{yr}}\right)\right) + w_0 \tag{7.8}$$

with s_{yr} is the number of seconds in a year; $a_{,t}$ the current age of the sub-adult Larval agent (in seconds); λ_{mass} is the taxon specific growth correction coefficient; w_{max} is the maximum individual mass for that taxon and w_0 is the mass of an individual larva. Once the Larva agent has recruited to the adult population that Larva agent is removed from the model's agent list.

8. ADVISER AGENTS

Adviser agents are subclasses of Monitor agents and are used in two ways in *InVitro*. The first is as an overall spatial record keeper, where every taxon in the model is recorded in the same grid for easy visualisation of overall biomass patterns when considering model output. This role is what determines the dimensions of the grids used by the Adviser agents – advisers store the biomass of each taxon in each cell at two resolutions – a fine scale grid and an overall total value.

The second role of the Adviser agents in *InVitro* is as a general habitat and environmental conditions monitor. When an agent in *InVitro* requires general information on its surroundings (e.g. when doing a habitat gradient movement search) then it polls the adviser for the value of the desired habitat components in the area it is searching. The adviser is not called upon when an enquiring agent needs specific information on its immediate area (i.e. what agents are immediately surrounding it, in that case it relies upon neighbourhood lists).

8.1 Filling adviser grids

Each time the adviser activates it begins by resetting the grid to zero. Then the entire agent list is stepped through, with each agent locating which adviser grid cell it falls in and updating the biomass value of that taxon in that grid cell. The first step in this update is a check to see whether the agent is within the domain of the adviser. If the agent is within the adviser's domain then the index of the grid cell that the centre of the agent is sitting in is determined using:

$$i = \left\lfloor \frac{x_a - x_{ll}}{d_x} \right\rfloor \tag{8.1}$$

$$j = \left\lfloor \frac{y_a - y_{ll}}{d_y} \right\rfloor \tag{8.2}$$

where i is the index in the x direction; j is the index in the y direction; x_a is the x coordinate of the updating agent; x_{ll} is the lower left x coordinate of the Adviser agent; d_x is the length of the side of individual adviser grid cells in the x direction; y_a is the y coordinate of the updating agent; y_{ll} is the lower left y coordinate of the Adviser agent; and d_y is the length of the side of individual adviser grid cells in the y direction.

Once the central grid cell indices have been found then the biomass values are updated. The biomass of the agent is allocated homogeneously across all the grid cells overlapped by the agent. The radius of the overlap (r) is given by:

$$r_i = \frac{r_{ag}}{d_i} \tag{8.3}$$

where i is the direction of interest (x or y); r_{ag} is the radius of the agent; and d_i is the length of the side of the individual adviser grid cells in that direction.

The indices of the adviser grid cells that are then updated are (i-r_i, j-r_j) to (i+r_i, j+r_j) – unless these indices extend beyond the number of cells in the adviser (so the $i \pm r_i$ is bounded below by 0 and above by the number of grid cells in the x direction of the adviser's domain, and similarly for the $j \pm r_j$). The total number of grid cells that are overlapped (n_{over}) is calculated as:

$$n_{over} = \left(i + r_{i+} - \left(i - r_{i-}\right)\right) \cdot \left(j + r_{j+} - \left(j - r_{j-}\right)\right) \tag{8.4}$$

with r_{i-} signifying the final lower overlap (which may have been truncated if it extended beyond the adviser's bounds) and similarly for r_{i+}, r_{j-}, and r_{j+}. The final amount of biomass allocated to each of the adviser grid cells (B_{up}) comes from:

$$B_{up} = \frac{B_{ag}}{n_{over}} \tag{8.5}$$

with B_{ag} the biomass of the updating agent. The total overall value for the taxon is also updated (simply be incrementing it by B_{ag}). The means of calculating of the biomass contributed by each updating agent is dependent on the type of that agent (see below).

Population agents

For Population agents the biomass B_{ag} is given by:

$$B_{ag} = \sum_k p_k \cdot \left(\left(1 - p_m\right) \cdot w_{k,f} + p_m \cdot w_{k,m}\right) \tag{8.6}$$

with p_k the proportion of the population in age group k; p_m the proportion of the population that is male; $w_{k,m}$ the weight of males in age group k; $w_{k,f}$ the weight of females in age group k.

Blastula agents

The biomass B_{ag} of Blastula agents is a simple sum of all the members of the various cohorts such that:

$$B_{ag} = \sum_k w_{ag,mat} \cdot N_k \tag{8.7}$$

where $w_{ag,mat}$ is the taxon specific weight at maturity (when the members would leave the blastula, due to computation demands this is used as a proxy for the weight rather than calling up age specific weights); and N_k is the number of individuals in cohort k.

Animal and Thing agents

The biomass B_{ag} of Animal and Thing agents is simply the mass of the individuals multiplied by the number of individuals represented by the agent.

Larva and Polyorganism agents

For Larva and Polyorganism agents the biomass B_{ag} must be found by summing over all the polygons in the area of the adviser. In this case the general procedure for finding which adviser grid cells to update is modified. The grid cell the centre point of each polygon lies in is found and the biomass of the polygon is added only to that grid cell.

Benthic agents

The procedure for the Benthic agents is slightly modified compared to the general procedure described above. For computational efficiency when the advisers are first initialised a record is kept of the indices of the gridded Benthic agent cells that fall within each adviser grid cell. Similarly the number of adviser boxes each benthic box is allocated across is recorded. Then each time the adviser is updated the biomass in the benthic cells that match each adviser cell is cross-referenced and the adviser cell value is updated using equation (8.5). If the Benthic agent is not represented by a regular grid then the same procedure as for Larva and Polyorganism agents is used.

CSurface Agents

For CSurface agents the biomass update per adviser grid cell does not use the general procedure outlined above. Instead the value of the CSurface at the centroid of the adviser grid cell is read in directly as the updated biomass value B_{ag}.

8.2 Retrieving values from adviser grids

As mentioned previously, when an agent in *InVitro* requires general information on its surroundings it queries the adviser agents on the condition of the resources in a site it is interested in. To maximise efficiency in this query a "meta-adviser" (an adviser which only tracks other advisers) is used as a filter on the query so that only one query needs be made for any one area. The "meta-adviser" handles the query and sums the values in the relevant grid cells of all advisers located within the area of interest. The answer returned to the agent making the enquiry consists of the average biomass of the resource of interest in the area of interest and the area covered by that resource.

8.3 Output from adviser grids

At intervals defined by the user, the contents of the adviser grids are written to flat text output files. These output files come in two forms. The overall value of each adviser is reported in a single combined file maintained by the "meta-adviser". This file is spatially explicit in that the values reported correspond to the domain of each adviser which is treated as a cell in a larger, coarsely resolved grid covering the entire model domain (i.e. the network of advisers when viewed through the meta-adviser makes up a one by one degree grid).

The other form of output file created by the advisers is a record of the values in each fine-scale grid cell of each adviser. This can either be considered on an adviser-by-adviser basis (so each adviser's fine-scale grid is evaluated independently in separate files) or via the meta-adviser, where all of the fine-scale grids are combined into a large grid spanning the entire model region. That latter is a very large grid, which can present problems for visualisation so it is more typical to consider the finer grids individually.

9. CATASTROPHE AGENTS

Catastrophic events are an important part of tropical systems such as the North West Shelf of Australia. To represent these events a subclass of Monitor agents was created – Catastrophe agents. In *NWS-InVitro* there are two main types of catastrophic events: natural events (e.g. cyclones) and anthropogenic actions (in this case dredging). The list of catastrophes that will occur in a simulation run are read-in from a file at the beginning of the run (which details their timing, spatial location and intensity). These are then entered into a list of events with the event at the top of the list the first one to occur. Once an event has occurred it is removed from the top of the list, and the next event becomes the current one, with its time of execution inserted into the time queue at that point. The time-step of the catastrophe agent is set so that its next step will be at the date associated with the new catastrophe at the top of the event list.

9.1 Cyclones

Cyclones are represented by taking the paths of real cyclones and applying them as a series of rectangular "footprints" that potentially damage or kill everything they overlap. These footsteps are defined by giving a start and endpoint and the width of the "path of destruction" either side of the straight line joining these points. The methods used to apply damage to the different kinds of agents that may be impacted by catastrophes are outlined below.

9.1.1 Damaging polyorganisms

When damaging polyorganisms that are not Benthic or Blastula agents, the form of the catastrophe and the degree of overlap with the polygon determine the outcome of the interaction:

1. If the source of the damage is marked as immediately lethal (e.g. destruction of sea bottom features by trawl gear) then any polygons completely covered by the footprint is terminated (the polygon is killed off and removed from the polyorganism).
2. If a polygon is bisected by the footprint (figure 9.1.1) then the section under the footprint is excised (dark grey area in figure 9.1.1) and new polygons are created from those parts of the original polygon that were outside the footprint (the white areas in figure 9.1.1).

Similar methods are used if the footprint is for a contaminant (e.g. acute toxicant contamination or smothering by an oil spill) rather than a catastrophe. A more detailed description of contaminant agents and the interaction with polyorganisms can be found in chapters 4 and 10 of this report.

Note that if the polyorganism is implemented as a regular grid then the above procedure is slightly modified. In this case any polygons completely under the footprint have their values degraded (if immediately lethal catastrophe then the value is set to zero), but the actual polygon is not removed (just emptied). Any polygons that are intersected by the footprint, but not completely covered, have their value reduced (based on the area of overlap) and their shape remains unchanged (no new polygons are created).

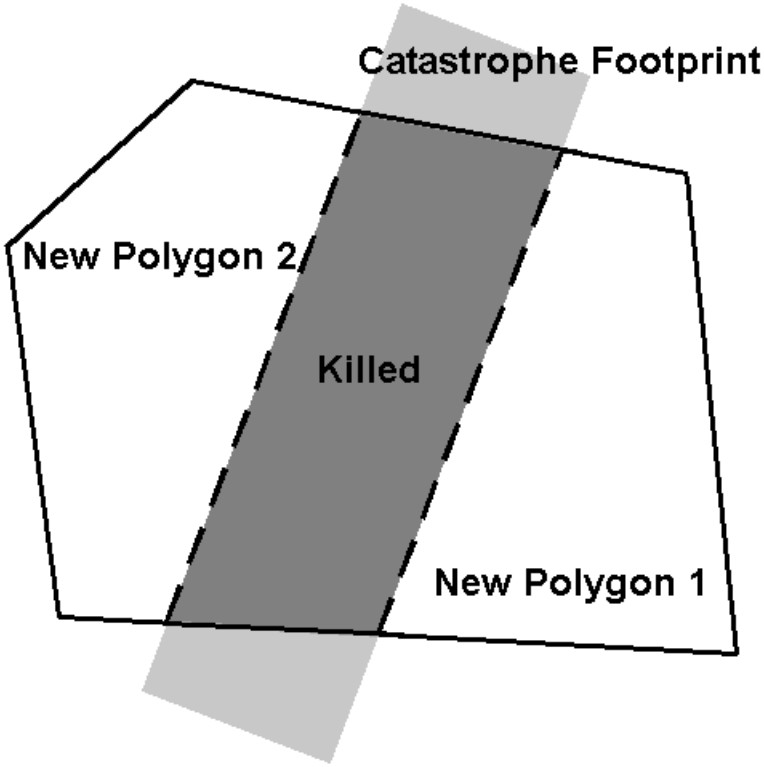

Figure 9.1.1: Diagram of the splitting of a polygon cut by the footprint of a catastrophe (grey rectangle). The dark grey area marks the intersection of the footprint and the original polygon (solid black edge); this area of the polyorganism is killed off and two new polygons are created.

9.1.2 Damaging Benthic agents

Catastrophic damage to a Benthic agent is similar to that for a polyorganism implemented on a regular grid. However, instead of immediately applying any damage done, the intensity of the pressure on that cell is recorded and when the benthic organism next acts that pressure is used in the mortality term in equations (6.1), (6.3) or (6.5).

The intensity of the impact of the catastrophe is determined by the following probabilistic relationships. These relationships are split into a decision tree (outlined in figure 9.1.2).

The first fork of the tree is given by the following test. A uniform random number ~U(0,1) is generated and if this is less than the proportion of the current benthic grid cell covered by small benthos (p_s), then the patch of small benthos is marked as hit if:

$$p_f > p_s \tag{9.1}$$

where the proportion of the grid cell under the footprint of the catastrophe (p_f) is given by:

$$p_f = \min\left(1, \frac{a_{cat,i}}{a_i}\right) \tag{9.2}$$

and $a_{cat,i}$ is the area of the catastrophe's footprint and a_i is the agent's area.

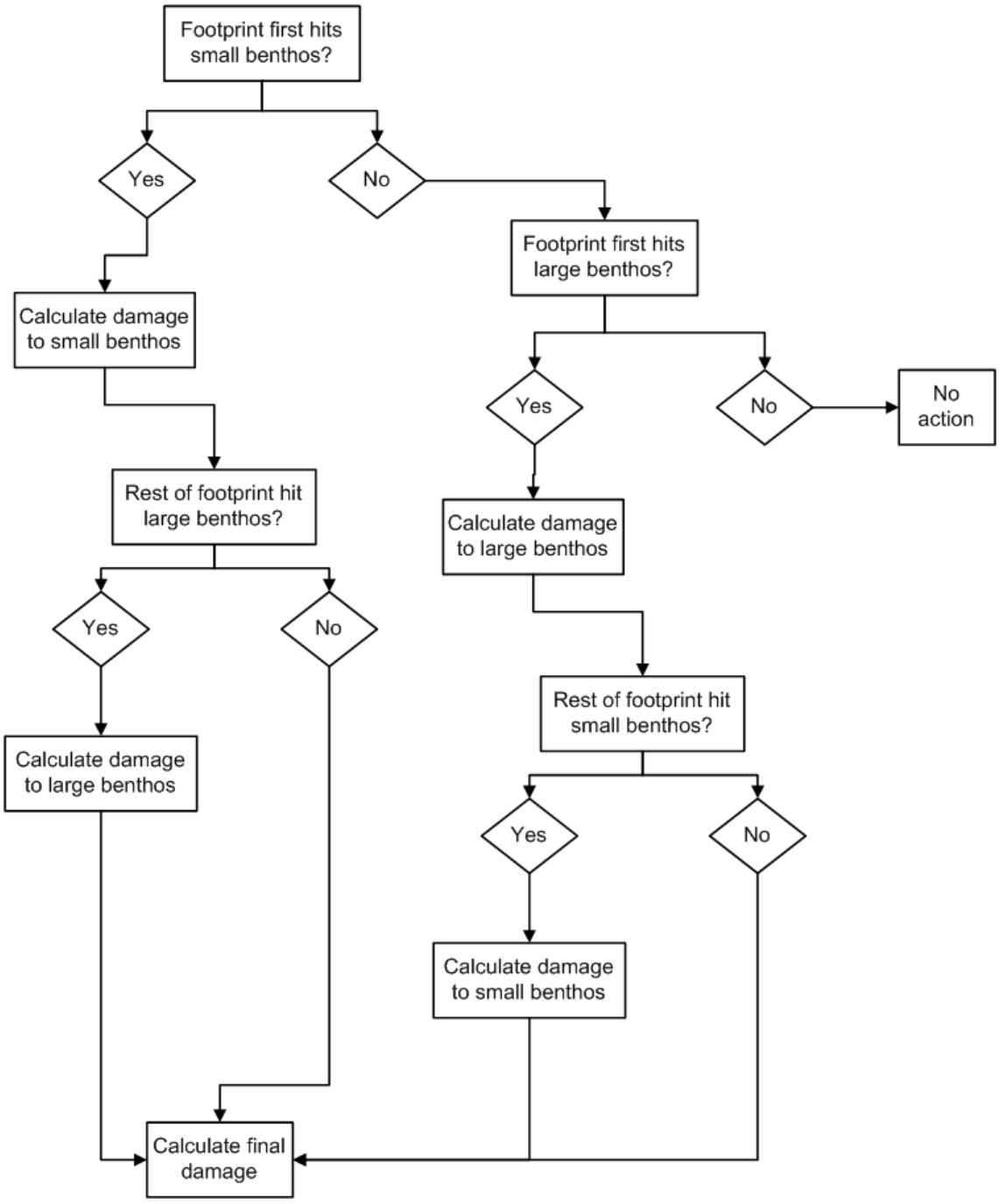

Figure 9.1.2: Decision tree used to calculate total damage to benthic habitat due to interactions with a Catastrophe agent.

If inequality (9.1) holds true then the procedure continues down that arm of the decision tree using the following equations (9.3) to (9.6); beginning with the calculation of remainder of the footprint (p_{lf}) as:

$$p_{lf} = \left(p_f - p_s\right) \cdot \left(\frac{p_L}{1 - p_S}\right) \tag{9.3}$$

with p_L the proportion of the current cell covered by large benthos. The value of p_{lf} is compared with another random number (~U(0,1)) and if p_{lf} is larger than the random number then the large benthos has also been hit. Once the patches have been marked as hit then the magnitude of the impact is determined using:

$$\begin{aligned} D_{L,t} &= \eta_L \cdot \kappa_{L,cat} \\ D_{s,t} &= \eta_s \cdot \kappa_{s,cat} \end{aligned} \tag{9.4}$$

where $\kappa_{L,cat}$ is the mortality coefficient for large benthos impacted by a catastrophe; $\kappa_{s,cat}$ is the mortality coefficient for small benthos impacted by a catastrophe; and the spatial overlap scalars η_L and η_s are given by:

$$\eta_L = \begin{cases} 1 & , p_L < p_f - p_S \\ \dfrac{p_f - p_s}{p_L} & , \text{otherwise} \end{cases} \tag{9.5}$$

$$\eta_s = \frac{p_f}{p_s} \tag{9.6}$$

If inequality (9.1) does not hold then the other arm of the decision tree is used, which employs equations (9.7) to (9.9). The first step on this arm is to compare the original random number against the total proportion of the cell covered by benthos (p_s+p_L). If the random number is less than the total cover then it is assumed a patch of large benthos is hit if:

$$p_f > p_L \tag{9.7}$$

In this case the remainder of the footprint (p_{lf}) is given by:

$$p_{lf} = \left(p_f - p_L\right) \cdot \left(\frac{p_s}{1 - p_L}\right)$$

In a mirror of the other arm of the decision tree, if the value of p_{lf} is larger than a newly generated random number (~u(0,1)) then the small benthos has also been hit and the magnitude of the impact is determined using (9.4) with:

$$\eta_s = \begin{cases} 1 & , p_s < p_f - p_L \\ \dfrac{p_f - p_L}{p_s} & , \text{otherwise} \end{cases} \tag{9.8}$$

$$\eta_L = \frac{p_f}{p_L} \tag{9.9}$$

9.1.3 Damaging animals and blastula

Damage to Animal and Blastula agents is much more straightforward than for Benthic agents. If the Animal agent is within the swath (footprint) of the catastrophe and the following inequality (9.10) holds then mortality is imposed on the Animal agent using equation (9.11).

The inequality used to see if the animal is hit is:

$$\zeta \cdot \log(D_I + 1) > 1 - p_D \tag{9.10}$$

where ζ ~U(0,1); D_I is the intensity of the catastrophe; and p_D is the probability offset (making the catastrophe more effective if non-zero) associated with the catastrophe (from the catastrophe forcing file).

The mortality (M_C) resulting from the catastrophe is as follows:

$$M_C = N_t \cdot \kappa_{cat} \tag{9.11}$$

with N_t the membership of the Animal agent; and κ_{cat} is the taxon specific mortality rate associated with catastrophic events (e.g. cyclone strikes).

9.1.4 Damaging vessels and boats

Vessels and boats caught in catastrophes also use the inequality (9.10) defined for Animal agents above. If the equality holds then the vessel is wrecked, but if the inequality does not hold then the vessel hoves to, rides out the event in some sheltered spot, and survives unscathed.

9.2 Dredging

The path of the dredge is defined in the same way as for cyclones (that is a series of rectangular footprints, defined using a start and end point and the width of the "path of destruction"). Any polyorganisms or Benthic agents under these footprints are damaged. The damage is applied in the same way as for cyclones, but with dredge specific rather than cyclone specific parameters (for the offset, mortality rates, etc).

10. CONTAMINANT AGENTS

Contaminant plumes in *NWS-InVitro* may be represented as either Tracers or grid-based CSurface agents. Tracers are computationally expensive, and their use is reserved for *ad hoc* plumes, rather than known and predictable outflows. Thus the majority of contaminants in *NWS-InVitro* are represented as CSurfaces. These CSurfaces are read in from pre-generated data files (Fulton et al. 2006b) based on the same time series that is used to drive the tides and wind SCSurfaces (chapter 1). The areal extent of each contaminant plume agent is limited to the region in which they can be expected to exert a significant influence.

Regardless of the type of contaminant agent used (Tracer or CSurface), the basic representation is as a spatially distributed relative concentration field which (usually) decays with distance from a peak value of one at the source down to zero at its greatest extent. The value at a specific location then is the proportion of the source remaining by the time the plume has reached that spot. This proportion is then scaled by the source concentration to give the final contaminant level. The source concentrations are loaded from a time series file. These values are either concentrations or masses (if time series of flows are also associated with the outfall). The final concentration ($C_{,c,t}$) at a point in the plume at time t is given by equation 10.1.

Thus there are two or three components to the calculation of the concentration of a contaminant at some location at a given time:

$$C_{c,t} = \begin{cases} \dfrac{p_{c,t} \cdot C_{source,m,t}}{\mathbf{F}} & \mathbf{F} > 0 \\ p_{c,t} C_{source,t} & \text{otherwise} \end{cases} \qquad (10.1)$$

with $p_{c,t}$ the location specific plume strength (~[0,1]); **F** is the flow from the source (from a time series file); $C_{source,t}$ is the concentration at the source (from a time series) and $C_{source,m,t}$ is the mass at the source if using the flow calculation rather than simply reading in a source concentration.

10.1 Interactions between contaminants and other agents

Nearly all agent types can potentially interact with contaminant plumes, though currently it is only Animal, Thing, Tracer and Polyorganism agents that do so. Population agents are do not currently interact with contaminants due to issues regarding the application of toxicant effects and the subsequent tracking of the proportion of the metapopulation that has been impacted on what is ostensibly a heterogeneous group – the dimensionality of this problem quickly increases taking it beyond the current resources (work on alternative formulations may alleviate the problem in the future).

For the interaction to occur the agent which is suseptible must contact a contaminant plume and be set as “interested in contaminants”. If this is the case then the level of the contaminant contacted and the duration of contact is recorded.

Contact with a plume can either be via movement of a motile agent into a plume or the advection of the plume over the location of a sedentary agent (e.g. an oyster lease). Once in contact with the plume, contamination is through absorption or consumption. The subsequent transfer and bioaccumulation of contaminants (specifically toxins) through trophic interactions has not yet been implemented. This means that the full extent of potential toxicant effects are likely to have been underestimated. This is less of an issue on the North West Shelf given the spatial extent and types of contaminants under consideration. In places that are more densely populated and impacted (such as south-east Australia) a trophic accumulation representation will need to be developed.

10.1.1 Contaminant uptake

At present there are the three modes for uptake of a contaminant once a plume has been contacted.

1. linear uptake (parameter is in parts/sec):

$$C_{a,t} = C_{a,t-dt} + dt \cdot \tau_l \cdot C_{c,t} \tag{10.2}$$

   where $C_{a,t}$ is the concentration of the contaminant in an agent at time t; $C_{c,t}$ is the concentration in the water column (or sediments) and τ is the species specific contaminant uptake rate.

2. sigmoidal uptake – this is ideal for representing interactions with a toxin that has no effect until a critical concentration is reached:

$$C_{a,t} = \frac{C_{c,t}}{C_{a,t-dt} + \left(C_{c,t} - C_{a,t-dt}\right) \cdot \exp\left(-\tau_s \cdot dt\right)} \tag{10.3}$$

3. piecewise-linear – currently only two segments, but could be extended to n-segments. This particular representation allows for the approximation of most concentration and uptake profiles:

$$C_{a,t} = C_{a,t-dt} + \alpha_{c,1} \cdot \left(C_{c,thresh} - C_{c,t}\right) \cdot dt + C_{a,t-dt} + \alpha_{c,2} \cdot \left(C_{c,t} - C_{c,thesh}\right) \cdot dt \tag{10.4}$$

   where $C_{c,thresh}$ is the transition point between the segments; and $\alpha_{c,i}$ is the slope of segment i.

The uptake curves for prawns were calibrated from data in Hashmi et al. (2002), however there was insufficient data to test the predictive strength of the data.

10.1.2 Mortality due to contaminant poisoning

Contaminants may have several biological effects. Acute exposure may cause death without substantially altering the tissue concentrations in the body of the impacted organism, while chronic exposure may gradually raise the tissue load to a lethal level. Alternatively, sublethal effects may occur, which affect the behaviour or reproductive capacity of the organism. Unfortunately the data on the sublethal effects of contaminants on the species included in *NWS-InVitro* are not currently adequate to allow for the expression of sublethal effects in the model. Consequently, the remainder of this chapter will concentrate on the representation of toxicant induced mortality (Haas et al. 1997).

For a single contaminant the LC50 (the *concentration* at which 50% of the population is killed in a given time) can be simulated via an exponential or linear decay function. Complications arise when there is more than a single contaminant (as typical in *NWS-InVitro*).

To illustrate the problem, suppose there is a fish school that is subjected to a toxic plume that ought to kill 50% of the school. Now suppose that another toxicant is present in the plume that also ought to kill 50%. How much of the school remains alive? The simple intersection of these mortality sets range from a complete overlap (meaning half the school escapes unscathed) to a complete mismatch (meaning the entire school is poisoned by one or other of the toxins). Inappropriately applied mortality may lead to systematic under (or over) estimation of the total mortality. The issue is properly addressed by constructing a multivariate co-mortality surface that maps the exposures of all the contaminants to some LC-centile. Unfortunately, the data necessary for this is not available and so *NWS-InVitro* approximates the mortality due to each of the contaminants in a plume by assuming that their mortalities are independent of each other.

Mortality from multiple contaminants in *NWS-InVitro* is dealt with by representing the impacted agent (e.g. school) as a unit *n*-cube with *n*+1 dimensions (where there are *n* contaminants). Each axis is associated with one of the n contaminants or "other mortality". The extent of the resulting *n*-cube along that axis represents the proportion of members of the initial agent which have (or would have) died as a result of that source of mortality. To record this in a simulation a vector $\mathbf{m_v}$, (of the same dimensionality as the *n*-cube) is maintained which has these mortality proportions as its ordinates. The vector is initially set to the origin and is constrained to move away from the origin monotonically in each ordinate (regardless of the source of the mortality, an agent cannot resurrect dead members). Once $\mathbf{m_v}$ moves away from the origin it induces two smaller *n*-cubes: **B** (Both impact) and **S** (Surviving) (red and blue respectively in figure 10.1.1).

The initial *n*-cube had a unit volume and was identified with the entire agent membership. By multiplying the volume of **S** by the initial size of the membership we arrive at the number remaining. The volume of **B** is similarly related to the number of members of the agent that have been killed by all of the potential causes of mortality. Note that the independence of the action of the contaminants follows naturally from the independence of the axes used to represent them. While this approach was devised to deal with the effects of poisoning, it can apply equally well to any situation where the agent's mortality from multiple independent sources needs to be tracked.

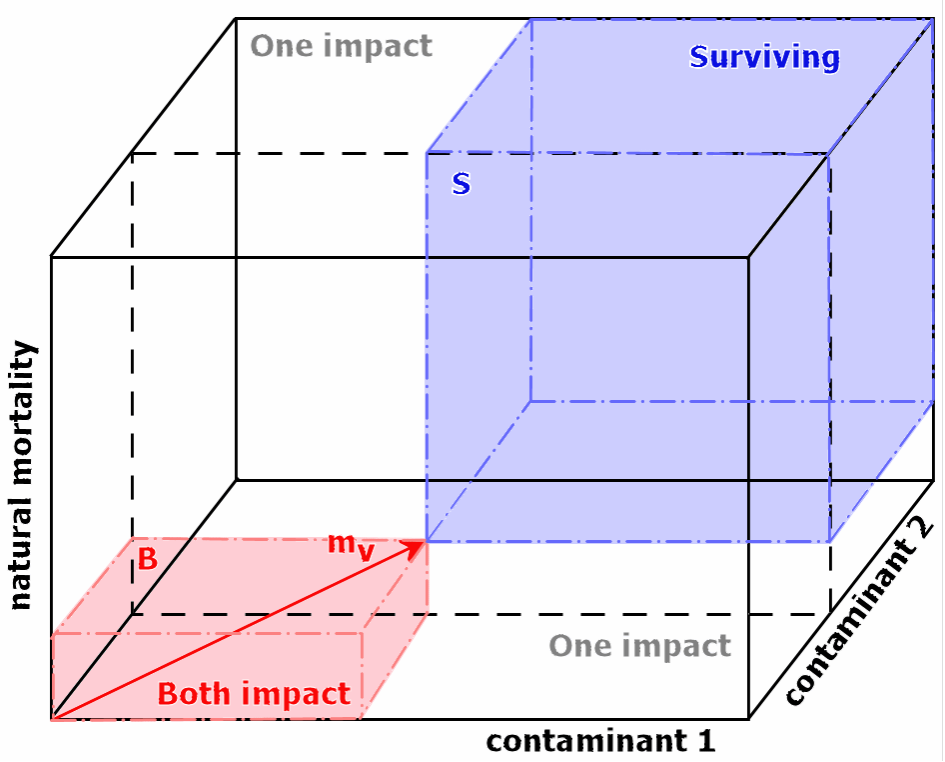

Figure 10.1.1: The 3-cube representation used for a system with two contaminants. The volume of the blue *n*-cube S is proportional to the population remaining, the volume of the red *n*-cube B is proportional to the population killed by both contaminants.

10.2 Contaminant decay

Once in the water column or sediments, contaminants (toxins) are reduced according to a standard half-life representation, with the rate of decay (λ_d) given by:

$$\lambda_d = -\frac{\ln(0.5)}{T_{half-life}} \tag{10.5}$$

where $T_{half\text{-}life}$ is the half life of the contaminant.

11. POPULATION BIOMASS AND FISH BIOMASS AGENTS

The Population Biomass (PopBiomass) and Fish Biomass (FishBiomass) agents are both types of Monitor agent. They have three purposes:

1. they are responsible for reporting individual Population and Fish agent characteristics (e.g. spawning biomass, and catch taken);
2. they aggregate values for specific characteristics across all agents of a particular species (or taxon); and
3. they store historical catch and effort data and apply these catches as historical fishing mortality to the Population and Fish agents.

The final role is the most extensive. It involves transforming the catch and effort data at a 0.5 by 0.5 degree spatial scale to the scale of the individual Fish and Population agents so that the mortality imposed is of the right order. Similarly they transform the historical data on to the appropriate scale to match the fishing zones for the Fisheries Management Authority agent so it can be used in the assessments (which are done at the scale of the zones).

11.1 Historical fishing mortalities

The historical fishing mortalities to be imposed on the Population and Fish agents is determined by solving the catch equation (11.1) for $F_{i,y}$:

$$C_y^i = \sum_a w_a V_a F_{i,y} N_{i,a,y} \frac{\left(1-\exp\left(Z_{a,y}^i\right)\right)}{Z_{a,y}^i} \tag{11.1}$$

$$Z_{a,y}^i = M + V_a F_{i,y} \tag{11.2}$$

where w_a is the nominal weight at age a, V_a is the selectivity of age a, $F_{i,y}$ is the harvest rate in area i, in year y, etc:

$$C_y^i = \tilde{C}_y^g k \exp(-\alpha_i D_{i,j}) \tag{11.3}$$

and $\tilde{C}_y^j$ is a datum of catch at location j in year y; α_i is the localised fishing coefficient; $D_{i,j}$ is the distance between the location of Population agent i, and that catch datum location j; and k is a scaling constant such that:

$$k = \frac{1}{\sum_i \exp\left(-\alpha_i \cdot D_{i,j}\right)} \tag{11.4}$$

The PopBiomass agent then goes further, calculating the catchability coefficient for the relevant Population agents, using:

$$q_i = \exp\left[\frac{\sum_y \ln(F_{i,y} / \bar{E}_y^i)}{n_y}\right] \tag{11.5}$$

where the average historical effort expended in year y scaled to the location of agent i ($\overline{E}_y^i$) is given by:

$$\overline{E}_y^i = \frac{k \cdot \exp\left(-\alpha_i \cdot D_{i,j}\right)}{n_j \sum_j E_{y,j}} \tag{11.6}$$

These catchabilities are then used by the Boat agents (see chapter 14) in the projection period of the simulation to generate the realised fishing mortalities.

12. VESSEL AGENTS

Vessel agents are a subclass of Thing agent types. In *NWS-InVitro* they are used to simulate the freighter and tanker traffic that traverse the marine environment in the Pilbara.

12.1 Movement

Vessel agents move in the same way as all other Thing agents, using an equation of the same form as (2.1) and (2.3). In place of habitat gradients however, are a series of waypoints the vessels use as a route guide.

The Vessel agents do not have any other behaviour and basically simply move from one port or fixture (e.g. Oil Platform) agent to another according to a specified schedule. The critical feature of this movement operation is the monitoring of the future path and the ability to respond to potential collisions with other Vessel agents.

12.1.1 Evasions

When a vessel is active, if it detects another vessel on a course that intersects its own then it attempts to evade turning away from its current path (opening the distance between the two vessel tracks). When the vessel judges the evasion successful (i.e. the tracks no longer intersect) then it returns to its original course (see an example track in figure 12.1.1). These evasion actions have the same associated fuel costs and bathymetry checks as any standard course change.

Even if two vessels could not successfully evade in reality, actual collisions are not simulated in *NWS-InVitro*. This is because the cost of calculating *ad hoc* plumes proved computationally prohibitive on the scale of the North West Shelf regional ecosystem. Consequently, the evasions are recorded so that the risk of collisions (and associated catastrophic impacts) can be evaluated even if the actual collisions are not played out.

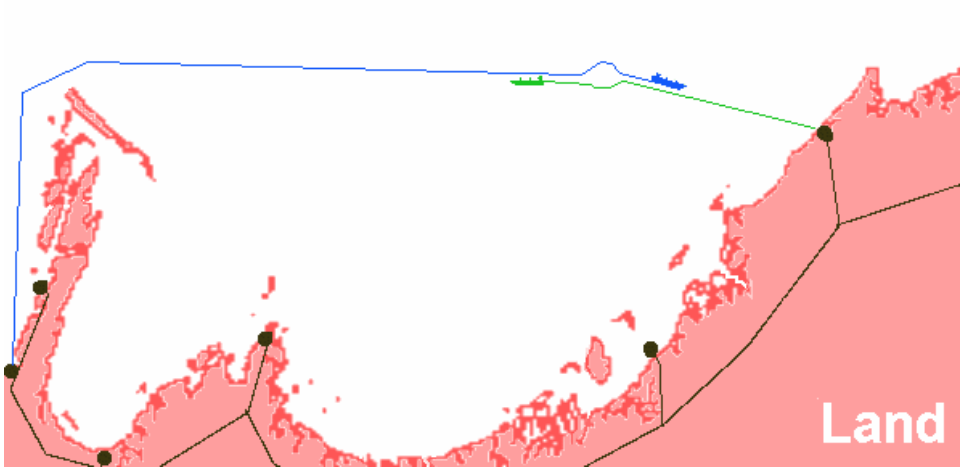

Figure 12.1.1: Example evasion vessel tracks.

13. PORT AND FIXTURE AGENTS

13.1 Fixtures

Fixture agents are a type of Thing agent. They do not have many specific features beyond being locations or markers that are used as destinations or navigation beacons by other agents. These agents may be motile, but are more typically held at fixed locations. In *NWS-InVitro* Fixture agents typically represent travel route waypoints, markers, buoys, and the sources of contaminant plumes.

13.2 Ports

Port (and rig) agents are a subclass of Fixture agents. They are the ultimate destination of vessels, and are defined based on their role for the Vessel and Boat agents. They can also be classified into "ports existing in the modelled region" (Pilbara coast in this North West Shelf case), and "ports existing elsewhere". This latter class of ports is nothing more complicated than points lying along the boundary of the modelled region, and are necessary for the vessel agents to move towards the edge of the modelled area – thus simulating movement to and from areas beyond the model domain. In *NWS-InVitro* these ports include: Mumbai, Calcutta, Osaka and US. Their locations were chosen to signify potential directions that vessel traffic would take as they left the modelled region; and they cover the general directions going west, north-west, north and north-east (figure 13.2.1). Table 13.2.1 lists the ports included in *NWS-InVitro*, and their classifications.

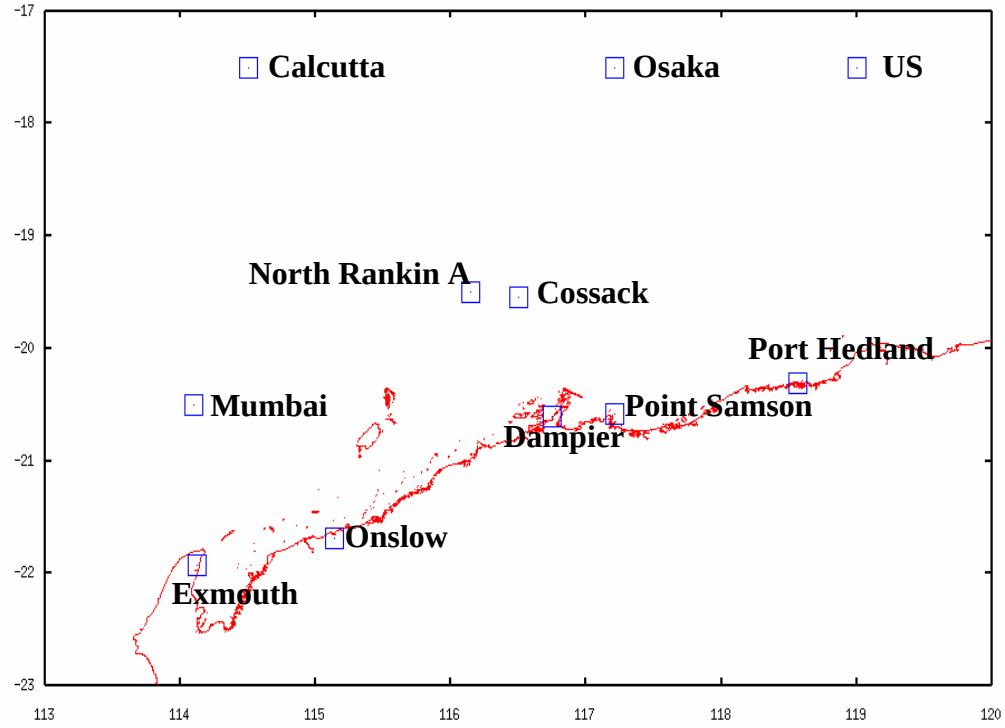

Figure 13.2.1: Locations of port agents in the North West Shelf region.

Table 13.2.1: List of Port agents.

Name	Notes
Exmouth	major prawn fishing port
Point Samson	major fish trawling port
Port Hedland	major fish trawling port
	major destination for traffic
Dampier	major fish trawling port
	major prawn fishing port
	major destination for traffic
Onslow	major prawn fishing port
North Rankin A	major destination for traffic
Cossack	major destination for traffic
Mumbai	major destination for traffic
US	major destination for traffic
Calcutta	major destination for traffic
Osaka	major destination for traffic

14. BOAT AGENTS

Boat agents are a type of Vessel agent. They are responsible for fishing either by trawling (finfish or prawns) or trapping (finfish). Since Fish and Population agents are constrained by fishing mortalities generated by catch data in the historical period of the simulation, Boat agents are only operational during the projection period. When they are active they display a highly complex behaviour, in an attempt to capture the fishing patterns and behaviour of real fishers. At its core, this behaviour is driven by the decision model outlined in figure 14.1.

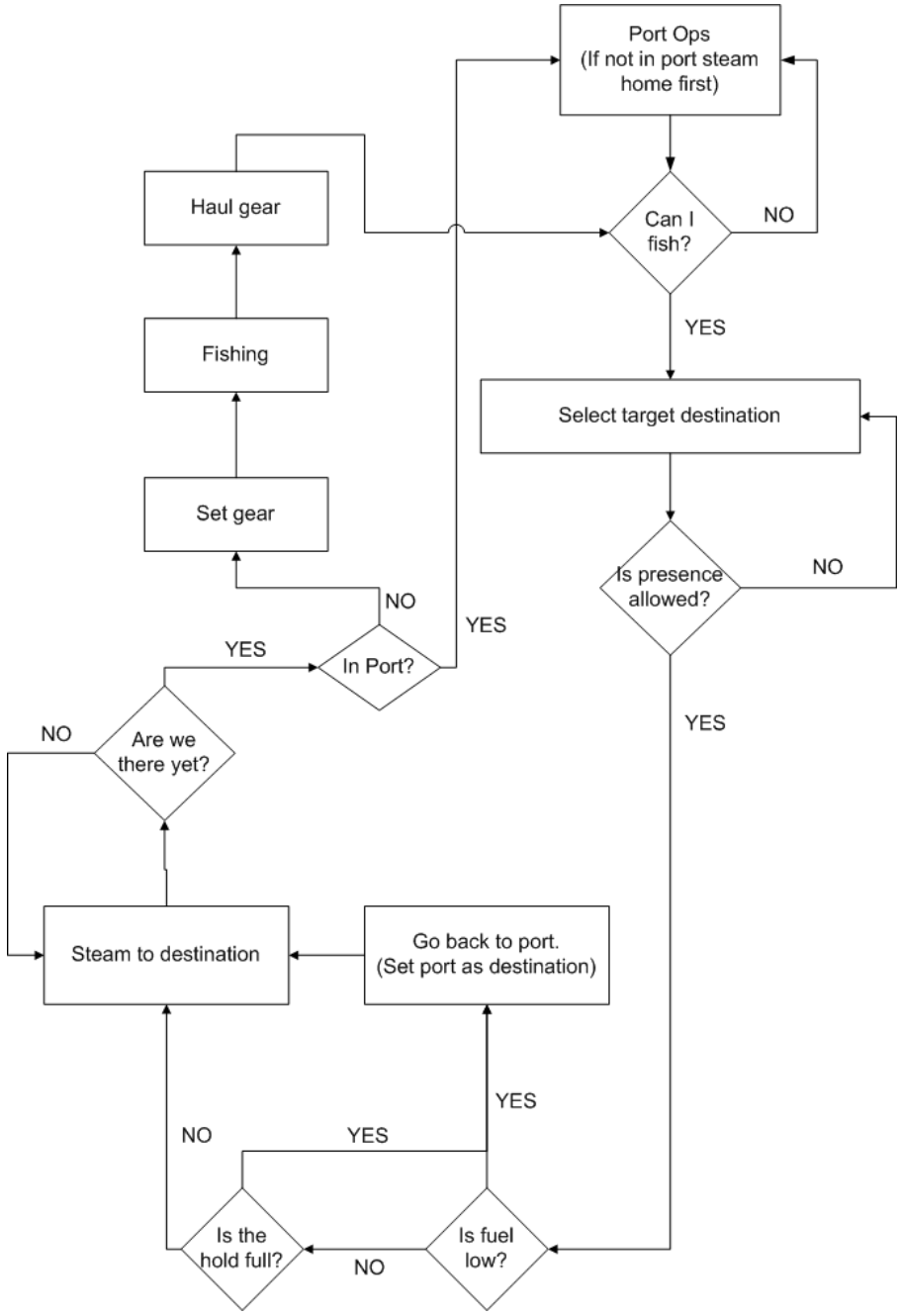

Figure 14.1: Diagrammatic representation of the fishing boat behaviour.

14.1 Selecting a target location

The Bayesian decision making process used by each fishing vessel to spatially allocate their fishing effort was based on each fishing vessel maintaining an internal representation of the spatial distribution of the state of the fish resource. The spatial resolution of the resource which was used in selecting where to allocate trawl shots and trap locations was one degree grid squares covering the North West Shelf. Within the one degree grid cell that was deemed to have the highest expected catch rate, trawl shot and trap placement locations and vectors were selected randomly from a record of previous trawl shots that started in the grid cell. This record of the state of fish resources in each cell was maintained and updated using a Kalman filter. The representation of the resource dynamics internal and specific to the fishing vessel (the prior distribution) was updated (using the numerical likelihood) on each occasion that the grid cell was fished.

The choice of fishing location by a fishing boat is based on the past experience of catch rates of each vessel for the different species. This information is maintained and updated as a Kalman filter as:

$$\hat{X}_{v,g,s,t+d} = G^d \cdot X_{v,g,s,t} \tag{14.1}$$

$$\hat{P}_{v,g,s,t+d} = G^{2d} \cdot P_{v,g,s,t} + d \cdot Q \tag{14.2}$$

where $X_{v,g,s,t}$ is the expected CPUE of species s, in grid cell g, at time t of fishing boat v; $\hat{X}_{v,g,s,t+d}$ is the predicted CPUE of species s, in grid cell g, at time t of fishing boat v after d days; $P_{v,g,s,t}$ is the expected CPUE error of species s, in grid cell g, at time t of fishing boat v; $\hat{P}_{v,g,s,t+d}$ is the predicted CPUE error of species s, in grid cell g, at time t of fishing boat v after d days; G is the transition or growth rate of the species assumed by the fishing boat (assumed to be 1); Q is the variability of the species assumed by the fishing boat (assumed to be 1). These assumed values were selected by tuning so that the final result replicates a de-trended random walk.

The correction step of the Kalman filter, based on a measurement in the form of a fishing sessions is:

$$X_{v,g,s,t+d} = \hat{X}_{v,g,s,t} + k(CPUE_{v,g,s,t+d} - \hat{X}_{v,g,s,t}) \tag{14.3}$$

$$P_{v,g,s,t+d} = \hat{P}_{v,g,s,t}(1-k) \tag{14.4}$$

where

$$k = \frac{\hat{P}_{v,g,s,t}}{\left(\hat{P}_{v,g,s,t} + H\right)} \tag{14.5}$$

and H is the variability of the CPUE assumed by the fishing boat (assumed to be 1).

In making a decision of where to fish (which 0.5 by 0.5 dregree grid cell) a weighted average across species of the $X_{v,g,s,t}$ is calculated for each fishing boat v at the current time t. Species weights are based on the species preferences of the individual boat. The grid cells are ranked according to the weighted average, and a trawl (or trap) location is selected at random from all previously logged trawl (or trap) locations logged to that

grid cell. If no trawl (or trap) locations are logged in that grid cell, a location is chosen randomly within it until a maximum of 50 trawl (trap locales) paths are logged for that grid cell and vessel.

The initial behaviour of each vessel prior to the start of the projection period is based on voluntary log book data of different vessels. These data initially consist of historical trawl locations, the catch rates of the different species group, and the 0.5 degree grid reference. As the boat fishes, these data are updated with the locations, catch rates and grid references of its simulated operations.

15. PLANE AND TRAP AGENTS

Plane and Trap agents are simple forms of Monitor and Thing agents (respectively) that have a supporting role in the fisheries activities of the Boat agents. They represent technologies, gear or behaviours that are crucial parts of the fishery that do not take place on the boats themselves. While traps are used by boats they are left in situ and have their own potential "experience track" (e.g. a cyclone may strike a boat but leave the traps in place) and as such must be treated as a separate agent. The distinct nature of the planes is simpler to appreciate. They fly independently from boats and simply update the boat-level knowledge base (and only for those boats that are marked as using spotter plane information, which is not a universal practice).

15.1 Plane agents

Plane agents are a form of Monitor agent, which polls the agents under its specified search track. Anything within its visual range (which is impacted by visibility and water clarity) is recorded and these records are used to update the Kalman filter for any fishing boat that is flagged as using spotter planes.

15.2 Trap agents

Trap agents are a form of the Thing agent type. They represent fish traps and are deployed by Boat agents during fishing operations. They are simple agents with a short set of characteristics to do with their location and fishing characteristics:

- ownership;
- size;
- soak time;
- area of attraction; and
- species specific impacts or interaction strengths (i.e. what species can be captured by the trap).

16. RECFISHER AGENTS

The Recfisher agent is a form of Monitor agent. This agent represents the recreational fishing pressure imposed by the local human population on the Population and Animal agents. Given the size of the human population and the scarcity of data on the topic, it was decided to represent this sub-model by applying a simple tithe based on the population size, rather than a more individual-based decision model.

16.1 Mortality due to recreational fishing

The number of members of a non-population Animal agent (typically fish) removed by recreational fishing (M_{rec}) is given by:

$$M_{rec} = \frac{N_f \cdot m_{rec} \cdot p_{rec} \cdot N_{human,p} \cdot k_{impact}}{d_{ramp} \cdot d_p + 1} \tag{16.1}$$

where m_{rec} is the tithe due to a single person fishing recreationally; p_{rec} is the proportion of the human population that fishes recreationally; N_f is the number of members in the Animal agent being fished; $N_{human,p}$ is the size of the human population in the population centre p; d_{ramp} is the distance to the boat ramp (current access point) in kilometres; d_p is the distance from the access point to the human population centre; and k_{impact} is the proportion of those the gear interacts with that are killed (incidentally or caught and discarded) or removed from the system (landed), which is set to one if of legal size otherwise it is set at to a user defined value.

If a Population agent is interacting with a RecFisher agent then the members removed is given by:

$$M_{rec} = \sum_a \frac{F_{rec,a} \cdot p_a \cdot \left(1 - \exp\left(-Z_a\right)\right)}{Z_a} \tag{16.2}$$

with p_a is the proportion of the population in age class a; Z_a is total mortality for age class a as defined by:

$$Z_a = F_{rec,a} + \frac{M \cdot dt}{s_{yr}} \tag{16.3}$$

where M is the annual natural mortality; s_{yr} is the number of seconds in one year; and $F_{rec,a}$ the fishing mortality due to recreational fishing pressure on population age class a given by:

$$F_{rec,a} = \left(V_{m,a} \cdot p_m + V_{f,a} \cdot \left(1 - p_m\right)\right) \cdot k_{impact} \cdot N_{human,p} \cdot p_{rec} \cdot \frac{dt}{s_{yr}} \cdot \overline{q} \cdot p_q \cdot \exp\left(k \cdot \left(d_{ramp} + d_p\right)\right) \tag{16.4}$$

with V the vulnerability of the population to the gear (i.e. selectivity); $\overline{q}$ the commercial catchability; and p_q is a scalar on the commercial catchability to give the recreational

catchability. It should be noted that for “individual-based” representations, the availability of a fish or school is explicitly determined by the nature of its location: that is whether or not it is in suitable depths and in unprotected areas. For population agents, the effect of protected areas on the availability of the fish is factored into this vulnerability which is based on the commercial fisheries.

16.2 Management of recreational fishing

The management procedures included in this agent include rules on discarding rates, legal size limits, prohibited species and spatial management zones. A fish can only be killed and landed by the RecFisher agent if all of these management measures allow it (i.e. it is a species that is both desirable and unprotected, of a desirable and legal size and not in a “no-take” marine protected area). Incidental mortality can also be applied to an agent that interacts with the RecFisher agent (see above).

17. FISHERIES MANAGEMENT AUTHORITY (FMA) AGENTS

The FMA agent is a form of management agent. It assumes that the fisheries being modelled are managed using a combination of reference points and gear, effort or spatial and temporal closures. The typical form of the FMA agent uses a fisheries stock assessment model and the decision procedure that uses them.

17.1 Management and assessment of scalefish

The assessment model implemented in *NWS-InVitro* is the age-structured model used by WA Fisheries to estimate the status of three commercially targeted species: *Lutjanus sebae*, *Epinephelus multinotatus*, and *Lethrinus hutchinsi*. The WA Fisheries stock assessment model is replicated in the FMA agent and is applied to *Lutjanus sebae*.

Basic dynamics

$$N^{s,\gamma}_{a+1,y+1} = \begin{cases} N^{s,\gamma}_{0,y} & \text{for } a = 0 \\ N^{s,\gamma}_{a,y} \exp(-(M + V^s_a F^\gamma_y)) & \text{otherwise} \end{cases} \tag{17.1}$$

where $N^{s,\gamma}_{a,y}$ is the number of fish in year y, management area γ, of age a, sex s; M is natural mortality; V^s_a is the selectivity of the fishing gear or vulnerability of animals age a, sex s, such that:

$$V^s_a = \left(1 + \exp\left[-\ln(3)\left(\frac{a - \nu_{0.5}}{\nu_{0.5} - \nu_{0.25}}\right)\right]\right)^{-1} \tag{17.2}$$

where "ln(3)" falls out of the solution for the logistic function given the use of lengths at 25% and 50% selectivity.

Recruitment to fishery

A Beverton-Holt stock recruitment relationship is used to determine the recruitment of 0 year-olds to the fishery. Specifically:

$$R_y = \frac{S_y}{\alpha + \beta S_y} \tag{17.3}$$

where

$$S_y = \sum_{\gamma=0}^{6} S^\gamma_y \tag{17.4}$$

$$\alpha = \frac{S_1}{R_1}\left(1 - \frac{(h - 0.2)}{0.8 \cdot h}\right) \tag{17.5}$$

$$\beta = \frac{(h - 0.2)}{0.8 \cdot h \cdot R_1} \tag{17.6}$$

with the 0.2 and 0.8 values from WA Fisheries (P. Stephenson pers. comm.) and with *h* is the parameter termed the steepness parameter; and the spawning biomass in management area γ in year *y* (S^{γ}_{y}) is given by:

$$S^{\gamma}_{v} = \sum_{a=0} N^{f,\gamma}_{a,y} \cdot p_a \cdot w^{f}_{a} \tag{17.7}$$

with p_a the proportion of sexually mature females of age *a* calculated using:

$$p_a = \left(1 + \exp\left[-\ln(19)\left(\frac{a - a_{0.5}}{a_{0.95} - a_{0.5}}\right)\right]\right)^{-1} \tag{17.8}$$

and w^{s}_{a} is the weight of animal sex *s*, age *a* is found as follows:

$$w^{s}_{a} = \alpha^{s} \exp\left(\beta^{s} \cdot l^{s}_{a}\right) \tag{17.9}$$

where α^{s} and β^{s} are the sex specific allometric parameters and l^{s}_{a} is the length at age *a* and sex *s* given by:

$$l^{s}_{a} = l^{s}_{\infty} \cdot \left(1 - \exp\left(l^{s}_{\lambda} \cdot \left(a - t^{s}_{0}\right)\right)\right) \tag{17.10}$$

with l^{s}_{a}, l^{s}_{∞} and t^{s}_{0} are the sex specific growth parameters, and A is the maximum age of fish, and it is assumed that no fish grow older than this age.

The recruitment in each management area of the fishery in year *y* is given by:

$$R^{\gamma}_{y} = \begin{cases} \dfrac{R^{\gamma}_{1}}{R_1} R_y & \text{for y} \le 2000 \\ \dfrac{R^{\gamma}_{1} e^{c_r N(0,1) - c_r^2/2}}{R_1} R_y & \text{otherwise} \end{cases} \tag{17.11}$$

where c_{γ} is a value (coefficient of variation) specified to allow random variability in $R_1{}^{\gamma}$ for the purpose of projecting the model into the future.

These parameters determine:

$$N^{s,\gamma}_{0,y} \tag{17.12}$$

with

$$N^{f,\gamma}_{0,y} = \rho R^{\gamma}_{y} \tag{17.13}$$

$$N^{m,\gamma}_{0,y} = (1 - \rho) R^{\gamma}_{y} \tag{17.14}$$

and ρ is the proportion of animals that are female.

Initial conditions

The initial conditions in the first year of the model are controlled by the parameters R_1^γ. These determine R_1 and S_1 where

$$R_1 = \sum_\gamma R_1^\gamma \tag{17.15}$$

and

$$S_1 = \sum_\gamma S_1^\gamma \tag{17.16}$$

Fishing mortality

Fishing mortality, as an annual harvest rate, is calculated using:

$$F_y^\gamma = \begin{cases} \dfrac{\sum_\delta C_y^{\gamma,\delta}}{B_y^\gamma} & \text{for } 1972 \le y \le 2000 \\ q_y^\gamma E_y^\gamma T_y & \text{otherwise} \end{cases} \tag{17.17}$$

where F^γ_y is the fishing mortality in year y in area γ; $C_y^{\gamma,\delta}$ is the catch by sector δ (trawl or trap) in year y, area γ; E^γ_y is the trawl effort (quota) projected into the future; T_v is a measure of effort (technology) creep; q^γ_y is the catchability coefficient that incorporates random variability such that:

$$q_y^\gamma = \overline{q}^\gamma e^{\sigma^2_{q,\gamma}(N(0,1)-\sigma^2_{q,\gamma}/2)} \tag{17.18}$$

with

$$\overline{q}^\gamma = e^{\sum_y \ln(F_y^\gamma \frac{U_y^\gamma}{\sum_\delta C_y^{\gamma,\delta}})/n_y} \tag{17.19}$$

$$\sigma^2_{q,\gamma} = \left(\frac{\sum_y \left(\ln\left(\frac{F_y^\gamma \cdot U_y^\gamma}{\sum_\delta C_y^{\gamma,\delta}} \right) \right)}{\overline{q}^\gamma} \right)^2 \tag{17.20}$$

and U^γ_y is the observed historical catch rate for the trawl sector, and n_v is the number of year for which there are data. Note therefore that the term given below denotes the equivalent trawl effort of the total cross sectoral catch.

$$\frac{U_y^\gamma}{\sum_\delta C_y^{\gamma,\delta}} \tag{17.21}$$

Fitting the model

The model is fitted by maximising the log-likelihood of the CPUE. Namely:

$$\lambda_{CPUE} = \frac{n_1}{2} \ln\left(\frac{2\pi}{n} \sum_{y} (U_y^{\gamma} - \hat{U}_y^{\gamma})^2 \right) \tag{17.22}$$

where n1 is the number of years of commercial catch data; n is the number of years over which the sum of squares are calculated and:

$$\hat{U}_y^{\gamma} = \overline{q}^{\gamma} B_y^{\gamma} \tag{17.23}$$

$$B_y^{\gamma} = \sum_{a=0}^{A} V_a^{s=f} N_{a,y}^{f,\gamma} p_a w_a^f + V_a^{s=m} N_{a,y}^{m,\gamma} (1 - p_a) w_a^m \tag{17.24}$$

The free parameters used to fit the model are the steepness h and the area-specific initial recruitment levels R_1^{γ}.

Projecting the model

Once the model has been fitted to the historical catch rate data, it is projected into the future under the estimated model. Two primary sources of uncertainty are incorporated into the projections. The first is the uncertainty in the catchability coefficient q^{γ}_v, and the second is uncertainty in the area specific, initial recruitment parameters R_1^{γ}. The final coefficient used in projecting the stock assessment model is the effort creep parameter T_v, which can incorporate trends in the catchability.

17.1.1 Example stock assessment

To illustrate the FMA agent, the stock assessment is applied to two model specifications (low and medium biomass levels), and two management strategies (with and without fisheries independent data) (For details on what these strategies are and what parameterisations are used for the model specifications in these cases see Fulton et al. (2006b)). The biomass estimates were compared to the actual operating model biomass of *Lutjanus sebae* (figures 17.1.1 to 17.1.4). The first stock assessment that was performed in a simulation is in 2000. This assessment is based entirely on the historical catch and effort data and not on data generated from the fishing boat agents. The biomass estimates of these are therefore the same across model specifications and management strategies. This stock assessment, based solely on the historical data, is closer to the actual biomass under the medium model specification, than the biomass under the low specification.

Under the medium model specification, the management strategy that incorporates fisheries independent data is better able to estimate the actual biomass. In contrast, under the low model specification, the management strategy without independent survey data consistently over-estimates the actual biomass. The addition of fisheries independent data under this low biomass level does result in biomass estimates from the assessment model that are closer to the actual values in the operating model, but they are still slightly over-estimated during some years. The projected biomasses however, tend to be over-optimistic and are not reliable predictions regarding future stock biomasses.

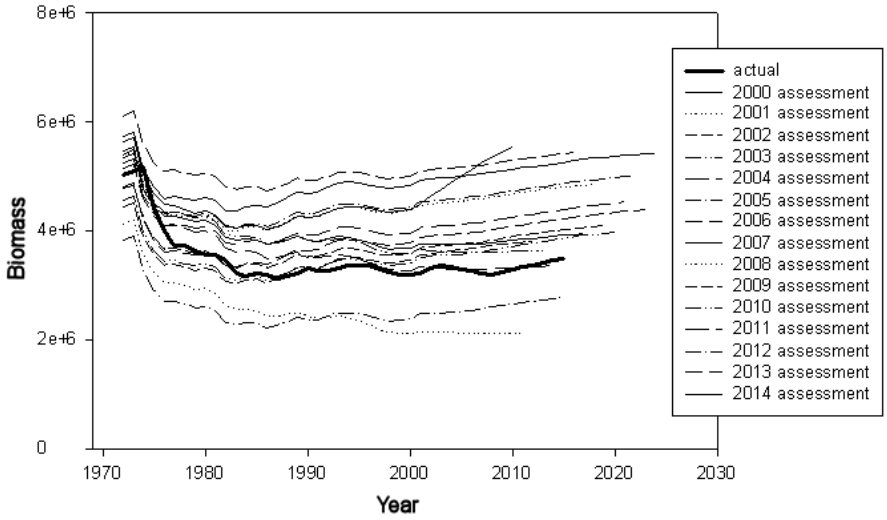

Figure 17.1.1: Biomass trajectories of *Lutjanus sebae* from the stock assessment model and the operating model for the management strategy that does not include fisheries independent data, under the medium biomass model specification.

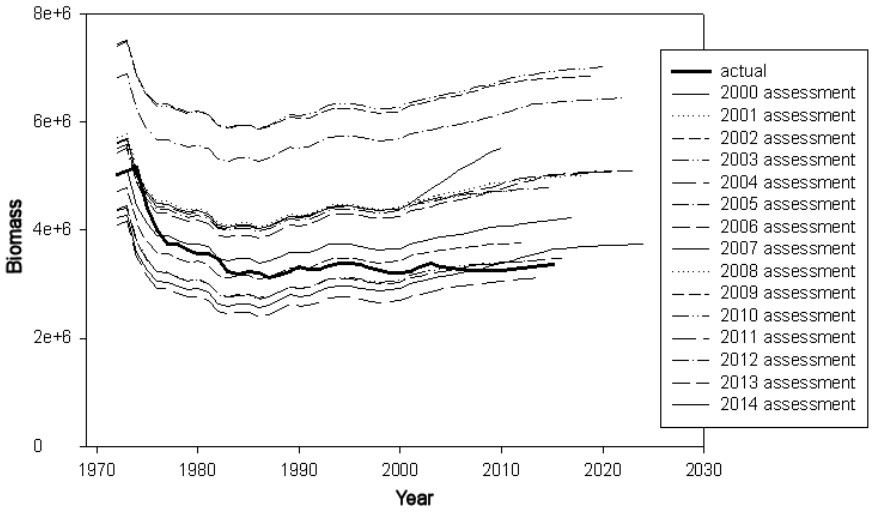

Figure 17.1.2: Biomass trajectories of *Lutjanus sebae* from the stock assessment model and the operating model for the management strategy that includes fisheries independent data, under the medium biomass model specification.

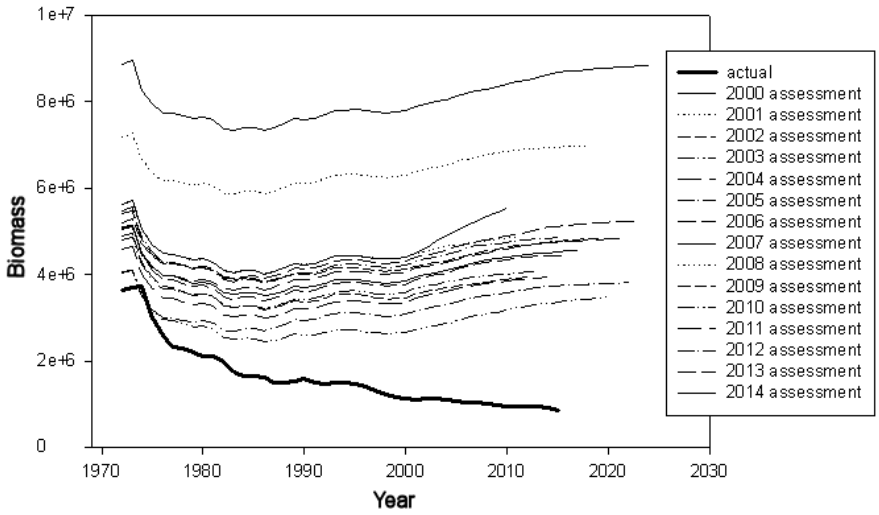

Figure 17.1.3: Biomass trajectories of *Lutjanus sebae* from the stock assessment model and the operating model for the management strategy that does not include fisheries independent data, under the low biomass model specification.

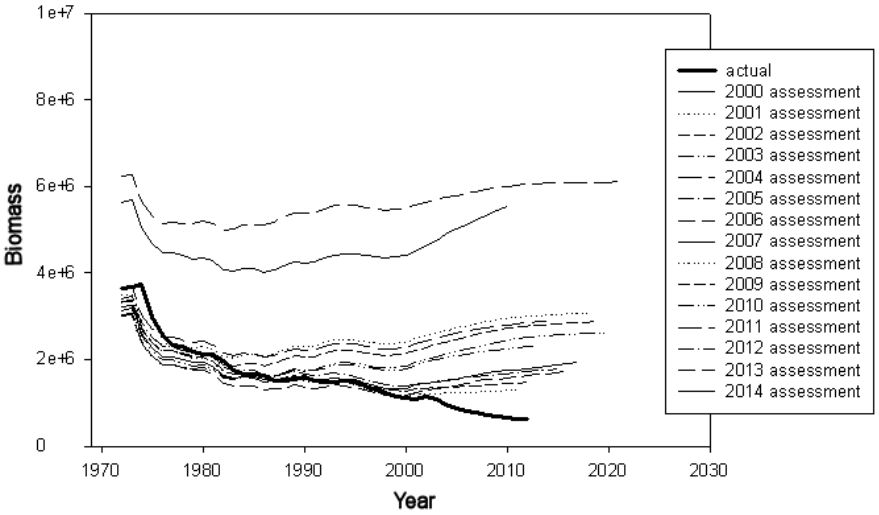

Figure 17.1.4: Biomass trajectories of *Lutjanus sebae* from the stock assessment model and the operating model for the management strategy that includes fisheries independent data, under the low biomass model specification.

17.1.2 Finfish fishery decision procedures

Armed with a fitted stock assessment model, the Fisheries Management Authority (FMA) agent of the *NWS-InVitro* model must make a decision in managing the simulated fishery. The decision procedures were based on the current decision procedural outlines (Pilbara Fish Trawl Fishery (Interim) Management Plan, 1997; Pilbara Trap Management Plan, 1992; P. Stephenson, pers. comm.). This entails projecting the estimated stock assessment model a large number of times (typically 1000 times) under the current management arrangements (i.e. current area closures and area effort quotas), and generating a distribution of spawning biomass representing the uncertainty associated with the fishery. Based on current actual management decision procedures, a management review is triggered when either:

$$P_{\gamma}(X < 0.3) < 0.25 \text{ or } P_{\gamma=all}(X < 0.4) < 0.5 \tag{17.25}$$

where $P_{\gamma}(X)$ is the distribution of relative spawning biomasses with:

$$X = \frac{S^{\gamma}_{final}}{S^{\gamma}_{initial}} \tag{17.26}$$

for γ and including all areas combined (i.e. summing over γ in numerator and denominator).

Under such conditions the simulated management review of the MSE model acts to find an effort quota that will guide the stock back above the trigger points. Reviews triggered by the latter condition indicate that the whole area is over-exploited. The response is to reduce the effort over the entire area (i.e. all management areas) by 10% until the condition is met.

In contrast, reviews triggered by the former condition indicates local over-exploitation. The simulated management response is to reduce the effort in the over-exploited area γ by 10% until the condition for this zone is met. The effort removed is then evenly allocated to the other areas, as long as it meets with condition that:

$$P_{\gamma}(X < 0.3) < 0.25 \tag{17.27}$$

Any effort that cannot be allocated to an area without it meeting the above condition is disregarded.

The inclusion of fisheries independent data

An extension of this decision rule procedure is the inclusion of fisheries independent data sources. This can be simulated by the addition of Boat agents to represent scientific fishing surveys. These boats select trawl locations randomly (first by trawl area, and then randomly within the area), subject to the same constraints as regular fishing boats (i.e. fuel reserves etc). The age distributions of the catch obtained by this boat are recorded without error, and used to improve the biomass estimates of the stock assessment model by updating the log-likelihood of the commercial CPUE data with the log-likelihood of the age distributions via:

$$\lambda_{age} = \frac{n_3}{2} \sum_{y} \sum_{s} \sum_{a} p^{s,\gamma}_{a,y} \ln\left(\frac{\hat{p}^{s,\gamma}_{a,y}}{p^{s,\gamma}_{a,y}} \right) \tag{17.28}$$

where n_3 is the number of years which the survey have been performed; $p_{a,y}^{s,\gamma}$ is the observed proportion of individuals in the catch of the survey vessel of sex s in year y that are age a; $\hat{p}_{a,y}^{s,\gamma}$ is the estimated proportion of individuals in the catch of the survey vessel of sex s in year y that are age a, i.e:

$$\hat{p}_{a,y}^{s,\gamma} = \frac{N_{a,y}^{s} V_{y}^{s}}{\sum_{s}\sum_{a} N_{a,y}^{s} V_{a}^{s}} \tag{17.29}$$

The new overall log-likelihood is then:

$$\lambda = \lambda_{CPUE} - \lambda_{age} \tag{17.30}$$

17.2 Management and assessment of prawns

NWS-InVitro can incorporate multiple prawn fisheries (e.g. three in the case of the North West Shelf: the Exmouth Gulf Prawn Fishery, the Onslow Prawn Fishery and the Nickol Bay Prawn Fishery). Each prawn boat can have multiple endorsements and be active in more than one fishery.

These fisheries can be managed based on seasonal fisheries openings, the declaration of some locations as nursery areas, and the opening and closing of other fishing areas. Reference limit points for acceptable catch levels or catch rates can also be used. If the reference limit rule is in use, the prawn FMA agent closes a randomly selected area in any fishery that displays *y* consecutive years of lower than expected total catches (below acceptable ranges). The closure then lasts for *x* years (the length of time of the current closure so far), at which time the area is re-opened. The timings used (the number of consecutive years of low catches, and the length of the subsequent closure) can be specified by the user (e.g. in the North West Shelf implementation they were two and three years respectively, which were estimated by inspection of data from previous prawn fishery closure regimes in WA, with the added constraint of a two year requirement so no single poor year could disproportionately affect the outcome). The standard management procedure uses only this catch level rule (with the catch ranges based on historical data), but it can be extended using the catch rate limits. The enhanced management strategy uses catch rate limits based on the industry-agreed voluntary catch rate limit of 16 kg/h.

17.3 Cross sector management constraints

FMA agents are one of the agent types that may be involved in cross sectoral management. If this form of management is operating then the FMA agents act normally, with the addition of extra triggers that can be influenced by the findings of other management agents. Therefore the stock assessment is not alone in determining effort quotas under the regionally coordinated management since fishing areas are (potentially) closed to fishing whenever any of the "public blackboard" management triggers are tripped. That is, effort reduction is implemented in the regionally coordinated management when (a) the target stocks or the biomass of turtles and sharks fell below limit reference points, or (b) the EPA agent puts out a notification of excessive contaminant levels within an area covered by a fishing zone (see the explanatory decision tree for regionally coordinated management in Appendix D).

18. DEPARTMENT OF TRANSPORT (DOT) AGENTS

The Department of Transport (DOT) agent monitors the arrivals and departures of transport vessels to the major ports in the modelled area (e.g. Port Hedland and Dampier). It also monitors the number of times these vessels have to invoke evasive manoeuvres in order to prevent a collision while in coastal waters, as an indicator of the likelihood of a vessel collision and possible spill.

The main management responsibility of the DOT agent is to ensure the port facilities can handle the traffic demanded by the economic conditions. This is done through an economic mechanism that attempts to maximise the profit of each port.

Port profit is defined as:

$$\pi_{p,m} = -\left(C_p - \frac{N_{p,m}}{30} \right)^2 \tag{18.1}$$

where $\pi_{p,m}$ is the profit of port p in month m; C_p is the vessel capacity of port p; and $N_{p,m}$ is the number of vessels that visited port p in month m.

A quadratic form of the port profit function was chosen to reflect the fact that profit is maximal at a specified level of use $N^*_{p,m}$ (i.e. C_p times the number of days per month). Higher usage levels will lead to over capacity usage of the port and sub-optimal profits. For reporting $\pi_{p,m}$ purposes the profit is scaled to the maximal profit level.

18.1 Decision procedure for DOT management

The management of port facilities involves monitoring the port usage $N_{p.m}$, and profit $\pi_{p,m}$. Each port has an over capacity threshold H_p of usage. If, in any given month, this threshold is exceeded, a port expansion is triggered to accommodate for the over capacity. Port expansion involves doubling the port capacity C_p, and also adding new shipping lanes (and incurring the associated environmental costs). For instance, the shipping lanes with and without port capacity expansion used in the NWSJEMS case of *NWS-InVitro* are shown in figures 18.1.1 and 18.1.2.

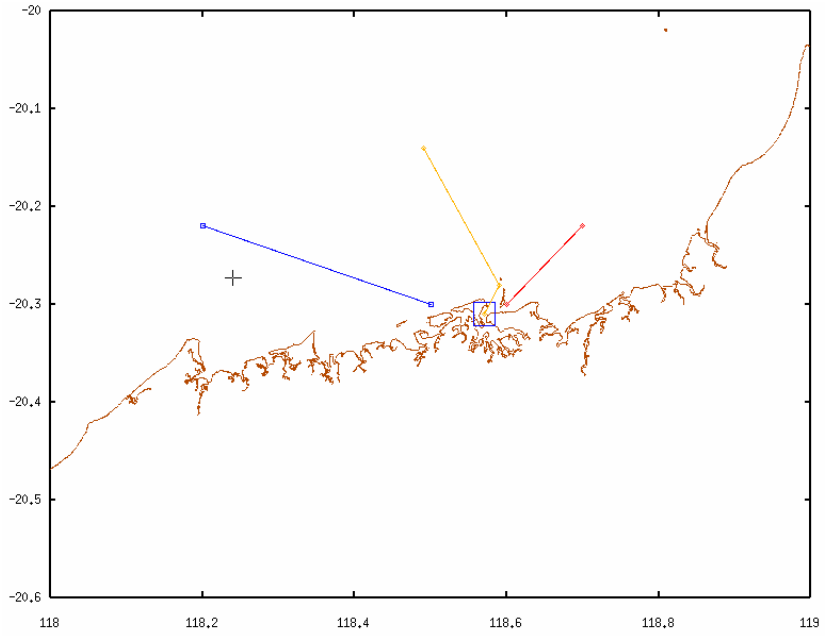

Figure 18.1.1: Map showing location of Port Hedland (square), the shipping route into and out of the port at the beginning of the projection period (yellow), and the shipping route into the port (red) and out of the port (blue) under port capacity expansion.

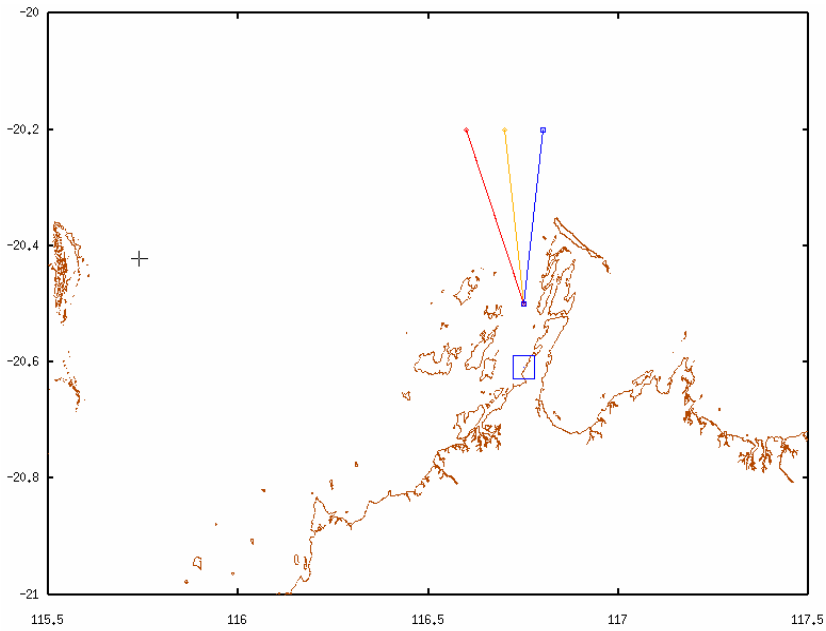

Figure 18.1.2: Map showing location of Dampier (square), the shipping route into and out of the port at the beginning of the projection period (yellow), and the shipping route into the port (red) and out of the port (blue) under port capacity expansion.

19. ENVIRONMENTAL PROTECTION AGENCY (EPA) AGENTS

The EPA agent is derived from the Monitor agent type. The EPA agent is used to control the levels of contaminants permitted at outfalls, and to trigger contaminant related management actions decreed by other management agents (such as the fisheries management [FMA] agents). It works in conjunction with several other agents, such as Biomass agents and FMA agents, to monitor various population levels. It also uses loggers (instantiations of Thing agents that log contaminant levels, see below) to monitor the contaminant levels in the water around outfalls.

Communication between the EPA agent and the other control agents is through a "public blackboard" on which they leave information about the state of the system as they see it. The other agents then respond to particular messages in their own way. In turn, the EPA agent may respond to messages indicating that various species are at low population levels by decreasing the levels of toxins allowed in the outfalls, or it may respond to an indication that population levels have recovered by reducing the level of restriction applied. Similarly, when monitored, concentration levels of a contaminant in the water exceed predetermined levels, the constraints on the contaminant are increased. Should sufficient time pass without additional infringements, the constraints may be relaxed. Any actions taken by the EPA agent are recorded in data files associated with the relevant outfall.

Control of the contaminant levels released at an outfall may be adjusted relative to the contaminant time series in two (possibly concurrent) ways:

- the mass released (based on an initial annual maximum) of the individual toxins permitted is reduced; and
- the volume of water which flows from the outfall is increased (thus reducing concentration).

19.1 Logger agents

Logger agents are forms of Thing agent and their main role is to periodically log their location and environmental conditions (primarily local contaminant levels). While this fairly passive recording role makes loggers similar in nature to fixtures, they are actually derived from Animal agents. This is to allow for greater flexibility in representation – specifically, to allow for animals to be identified as also being loggers (much like putting a satellite tag on an animal in reality).

20. OTHER AGENTS

20.1 Purity

The Purity agent is a type of Monitor agent that checks for the probity (internal consistency) of each agent. This agent type is not used in the final simulation runs of *NWS-InVitro*, but it is a useful model development tool.

20.2 DPI

The DPI agent is derived from the Monitor agent. It was originally designed to try and govern anthropogenic activities such as contaminant releases and coastal development. The first of these is now handled by the EPA agent, but in future models where more explicit coastal development processes need to be considered, this agent type will be expanded.

20.3 OilCo

This variant of the Monitor agent type was originally written to handle development of the oil and gas sector. There was insufficient information available on the processes and decisions used in this sector and so development of this agent was not completed. "Painted" scenarios were used in place of a dynamic sub-model for this part of the *NWS-InVitro* model. In the future, if more information becomes available, this agent could be usefully elaborated.

REFRENCES

Axtell, R.L., Epstein, J.M., Dean, J.S., Gumerman, G.J., Swedlund, A.C., Harburger, J., Chakravarty, S., Hammond, R., Parker, J., and Parker, M., (2002). Population growth and collapse in a Multiagent model of the Kayenta Anasazi in Long House Valley. Proc. Natl. Acad. Sci. U.S.A., 99, 7275-7279.

Barabanov, M. (1997). A Linux-based Real-Time Operating System. Masters thesis New Mexico Institute of Mining and Technology. Socorro:New Mexico. Also available at http://www.ee.ryerson.ca/~courses/ee8205/projects/RT-Linux-Report.pdf

Butterworth, D.S. and Punt, A.E., (1999). Experiences in the evaluation and implementation of management procedures. *ICES J. mar. Sci.*, 56, 985-998.

Condie, S., Andrewartha, J., Mansbridge, J., and Waring, J., (2006). *Modelling circulation and connectivity on Australia's North West Shelf. NWSJEMS Technical Report No. 6*. CSIRO, Hobart, Tasmania.

DeAngelis, D. and Gross, L. (Eds.), (1992). *Individual-Based Models and Approaches in Ecology*. New York, NY: Chapman and Hall.

Ebenhöh, W.C., Baretta-Bekker, J.G., and Baretta, J.W., (1997). The primary production module in the marine ecosystem model ERSEM II, with emphasis on the light forcing. *Journal of Sea Research*, 38, 173-194.

Epstein, J., (2002). Modelling civil violence: an agent-based approach. *Proc. Natl. Acad. Sci. U.S.A.*, 99(suppl. 3), 7243-7250.

Fulton, E.A., Hatfield, B., Althaus, F. and Sainsbury, K., (2006a). *Bentic Habitat dynamics and models on Australia's North West Shelf. NWSJEMS Technical Report No. 11*. CSIRO Marine and Atmospheric Research.

Fulton, E.A. McDonald, A.D., Hayes, D., Lyne, V., Little, L.R, Fuller, M., Condie, S., Gray, R., Scott, R., Webb, H., Hatfield, B., Martin, M., and Sainsbury, K., (2006b). *Management Strategy Evaluation Specification for Australia's North West Shelf. NWSJEMS Technical Report No. 15*. CSIRO Marine and Atmospheric Research.

Fulton, E. A., Smith, A. D. M., and Punt, A. E., (2005). Which ecological indicators can robustly detect effects of fishing? *ICES Journal of Marine Science*, 62, 540-551.

Fulton, E.A., Fuller, M., Smith, A.D.M. and Punt, A.E., (2004). *Ecological Indicators of the Ecosystem Effects of Fishing: Final Report*. Australian Fisheries Management Authority Report, R99/1546.

Grimm, V., (1999). Ten years of individual based modeling in ecology: What have we learned and what could we learn in the future? *Ecol. Model.*, 115, 129-148.

Haas, C.N., Kersten, S.P., Wright, K., Frank, M.J. and Ciambi, K. (1997) Generalization of Independent Response Model for Toxic Mixtures, *Chemosphere* **34**(4), pp 699-710.

Haddon, M. (2001). *Modelling and Quantitative Methods in Fisheries*. Chapman & Hall/CRC.

Hall, S.J., (1999). *The effects of fishing on marine ecosystems and communities*. Blackwell Science Ltd, Oxford.

Hashmi, M.I., Mustafa, S., Tariq, S.A., (2002) Heavy metal concentrations in water and tiger prawn (Penaeus monodon) from grow-out farms in Sabah, North Borneo. *Food Chemistry* **79**, pp 151-156.

Jones, H.A., (1973). Marine geology of the North-West Australia continental shelf. *Bur. Miner. Resour. Geol. Geophys. Bull.*, 136, 19pp.

Levins, R., (1969). Some demographic and genetic consequences of environmental heterogeneity for biological control. *Bull. Entomol. Soc. Amer.*, 15, 237-240.

Lyne, V., Gray R.C., Sainsbury K.J., and Scott, R. (1994). Integrated biophysical model investigations. *Research on Jarosite dumping at sea: a program of research on Jarosite dumping at sea by Pasminco Metals E-Z: Physical dispersal of Jarosite and surveys of marine biota heavy metal levels 1991-1994*, 5, 149pp.

McLoughlin, R. J. and Young, P. C., (1985). Sedimentary provinces of the fishing grounds of the North West Shelf of Australia: grain size frequency analysis of surficial sediments. *Aust. J. Mar. Freshw. Res.*, 36, 671-681.

Pilbara Fish Trawl Interim Managed Fishery Management Plan 1997, Western Australia Fish Resources Management Act 1994.

Pilbara Trap Management Plan 1992, Western Australia Fish Resources Management Act 1994.

Punt, A.E., Smith, A.D.M., and Cui, G.,(2001). Review of progress in the introduction of management strategy evaluation (MSE) approaches in Australia's South East Fishery. *Mar. Freshwater Res.*, 52, 719-726.

Sainsbury, K. J. (1988). The ecological basis of multispecies fisheries, and management of a demersal fishery in tropical Australia. In: Gulland, J. A. (Ed.), *Fish population dynamics*, (2nd ed.), Chapter 14, pp.349-82, John Wiley & Sons Ltd.

Sainsbury, K. J., (1991). Application of an experimental approach to management of a tropical multispecies fishery with highly uncertain dynamics. *ICES Mar. Sci. Symp.*, 193, 301-320.

Sainsbury, K.J., Campbell, R.A., Lindholm, R., and Whitelaw, A.W. (1997). Experimental management of an Australian multispecies fishery: examining the possibility of trawl-induced habitat modification. pp.107-112. In: Pikitch, E.K., Huppert, D.D., and Sissenwine, M.P. (Eds.), *Global Trends: Fisheries Management.* American Fisheries Society: Bethesda, Maryland.

Sainsbury, K.J., Punt, A.E., and Smith, A.D.M., (2000). Design of operational management strategies for achieving fishery ecosystem objectives. *ICES J. Mar. Sci.*, 57, 731-41.

Stephens, D.W. and Krebs, J.R. (1986). *Foraging Theory*. Princeton University Press, Princeton, New Jersey.

Tesfatsion, L., (2001). Introduction to the special issue on agent-based computational economics. *Computational Economics*, 18, 1-8.

Tilman, D. and Kareiva, P. (Eds.), (1997). *Spatial ecology: the role of space in population dynamics and interspecific interactions.* Princeton University Press, Princeton, New Jersey, USA.

Van Dyke Parunak, H., Savit, R., and Riolo, R. (1998). Agent based modelling vs equation based modelling: a case study and user's guide. In: Sichman, J., Conte, R., Gilbert, N. (Eds.), *Multi-Agent Systems and Agent Based Simulation (Lecture Notes in Computer Science)*, Springer, pp.10-25.

Waszniowski, L. and Hanzalek, Z. (2004). Analysis of Real Time Operating System Based Applications. In: Kim G. Larsen, P. Niebert. *Formal Modeling and Analysis of Timed Systems. First International Workshop, FORMATS 2003*, Marseille, France, September 6-7, 2003. *Lecture Notes in Computer Science:* Springer-Verlag. pp.219-233.

Yap, P. (2002). *Advances in Artificial Intelligence*: 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002 Calgary, Canada, May 27-29, 2002. Proceedings. Lecture Notes in Computer Science: Publisher: Springer-Verlag.

APPENDIX A: NAMING CONVENTIONS

Table A.1: Naming conventions used in equation construction – to be used as a guide when reading the equations in the text of this document.

Symbol	Description/Notes
English alphabet	
A	area
a	age
B	biomass
b	width of an individual (e.g. individual animal)
c	contaminant
C	current vector
C	concentration
D	damage (e.g. level of damage done to benthics by cyclones)
d	depth or distance
dt	time-step
E	effort
e	the natural number e, though usually written exp in the equations; used as a subscript to indicate evader parameters in evader-threat equations
F	fishing mortality (numbers taken/killed by fishing operations)
f	stands for feared - used as a subscript to indicate threat parameters in evader-threat equations
F	flow
G	fishing boat Kalman filter transition rate
g	gut contents
H	variability of CPUE assumed in fishing boat Kalman filter; port capacity
h	stands for hunter - used as a subscript to indicate predator parameters in predator-prey equations; steepness parameter
I	Irradiance
i	used as an index in sums etc
j	used as an index in sums etc or as random number placeholder
K	carrying capacity
L	Parameters or values associated with locations or movement (e.g. L_l)
$L(x_t,y_t,z_t)$	location in 3D space at time t
l	length of an individual (e.g. individual animal)
M	numbers dying (mortality)
m	mortality rate; month
N	numbers (e.g. animal abundance)
n	number of data points; number of cells in grids etc
P	catch per unit effort (CPUE) error
p	proportions or probabilities; or stands for prey - used as a subscript to indicate prey parameters in predator-prey equations; port
Q	variability of species assumed by fishing boats
q	catchability – not used yet but left for catchability in FMA agent description
R	reproduction or recruitment (so numbers spawned etc)
r	radius
R	diffusion scalar
S	spawner biomass
s	speeds; sex

Symbol	Description/Notes
s_{yr}	seconds in a year
t	time
T	technology creep index
U	light limitation
v	direction vector
V	vulnerability or selectivity
W	wind vector
W_m	weight ogive for metabolic functions
W	weight of an individual (e.g. individual animal)
X	longitudinal coordinate (projection in metres); catch per unit effort (CPUE)
Y	latitudinal coordinate (projection in metres); year
Z	vertical coordinate
Greek symbols	
α	parameters (e.g. Beverton-Holt alpha)
β	parameters (e.g. Beverton-Holt beta)
γ	parameters (e.g. length of fallow periods); fishing zone
δ	desirability weightings for animal movement or flags (i.e. anything where calculate a value and if that is > 0 then continue to the next step of the decision tree) or probabilities (where must then draw random number < or > etc to continue on)
ε	parameters
ζ	parameters
η	habitat parameters – suitability, preferences and thresholds
Θ	parameters
θ	parameters and angles
ι	ratings (e.g. of prey or threat – the preferences and fears in the agents files)
κ	rate parameters
λ	coefficients (e.g. taxon specific length-weight parameters); log-likelihood
μ	coefficients (e.g. fecundity limits, growth rates)
υ	Parameters
ξ	Parameters
π	constant = 3.141….; profit
ρ	proportions
σ	random numbers
τ	contaminant uptake parameters
υ	selectivity parameters
Φ	random number
ϕ	diffusion constants
χ	parameters (e.g. number of benthic age classes)
ψ	sediment suitability rating
ω	scalars and weighting factors (e.g. with respect to environmental assessments, and wind and currents when working out velocities)
φ	parameters
ϖ	value of external forcing function
Math symbols	
[ ]	Use only the integer portion of the value within the [ ] (i.e. used it where, in the code, there was a floor() call)

Table A.2: List of parameters referred to in the text of this document with the name they are given in the model agent configuration files (and code).

Variable	Role	Parameter name in agent file
	Introduced in Ch. 1	
ρ_{rad}	proportion of diffusion that is radial	RadialProportion
ϕ_{diffR}	radial diffusion	RadialDiffusion
ϕ_{diffA}	areal diffusion	ArealDiffusion
	Introduced in Ch. 2	
ω_w	wind weighting coefficient	Wind_k
ω_c	current weighting coefficient	Current_k
d_{max}	maximum depth for taxon	maximum_depth
d^B_{opt}	best depth for taxon	best_depth
d_{min}	minimum depth for taxon	minimum_depth
d^A_{opt}	best depth of seabed for taxon	best_sea_depth
ω_d	weighting coefficient for sea depth preference	ScalingBestSeaDepth
$\eta_{pref,i}$	preference weighting for a given habitat	habitat:*habitat-type*
$\eta_{thresh,i}$	threshhold for habitat type	habitat:*habitat-type*_thresh
Θ	search exponent (looseness)	SearchRadiusExponent
$\kappa_{h,gape}$	gape/width scalar	gape/width
$\kappa_{h,gut}$	gut capacity scalar (capacity/mass)	capacity
$\iota_{p,h}$	preference of predator *p* for prey *h*	prefers:*prey-taxon*
$\iota_{e,f}$	fear of *f* according to *e*	fears:*predator-taxon*
d_{safe}	safe zone radius	safe_range
κ_t	interval between application of natural mortality	cull_period
a_{max}	maximum permitted age for species	Maximum_Age
κ^0_M	base mortality rate	rate_mortality
T_0	age at recruitment to adult population	recruitment:age
$\kappa_{peakmet}$	peak metabolic rate	Peak_Metabolism
λ_l	length coefficient	length_t
λ_w	weight coefficient	weight_t
w_{max}	maximum weight for taxon	Max_mass
λ_{mass}	growth correction coefficient	mass_lambda
a_{mat}	age at sexual maturity	breeding:age
$t_{startcycle,d}$	start of breeding period	breeding:start
$t_{period,d}$	duration of one breeding cycle	breeding:period
Ω	offset	breeding:offset
γ_{spawn}	fallow period after spawning	breeding:fallow_period
$\varpi_{ref,i}$	reference level for external fecundity modifier	fecundity:referencelevel*i*
μ_f	base fecundity	fecundity
μ_r	condition specific rate modifier	fecundity_rate
N_{std}	standard size for a spawned group	group_size
A	number of age classes	maximum_age
	Introduced in Ch. 3	
M	natural mortality rate	Rate_Mortality

Variable	Role	Parameter name in agent file
$\nu_{0.25}$	Length at 25% selectivity	sel25
$\nu_{0.5}$	Length at 50% selectivity	sel50
α	Beverton-Holt α	bhA
β	Beverton-Holt β	bhB
l_∞	Maximum length of an animal in a population	Flinf and Mlinf
κ	von Bertalanffy steepness parameter	Flength_lambda and Mlength_lambda
t_0	age of recruitment for populations	Fvbt0 and Mvbt0
λ_1	allometric length-weight relationship scalar	Flengtha and Mlengtha
λ_2	allometric length-weight relationship exponent	Flengthb and Mlengthb
β_j	spatial recruitment factor from blastula or larva	SpatialLarvalFactor
	Introduced in Ch. 4	
ρ_{growth}	radial growth coefficient	spawn_growth_proportion
	Introduced in Ch. 5	
μ_s	horizontal growth for small benthos	Small:SpreadRate
λ	index of spread for logistic growth function	SpreadGrow
ν	inflection point of growth logistic	MiddleGrow
ξ	rate of recruitment for smallest specimens	Small: RecruitRate
κ_{rec}	constant recruitment term for external sources	RecruitConstant
κ_s	natural mortality rate for small benthos	Small:MortalityRate
θ	index of spread for logistic age-structured mortality	SpreadDie
φ	inflection point for logistic age-structured mortality	MiddleDie
ω	vertical growth rate	ProbabilityGrowBig
ϕ	index of spread for the logistic function for the transition to the large group	SpreadBig
ε	inflection point for the logistic function for the transition to the large group	MiddleBig
χ	number of age classes	NumberAgeGroups
μ_L	horizontal spread of large benthos	Large:SpreadRate
ϖ	coefficient of depth effect on spread	Large:DepthCoefficient
ζ	coefficient of sediment effect on spread	Large:SedimentCoefficient
κ_L	natural mortality rate for large benthos	Large:MortalityRate
μ	horizontal growth rate for macrophytes	SpreadRate
I_{top}	irradiation at sea surface	irr_top
γ	light extinction coefficient	k_extinction
κ	natural mortality of macrophytes	MortalityRate
	Introduced in Ch. 6	
$\kappa_{cap,a}$	mass of juveniles supported per kg m^{-2} of habitat	CarryingCapacityAlpha
$\kappa_{cap,b}$	minimum area to support one kg of juveniles	CarryingCapacityBeta
	mortality rate	Larva:mortality
a_{mat}	age at which the juvenile recruits	Recruitment:Age

Variable	Role	Parameter name in agent file
	Introduced in Ch. 7	
K_{α}	mass of juveniles supported per kg m^{-2} of habitat	CarryingCapacityAlpha
K_{β}	minimum area to support one kg of juveniles	CarryingCapacityBeta
κ_{aggreg}	aggregation rate as recruitment approaches	aggregate_rate
ρ_{M}	proportion of population which is male	propMale
	Introduced in Ch. 9	
$\kappa_{L,cat}$	mortality coefficient for damage for large benthos due to storms	Large:DamageRate:Storm
$\kappa_{s,cat}$	mortality coefficient for damage for small benthos due to storms	Small:DamageRate:Storm
κ_{cat}	mortality coefficient due to storms	DamageRate:Storm
	Introduced in Ch. 10	
τ_{l}	linear uptake rate for a contaminant	*cont*:L_Uptake
τ_{s}	sigmoidal uptake coefficient	*cont*:NL_Uptake
$\alpha_{c,1}$	the first segment of a piecewise linear uptake	*cont*:Controlled_Slope
$\alpha_{c,2}$	the second part of a piecewise linear uptake	*cont*:Uncontrolled_Slope
$C_{c,thresh}$	the transition threshhold concentration between $\alpha_{c,1}$ and $\alpha_{c,2}$	*cont*:Controlled_Uptake
$T_{half-life}$	Halflife of contaminant	*cont*:halflife
λ_{d}	decay exponent	*cont*:lambda
	Introduced in Ch. 18	
C_{p}	number of vessels a port can accomodate	Capacity
H_{p}	capacity threshhold	Threshold

APPENDIX B: C++ IMPLEMENTATION OF NWS-INVITRO

B.1 Introduction

When implementing a model in code there are always a few details that are not initially clear from the description of the mathematical formulation. The following is a description of the computer software used to implement *NWS-InVitro*. It is written in the C++ computer language for use on computers with the LINUX operating system installed. This guide complements the model description given in the main body of the report.

This appendix concentrates on the more abstruse parts of the code: namely those parts which are either not clearly a part of a sub-model, or those which would benefit from more description than is appropriate in the main document. Any sections not covered explicitly in this appendix were coded in as written in the main document chapters, and their implementation was straightforward and required no extra explanation here.

Note: Throughout this appendix words in the Arial font (e.g. species.hxx) format indicate C++ filenames and words in the `Courier` font (e.g. `species_data`) indicate actual code sections.

B.2 Model configuration files

As is clear from the body of the report, all components of the model have been implemented as agents. Initialisation of the agents is the first step in running the *NWS-InVitro* software. There are two basic systems for initialising agents specified in files:

- loading from "cfg" files; or
- loading from a "spinup".

The former, cfg (abbreviated from config) initialisation, is what has typically been used in the North West Shelf project. cfg files are essentially simple text lists of individual agents to be instantiated in the simulation (with some specific parameterisation and any information that makes each agent unique). In this scheme the agents required for a simulation are listed in/as one or more files and they are instantiated with values taken from parameterisation files or default values when the program begins. "Spinup" files are generated by dumping the state of the model's agents at the end of a run. This process almost preserves the entire state of each agent, so agents loaded from a "spinup" file are nearly indistinguishable from agents which have run from the time they were instantiated in the first run. This facility allows a number of "future realisations" from a common starting point. Since we did not use the spinup mechanism to any significant degree, and it has not been maintained, we will not describe it here except to note that the code for the "spinup" files resides in species.cxx, species.hxx with routines in each of the classes. Loading and dumping the spinup data use the `species_data` routines and data structures which are usually employed to deal with initialising the sub-model's parameters.

Almost all of the "generic" code concerned with the initialisation of agents listed in CFG files is found in either cratchett.cxx, agentlist or manifest.hxx. The code in

agentlist and in manifest.hxx consists of macros which use textual (context-free) substitution to resolve to calls to the appropriate constructors and initialisation routines. This works because macros are textual substitutions so the normal name resolution in C++ applies. The code in cratchett.cxx deals more with the parsing of the CFG file.

B.2.1 Initialisation by CFG files

Multiple CFG files may be specified on the command line and each is processed in the order specified. "Fully processed" means that the agents in that file will be completely constructed and initialised, but no initialisation occurs if the agents being depend on other agents.

`LoadAgent()` is a code procedure that steps through the file and dispatches each line to `insert_config_line`. The line is broken down into a base taxon, a species (agent file), a pass number (either by default using `WhichPass()` or explicitly specified in the file), and a set of arguments with which to initialise the agent. This data set is sorted and executed in the order indicated by its position in the file and its pass number.

`WhichPass()` is a code procedure that takes the classname and returns a value indicating the default pass through the CFG tree in which the agent ought to be initialised. If agents are not processed in the correct order a situation may occur such as an agent erroneously spawning to Larva agents rather than Blastula since part of its initialisation tests for the presence of Blastula to indicate which method is appropriate.

Once the file has been completely entered into the tree, the tree is traversed in several passes. During each pass, the passnumber is compared with the pass and the agent's cfg node entry, if they match the agent is initialised using `cfg_run()` and marked as "initialised".

`cfg_run` is a code procedure that calls `make_agent` procedure which acts a dispatch routine to the macros mentioned above. These macros are particularly complex and deserve some discussion in their own right. The following code fragment can be found at the beginning of the file Agentlist:

```
#define InstAgent(CLASS,sp,s) \
    junk = (!strcasecmp(baseclass,#CLASS) \
 && (a = agentlist[(LastAgent = (NewAgent(CLASS, sp, s))) + 1]))
```

with macro calls further down which look like the following:

```
InstAgent(Agent, sp, s);
InstAgent(Catastrophe, sp, s);
```

These calls are part of the "agent recognition code" which maps the lines of text in the configuration files with routines to instantiate and initialise the agents. The macro textually includes a set of assignments to the variable a if and only if the baseclass

matches string specified as the first argument in the InstAgent(). Notice that the first argument is not written as a string – this is important. The "#" in front of CLASS turns whatever is passed, into a string constant when the macro is textually expanded. Notice, that CLASS is passed to NewAgent() *without* the "#". If the baseclass and the CLASS match, then the first part of the "&&" is true, so the second part is evaluated and we get our assignment to a. There should *never* be more than one InstAgent() clause for a class, and if a class fails to initialise at all, there is a good chance that there is no InstAgent() clause corresponding to it. InstAgent()appears **only** in Agentlist – if you see it anywhere else there is something wrong.

`NewAgent` is another piece of complex code, which is found in the file manifest.hxx. There are at least four incarnations of it, the default looks like this:

```
  #define NewAgent(CLASS, sp, args...) \\
({ int na; for (na = 0; na < nagents && agentlist[na]; na++); \\
       if (na >= nagents) na = -1; \\
       (agentlist [na >=0?na:nagents] = new CLASS); \\
((CLASS *)agentlist[na>=0?na:nagents])->Init(sp, ## args); \\
       if(na<0) nagents++; \\
       (na>=0?na:(nagents-1)); })
```

The initialisation of agents is always done through a call to `NewAgent.` This is because there are two entry points to the path which create agents, and `NewAgent` ties them together in a single apparent interface

The first thing the code does is find the first empty place in the agentlist array. The first empty place may occur before the last occupied cell (due to the termination of another agent), and if this is the case the variable na is set to the appropriate index, otherwise na is set to -1 and nagents is used to dereference the array. This empty cell is assigned the value returned by the constructor for the indicated CLASS (remember, it was not passed as a string and we are dealing with textual substitutions). The cell is explicitly cast to the correct class and the initialisation routine `CLASS::Init(char *, char *)` is called. This macro returns the index of the cell in Agentlist which was assigned the new agent.

The macro implicitly selects the appropriate constructor since `CLASS` is textually substituted with whatever the appropriate class is. Once this has happened, a call to the agent's `Init()` routine is made. In the code of the modelling framework there are two distinct forms of initialisation:

- that used when initialising with the parameters represented in a character string; and
- that used when the parameters are passed on the stack.

The pasting operator (# #) in the macro appends the arguments `NewAgent` was passed, to `Init()` textually, so

```
NewAgent(Flobberworm, "flatus-horribilis", length, mass, gas)
```

gets expanded to

```
(agentlist[na>=0?na:nagents] = new Flobberworm);
((Flobberworm *)agentlist[na>=0?na:nagents])->
               Init("flatus-horribilis", length, mass, gas);
```

while

```
NewAgent(Flobberworm, "flatus-horribilis", "42 1.4 98");
```

becomes

```
(agentlist[na>=0?na:nagents] = new Flobberworm);
((Flobberworm *)agentlist[na>=0?na:nagents])->
               Init("flatus-horribilis", "42 1.4 98");
```

The constructor called is the same for each of them, but the initialisation routine is not. Typically the second `Init()` will almost always call the first after the parameters have been parsed or generated, but this is not necessarily the case.

B.2.2 Species parameters

All of the parameterisation data for the model resides either in the files that select the agents to be included in the run, or in parameter files that are read in at the start of the system. These files contain data that may be selected according to the type of the agent (for entities which may be represented by more than one type of sub-model), or the scenario selected. By concentrating the parameters of an agent in files tied to the entity being modelled rather than the attribute modelled, alternative hypotheses or representations for the entity can be readily catered.

The species.hxx and species.cxx files contain the definition for the `Species` class, of which the global variable `species_data` is the most important representative. This class is used to hold the whole parameterisation of the model (at least at the start), and to load the state of a model from a dump file. The parameterisations of the various sub-models are taken from files whose location is specified on the command line, either explicitly or implicitly by naming a directory that contains them. These files are read, parsed and converted to a usable form.

When the command line parser encounters the "`-species`" option it passes the next string on the command line to the routine `load_species()` which is a wrapper for the more general `load_parameters()`. This routine takes a name and a pointer to a

structure to populate – if the pointer passed is null a new structure is created. If the name is associated with a file, the contents of the file are read into a buffer and are "wrapped" with some syntactic sugar to include the name of the file in the resolution path to any given entry in the file. This simplifies the parsing, since the same code can be used for handling files in directories (which implicitly specify a highest level tag with the filename) and simple files. Alternatively if the name is associated with a directory, the code descends (recursively) into the directory and the `load_parameters()` routine is called for each (valid) entry in the directory – some classes of names are excluded: backup files, names that begin with a dot, and such like.

The work is done (again recursively) by the routine `pfile()`. This routine, and the routines it calls, step through the buffer building up the data structure.

Once loaded, the process of resolving a set of "tags" (like "*kingprawn:juvenile:Bh*") consists of converting them to separate strings ("*kingprawn*", "*juvenile*", "*Bh*") and stepping through the tree structure till the end is reached. During initialisation the code tends to create paths as you request or access them (defaulting to zero or null), but once initialised these paths are not necessarily automatically created. In particular if a path includes a "+" at the beginning of a tag, that tag and the path leading up to it must be present or the program aborts. Similarly a "~" will cause a warning message, but not abort. This is so we may flag particular parameters as parameters which must be specified.

Some species files, such as the one below which is taken from the front of NBoutfall have parameters which look like so:

```
base_taxon = "Outfall"

    tag = "NickolBay"

    scale {
        default = 1.0
        Low = 0.8
        Lower = 0.6
        Lowest = 0.4
        High = 2.0
        Rabid = 4.0
    }

    #offset = 0.01
```

This indicates that any parameter which isn't specified in NBoutfall should be looked for in Outfall. Outfall is the least speciated taxon of the agent, unless it too, has a "base_taxon =" parameter. For completeness (and to show that it *does not* have a base_taxon), here is Outfall:

```
immutable = 1
is_a_contaminant = 1
lethal_contaminants {
}
```

In agent.hxx there is a lengthy set of macros which wrap many (but not all) calls into the species database. The code looks like so:

```
#define Has_Parameter(sd, t, args...) \\
    ({int f = 0; \\
        if (!t && !taxon_depth) f = 0; \\
        else if ((!t || t == taxon) && taxon_depth) \\
                f = (sd)->Has_Param(base_taxon, ##args); \\
        else if (t && !taxon_depth) f = (sd)->Has_Param((t),
##args); \\
        else f = (sd)->Has_Param((base_taxon), ##args)
                || (sd)->Has_Param((t), ##args); \\
        f;})

#define ParameterGroup(sd, t, args...) \\
    ({ParamBlock *f = 0; \\
        if (!t && !taxon_depth) f = 0; \\
        else if ((!t || t == taxon)  && taxon_depth) \\
                f = (sd)->Get_ParamGroup(base_taxon, ##args); \\
        else if (t && !taxon_depth) \\
                f = (sd)->Get_ParamGroup((t), ##args); \\
        else if ((sd)->Has_Param((base_taxon), ##args)) \\
                f = (sd)->Get_ParamGroup((base_taxon), ##args); \\
        else f = (sd)->Get_ParamGroup((t), ##args); \\
        f;})
#define ParameterBlock(sd, t, args...) \\
    ({ParamBlock *f = 0; \\
        if (!t && !taxon_depth) f = 0; \\
        else if ((!t || t == taxon)  && taxon_depth) \\
                f = (sd)->Get_ParamBlock(base_taxon, ##args); \\
        else if (t && !taxon_depth) \\
                f = (sd)->Get_ParamBlock((t), ##args); \\
        else if ((sd)->Has_Param((base_taxon), ##args)) \\
                f = (sd)->Get_ParamBlock((base_taxon), ##args); \\
        else f = (sd)->Get_ParamBlock((t), ##args); \\
        f;})
```

```
#define Parameter(sd, t, args...) \\
   ({double f = 0; \\
        if (!t && !taxon_depth) f = 0; \\
        else if ((!t || t == taxon)  && taxon_depth) \\
                f = (sd)->Param(base_taxon, ##args); \\
        else if (t && !taxon_depth) \\
                f = (sd)->Param((t), ##args); \\
        else if ((sd)->Has_Param((base_taxon), ##args)) \\
                f = (sd)->Param((base_taxon), ##args); \\
        else f = (sd)->Param((t), ##args); \\
        f;})

#define SParameter(sd, t, args...) \\
   ({char *f = 0; \\
        if (!t && !taxon_depth) f = 0; \\
        else if ((!t || t == taxon)  && taxon_depth) \\
                f = (sd)->S_Param(base_taxon, ##args); \\
        else if (t && !taxon_depth) \\
                f = (sd)->S_Param((t), ##args); \\
        else if ((sd)->Has_Param((base_taxon), ##args)) \\
                f = (sd)->S_Param((base_taxon), ##args); \\
        else f = (sd)->S_Param((t), ##args); \\
        f;})
```

This inserts the agent's `base_taxon` array into the resolution path. When a call to `Parameter(taxon,"lethal_contaminants","bitterns")` is made, it will attempt to resolve it by trying the "most speciated" (analogous to picking "mouse" as a type of "rodent", which in turn is a "mammal") taxon in the list first (`base_taxon[0]`) and proceeding, each in turn, to the least speciated until it resolves the path to the parameter. When there is a set of `base_taxon` names relevant to an instance of an agent, the taxon for that agent is set to the value of the least speciated taxon in the list.

As an aside, at the beginning of the species.hxx file there are quite a few `#define` statements that look something like this:

```
   #define DECLARE_ALL_STRINGARGS_N char *s1, char *s2, char
*s3, char *s4,\\              char *s5, char *s6
   #define DECLARE_ALL_STRINGARGS char *s1 = 0, char *s2 = 0,
char *s3 = 0,\\              char *s4 = 0, char *s5 = 0, char
*s6 = 0
   #define DECLARE_ALL_STRINGARGS_1 char *s1, char *s2 = 0,
char *s3 = 0,\\               char *s4 = 0, char *s5 = 0, char
*s6 = 0
   #define DECLARE_ALL_STRINGARGS_12 char *s1, char *s2, char
*s3 = 0,\\
        char *s4 = 0, char *s5 = 0, char *s6 = 0

   #define STRINGARGS_HEAD s1, s2, s3, s4, s5
   #define STRINGARGS_TAIL s2, s3, s4, s5, s6
   #define ALL_STRINGARGS s1, STRINGARGS_TAIL
```

```
   #define STRINGARGS_TAIL_OUT s2 << ":" << s3 << ":" << s4 <<
":" \\
             << s5 << ":" << s6
   #define STRINGARGS_OUT s1 << STRINGARGS_TAIL_OUT

   #define HEAD_OK (s1 && *s1)
   #define TAIL_OK (s2 || s3 || s4 || s5 || s6)

   #define LAST_STRINGARG s6
   #define LAST_VALID_STRINGARG
(s6?s6:(s5?s5:(s4?s4:(s3?s3:(s2?s2:s1)))))

   #define SHIFT_STRINGARGS ({if (s0) {s0 = s1; s1 = s2; s2 =
s3; s3 = s4; \\
       s4 = s5; s5 = s6; s6 = 0;} s1;})
```

The purpose behind these is to make the definition of the routines in the class which manipulate and access the data in `Species` structures consistent, and easy to modify. Early in the development of the model, only three strings were passed. This proved to be inadequate, so it was increased to four. This was also inadequate, and so the definitions and calls within the species.cxx and species.hxx files were recast to use the macros so that any future changes would not require a tedious (error-prone) procedure-by-procedure edit.

While the main reason for the species.cxx and species.hxx files is to define `Species`, two other classes are also defined – `ForcingVariable` and `ForcingSystem` (which has an array of `ForcingVariables`). These are used as time series variable. More recently there is a `timeseries` class and a `TimeSeries` agent which (ultimately) will be what gets used in place of `ForcingVariables` and `ForcingSystems`. These classes/agents are not dealt with in this document.

****WARNING****

An agent requesting `species_data` information about another agent ought to use the `species_data->Param(...)` call directly: failing to do so will almost certainly mean that you get either an empty parameter (zero or null), or a value corresponding to some other entity: the base_taxon chain will be used in preference to the "taxon" specified.

B.2.3 Initialising agents

Every agent exercises the constructor of the base class `Agent`. This class is what gives the other agents the machinery to use the scheduler and the run queues. It is also the case that part of the initialisation path for `Agents` will be exercised every time an agent of any class is initialised. In this context the initialisation of an `Agent` is examined in more detail.

The first step in the construction of an `Agent` is zeroing the memory of the agent. Note that this applies only to the agent level memory and does not imply that the class variables derived from an `Agent` are zeroed. This memory zeroing is done here since it is common to all the agents.

There are several class variables in `Agent` which are of importance to the initialisation process. Most important is `initialising` which is a three state variable. This variable basically controls when `Dependent_Init()` is called. Next is the variable `id` which is initialised from a static variable `Ego` which assigns a unique serial number to each agent. `id` is the only way to uniquely identify agents without relying on pointers to fixed memory locations.

```
Agent() MEM_WATCH_INIT {
   memset(this, 0, sizeof(*this));
   debug_state = 0; // this variable is used to hold state
information
                  // on an agent-by-agent basis for things like
raising
                  // signals only once or other such things
   initialising = -1; // starts at -1, is 1 during the init
process
                    // and goes to 0 when fully initialised
   id = Ego++;
   priority = NewPriority(DEFAULT_PRIORITY);
   int_i_am = string_register->String("agent");
   i_am = string_register->String(int_i_am);
   state = 0;
   //Set_State(state, NotYetRun); // not yet active and
introduced into
                                  // run queue
   check_dependencies = 0;
   agent_ix = -1; // not yet initialised
```

The variable `i_am` is used in identifying which class the agent is at a very low level in the code. `check_dependencies` is used to indicate whether the state of other agents upon which the agent may depend should be checked each time the agent gets a slice of time. In general this will not be the case. `agent_ix` is a convenience variable which stores the index of the agent in the `agentlist`. The rest of the initialisations in the constructor basically set things to either reasonable defaults, or known "dud" values that trigger other actions. Every constructor of a derived class must call this constructor, this may be indirectly (through its parent class for example).

After `NewAgent()` calls the constructor, it calls one of the `Init()` routines. These routines are not virtual routines, and the type resolution rules force the call to go to the

`Init ()` routine of the appropriate class. The `Init()` routines of `Agent` are very simple, but they are very similar in form to those of the other classes.

So we get from agent.hxx:

```
/*— void Init(char *genera, char *cfgstr) */
void Init(char *genera, char *cfgstr) {
// char line[128];
char commence[128] = " ", finish[128] = " \ ";
if (sscanf(cfgstr, " % [0-9/:,] % [0-9/:,] ", commence, finish)
! = 2) {
fprintf(stderr, "agent error \\n ");
exit(1);
}
subjective_time = startat = parse_time(commence);
endat = parse_time(finish);
if (startat == BeforeEpoch) startat = subjective_time =
DEFAULTTIME;
if (endat == AfterEpoch) endat = DEFAULTTIME;
Init(genera, startat, endat);
} ;
```

This basically parses the cfg string (from the config file) and passes it on to the other `Init()` routine. This is applied to every agent.

Turning to the explicit `Init()` routine we obtain from agent.cxx the following:

```
/*— void Init(char *genera, Time stat, Time enat) */
void Agent::Init(char *genera, Time stat, Time enat) {
Independent_Init(genera, stat, enat);
if (! Failed_Init_Dependency()) {
Dependent_Init();
}
Register();
} ;
```

This `Init` is characteristic of most agents, though some have additional code to do things like test whether the supplied values lie within allowed ranges (say, geographic location, total population, etc). The salient features of this routine are:

- first, call `Independent_Init`;
- a test is made to see if the call to `Dependent_Init()` may proceed using the virtual function `Failed_Init_Dependency()`;
- if it is clear, the call to `Dependent_Init()` is made; and
- finally a call to `Register()` is made to enter the agent into the run queue.

Notice that all the parameters of `Agent::Init(char *, Time, Time)` are passed on to the `Independent_Init()` routine. `Independent_Init()` is responsible for initialising all the parts of the agent which do not depend on the presence or state of initialisation of any other agents. By initialising the agents in this two-step process and keeping the amount each must know/remember about the others to an absolute minimum, we by-pass problems with "circular" initialisation. Values which need to know about other agents (i.e. where to find the spawning habitat agent) can be left till all the agents have completed their `Independent_Init()` and are correctly inserted into the agentlist.

Looking through `Agent::Independent_Init()` almost the first thing it does is to test if it has already done its `Independent_Init()`. If so, it quietly returns.

```
void Agent::Independent_Init(char *genera, Time stat, Time
enat) {
<...>
if (initialising >= 0) return;
VERBOSE( "Independent_Init ", genera);
taxon = string_register->SString(genera);
base_taxon = species_data->S_Param(taxon, "base_taxon ");
if (base_taxon) {
char *tt = base_taxon;
base_taxon = taxon;
taxon = tt;
}
<...>
```

What follows the test is a little bit of prestidigitation. `taxon` and `base_taxon` form a pair of strings used both in the initialisation and identification of an agent. On entering the `Independent_Init` code, `taxon` holds the string specified in the species part of the cfg file. If a `base_taxon` is specified in the species file the values of taxon and `base_taxon` are swapped. The motivation behind this sleight of hand is to allow 'generic' species files (such as "Outfall") and allow speciation ("NickolBayOutfall") to overlay the default values. This has a deep effect on the resolution of parameter values from the `species_data` structure. When a call to `Parameter()` is made the values associated with the contents of `base_taxon` are used in preference to the values associated with the contents of `taxon`. If no `base_taxon` is specified, the value is resolved. Note that issues to do with naming can arise because there are alternative mechanisms for parameter loading, so:

**** WARNING ****

Scope exists for errors to creep in when an agent requests a value of another taxon. A candidate scenario for failure might be when an agent with a `base_taxon` of "wombat" executes code like so:

```
  friends_radius = Parameter(species_data, friend->taxon,
"radius");
```

The radius associated with "wombat" will be returned rather than the radius associated with the taxon of the friend. Moreover, trying to resolve something like:

```
misc_value = species_data->Param(base_taxon, "MValue");
```

will fail if no `base_taxon` is specified.

The following code fragment in `Agent::Independent_Init` sees to the higher level identification data of the agent. Notice that vessels have an idiosyncratic naming scheme (the `name` of a vessel or boat does not necessarily correspond to its `taxon` in any way). The start and end times are also initialised here.

```
    if (isa(A_VESSEL)) {
       sprintf(line, " % s_% s- % 07ld ",
           Query(this,"vesselname "),taxon,(long)id);
    }
    else {
       sprintf(line, " % s- % 07ld ",taxon,(long)id);
    }
    name = string_register->SString(line);
    Set_State(state, Active);
    VERBOSE( "Independent_Init ", name);
    subjective_time = startat = stat;
    endat = enat;
    if (species_data->Has_Param(taxon, "starttime "))
       subjective_time =
       startat = (Time)Parameter(species_data, taxon, "starttime
");
    if (species_data->Has_Param(taxon, "finishtime "))
       endat = (Time)Parameter(species_data, taxon, "finishtime
");
    if (! startat) subjective_time = startat = starttime;
  if (! endat) endat = finishtime;
```

The loop below is used to determine if the specified `taxon`/ `base_taxon` has a species file from which to take initialisation data. If no such file exists, an "empty" entry in the `species_data` structure is created.

**** WARNING ****

There may be conceivable interactions with the `base_taxon` (though none have yet arisen).

```
   for (sd_species = 0; sd_species < species_data->parameter_set-
    >nentries; sd_species++) {
   if    (strcasecmp(species_data->parameter_set->e[sd_species].tag,
    taxon)) continue;
   else break;
   }
   if (sd_species >= species_data->parameter_set->nentries
   ——! species_data->Get_ParamGroup(taxon)) {
   char *def = 0, *image = 0;
   int linenum = 0, dunny = 0;
   VERBOSE( "SpecDefault ", "Did not find an agent file for " «
taxon « ", " « name « " - creating a default ");
   image = (char *)calloc(strlen(taxon) + 20, sizeof(char));
   sprintf(image, " % s { \\n } \\n ", taxon);
   def = image;
   dunny = species_data->allow_construction;
   species_data->allow_construction = 1;
   species_data->parameter_set  =  species_data->pfile(species_data-
    >parameter_set, &def, "-default- ", linenum);
   species_data->allow_construction = dunny;
   for (sd_species = 0; sd_species < species_data->parameter_set-
    >nentries; sd_species++) {
   if    (strcasecmp(species_data->parameter_set->e[sd_species].tag,
    taxon))
   continue;
   else break;
   }
   if (sd_species >= species_data->parameter_set->nentries) {
   cerr « "Airbag deployed. Unable to initialise "
   « taxon « " - aborting. Please use Only Genuine Parts \\n ";
   abort();
   }
   free(image); // free checked and ok
   }
   sd_data = species_data->Get_ParamGroup((ParamBlock *)0, taxon);
   if (! sd_data) {
   cerr « "Oh bother. My head is stuck in the "
   « taxon « " \\n " « species_data->parameter_set « " \\n ";
   // abort();
}
```

The rest of the routine initialises important variables like tick lengths, and goes to some pains to make sure they are reasonable (big_ticks are no smaller than little_ticks, etc). Contaminants are initialised using an almost identical structure to that used in the agentfile to hold both the data about contaminants in organisms as in the contaminants associated with an outfall. It means that initialisation is simpler, and "reasonable defaults" in the outfall file can be kept to be cut and paste in for organisms that there is no data for.

The last bits of code in the routine sets the start and finish times for the agent and then sets the initialising flag to indicate that the `Independent_Init` has been successfully completed.

```
...
if (! startat) subjective_time = startat = starttime;
if (endat <= startat —— ! endat) endat = finishtime;
initialising = 1;
} ;
```

The routine `Failed_Init_Dependency` (in all its incarnations) must be a virtual function since it is also called in `do_what_you_do` which is what gets called by the scheduler. The reason it is called there is that it is also called in the `Independent_Init()` of each class to see if they can complete their initialisation – if they cannot, this initialisation is deferred till the first time the agent gets loaded into the run queue.

NOTE: If an agent which depends on another is *not* `Register()`ed, it may never run!

**** WARNING ****

`Dependent_Init()` must be a virtual function* too for the same reasons as for `Failed_Init_Dependency`.

*For a comprehensive description of the differences between a "virtual function" and other sorts of member functions in C++, consult the nearest C++ reference manual. Briefly, virtual functions are called through a pointer table associated with the class, while "normal" function are called directly. This means that we can use a single array of pointers to keep track of all the agents and that the specific type of agent is not necessarily important in interactions.

For classes derived from `Agent`, think of the initialisation process as recursive – at each stage the initialisation of the "current class" is dependent on the initialisation of the parent class, and so it dives down. The determination as to whether the `Dependent_Init` can go ahead is much the same, except the call to that first entry point may come from within the base class, `Agent`. The current text-based taxonomy of `Agents` looks like this (graphical representations can be found in chapter 1 of the main document):

Agent: Environment Monitor OilCo Thing
Monitor: Adviser Biomass Catastrophe DOT DPI EPA FMA FishBiomass GraduatePop POPBiomass Purity RecFisher Tracker
Environment: Projection Variable
Projection: CSurface
CSurface: Cadastre DSurface GSurface SCSurface TSurface Tracer
Tracer: PolyOrganism
PolyOrganism: Benthic Larva Seastar
Thing: Animal Fixture Trap Vessel
Animal: Bird Blastula Fish Mammal Population Reptile
Vessel: Boat
Fixture: Port Rig

There is one significant difference between the `Environment` class and its children and the others: `Environments` can take whole lists or directories of time stamped files as well as single files. This is accomplished by using calls (in `CSurface::before_before_behaviour`) to check and possibly refresh the data layers which may have been loaded by `Environment::set_source_data()` (called by `Environment::Independent_Init()`).

B.3 Run-queues and sequencing

The basic form of the Scheduler (which handles temporal stepping within *NWS-InVitro*) is described in chapter 1 (section Agent sequencing), but to facililtate understanding more details regarding its implementation are given below. The routines `cycle` and `enter_queue` sit at the heart of the scheduler. These routines are responsible for ensuring that each agent's requests for time are recorded in the priority queue and serviced. Requests for time may not be granted as the agent requests – they may be truncated to fit in with constraints imposed by `Monitor` agents.

B.3.1 Controlling the flow of time – Agents and Queues

In some ways the *NWS-InVitro* model operates like the queue at a counter. The customers (agents or sub-models) enter the queue and each is able to conduct some or all of their business when their turn comes. It may be that the customers have to join the queue many times before their business is complete, and so it is with agents. This piece-wise approach to getting things done is not usually how the world works, so a more realistic illusion of simultaneous action is created by making the time-steps (the amount each customer can do) relatively small. While the "counter and queue" does not have features that correspond to all the variations which exist within *NWS-InVitro*, it is a reasonable metaphor.

Each sub-model in the system is fundamentally represented as a member of the C++ class Agent. This class provides the basic linkages between the modelling framework (*InVitro*) and the sub-models which flesh out in the system. Roughly speaking these linkages fall into three main categories: those concerned with managing the agent's behaviour in the queues, those concerned with managing the agent's interactions with

other agents, and those which deal with the bookkeeping associated with running the simulation. This section deals with the first of these groups of linkages.

There are two queues from which agents may be run for each pass through the main loop of the system: the so-called "standing queue" and the "ready-to-run" queue (run-queue). Agents in these two queues have fundamentally different behaviours: all agents in the standing queue are required to be contemporaneous with the agent at the top of the run-queue, while no such restriction exists on the agents in the run-queue. Agents in the standing-queue are processed in the order in which they were initially inserted into the queue. In contrast, the agents in the run-queue are sorted by the time at which they wish to run and a priority. They are also randomised within time-priority blocks to ensure no systematic preference based on queue order. The motivation behind having these two queues comes from the observations that some sub-models (such as ocean currents) are both unlikely to be affected by the action of other models, and must be synchronous with all the other sub-models when they are run. Each agent also maintains its own queue of times at which it would like to begin a time-step. So at any given time, an agent is represented only once in the union of the run-queue and standing-queue: all the information regarding its scheduled time-steps is kept in the agent's own memory.

B.3.2 Running through the queues

As each pass through the main loop of the simulation begins, each of the entries in the standing-queue is checked. Any sub-model which is not yet up to the subjective time of the head of the run-queue is run for an appropriate amount of time to bring it into synchrony. The ordering of the standing-queue is not specified and must be assumed to be random at its best and pathologically sorted at its worst. In practice this is not a problem: sub-models in this queue must not be dependent on the state of any other sub-model.

The run-queue is maintained by inserting the agents that are ready to run into a priority queue, sorted principally on time. In order to ensure that sub-models which "ought to be run first" *are* run first, each sub-model (of which there may be many representative agents) is assigned a priority which forms a secondary sort key. A third sort key is also used to randomise the order within each priority group within each time-step. At the end of each pass through the main loop the agent at the top of the priority queue is run for one time-step. It is left up to each agent to reintroduce itself into the priority queue.

The main loop then has the following basic structure:

```
    while (there are any agents in the run-queue) {
          denote the subjective time of the agent at the head
                of the run-queue by "NOW"

          for (each agent in the standing-queue) {
              if (this agent has a subjective time earlier than
NOW) {
                  Run this agent from the standing-queue till its
                           subjective time equals NOW
              }
          }

          run the agent at the head of the run-queue for its
chosen
                  time step
    }
```

The simulation is permitted to run as asynchronously as possible so that the time-steps can then be optimised for the various sub-models the agents embody. It is inefficient to make the sponge beds operate at time-steps which are appropriate for prawns since in the absence of other sources of disturbance a sponge bed changes relatively slowly; so time-steps are adopted for the agents (in this case sponges) which have the property that the state of the agent is "close enough" for the entire duration of the time-step regardless of where in that time-step the subjective time-step of the sponge may lie. This is analogous to the notion that for a continuous function, f, and a value ε, then for any x in the domain of f, there is a t such that $x-t$ and $x+t$ are in the domain of f and $|f(x-t)-f(x+t)|<\varepsilon$.This assumption is reasonable since the sponges are (by-in-large) passive participants in the interaction, and the processes they evince in the simulation are largely continuous in time.

Other more active agents may wish to be active participants in an interaction (say a shark and a fish). In order for this interaction to work the participants must be synchronous – a shark cannot catch a fish if the fish was where the shark is twenty minutes earlier (fish's time). That sentence is difficult to follow since it is expected that everything to happen "at once". Apart from interactions between agents with very specific properties, interaction is not permitted unless agents are synchronous. This is achieved by providing a means for the agents to signal one another so that they may become synchronous. In this example, the shark introduces a "short tick" into its own timestream which brings it to the same subjective time as the fish, and it sends a signals to the fish informing it that it ought to change to a brief tick duration in order to resolve the interaction.

With this structure, agents step through the simulation keeping "as temporally close as possible" to the other agents in the system. There is also no way that an agent can either lose seconds or gain extra seconds. If the agent's subjective time is ahead of what the system perceives the time to be it either re-introduces itself into the run-queue without execution, or it only runs for the balance of it's time-step ($dt'=dt-(subjtime-now)$).

Conversely, if an agent is behind the system's notion of what it ought to be, it catches up – though the same constraints with respect to interaction with other agents still hold.

A typical interaction between two agents can be illustrated by the events surrounding the interaction between the shark and its intended dinner guest. At the outset the shark has decided that hunger is the most imperative mode of behaviour. This causes it to query a spatial data store for the location of near neighbours. The shark examines the properties of these neighbours (sponges, fish, seagrass, a boat), and ranks them according to their value as prey. If the winner of this contest is both temporally coincident and physically close enough to the shark the shark may eat the fish outright. Otherwise the shark informs the fish that it should reset its default tick length to a duration appropriate for this interaction, and both modify their internal queue of times at which they wish to run so as to become synchronous, and the shark re-introduces itself into the run-queue.

At this point in the chain of events there is no way to say whether the fish gets to run before the shark does or the other way around. If the shark runs first, it can close in on the fish, and if it gets close enough it can eat it, otherwise the fish gets to go first (and so increase the gap between it and the shark or find some shelter). The temporal rule which operates here is that the shark and the fish must have a subjective time which coincides sometime in the shark's time-step in order for the shark to eat it. Physically, both apply essentially the same process – they calculate a vector toward or away from some salient feature of the local environment (lunch, teeth or safety) and move with all possible speed in that direction (in a more finely resolved model, inertia and evasive behaviour would be taken into account at this point).

B.3.3 Monitors and the run-queue

Cycle will correctly process each agent in the queue for a suitable amount of time which is determined by the agent's preferred step length and the future times it wishes to run. There is, however, a class of models which are somewhat similar to the agents in the standing-queue, but rather than having their time-step determined by the other agents, they determine a time at which other agents must run. These agents, called `Monitors`, are executed from the run-queue with a high priority so that they are guaranteed to act before other agents. Monitors have a list of targets in which they are interested. Whenever a non-`Monitor` agent enters the run-queue it is checked to see if it matches any `Monitor`'s target list, and if so, an entry is made in its "times to run" queue which will ensure that it is contemporaneous when the `Monitor` is run. In this way, models of recreational fishing, for example, can be implemented which affect mortality each weekend, and be confident that all of the fish removed from the system are synchronous at that time. Similarly cyclones and dredging with `Monitor` agents are modelled.

Because `Monitors` play such a deep role in the scheduling of time-steps, much of the code associated with inserting an agent into the run-queue is associated with ensuring that the constraints these `Monitors` impose are obeyed.

B.4 Implementation of Environment agents

Environment agents are discussed in the main body of the document. The details given here are with regard to the supply of data to the data layer variants of the environmental agents.

B.4.1 Grid-based environments

Many of the data used in the simulation are specified as grid-based environment agents. The archetypal example of this is the bathymetry. During the course of the simulation, many agents will interrogate the bathymetry agent for the depth of the seafloor at a particular location. The `CSurface` class which presents this data to the rest of the simulation has the facility to transparently convert the ordinate space of its underlying data from whatever the native space of the data might be to the common metric space of the model. Moreover, the agents requesting the datum have no notion of either the internal representation of the data or the extent of any processing which may be applied to the data. While it is possible to interpolate between grid-cells to present a smooth surface to the agents querying `CSurfaces`, this is avoided to keep the computational cost down.

Environment grids are arranged as regular geo-referenced arrays, the rows and columns of which correspond either to parallels and meridians or to regular ordinates in a projection into *x*-*y* model space. Bathymetry is represented by a grid of floating point numbers, and a field like that of currents or wind can be represented by a series of time-indexed grids of vectors.

Environments can be made to change through time. It is clear that ocean currents *do* change, for example. The base environmental data can be specified in one of several ways:

- as a simple file;
- as a directory which contains many files whose names correspond to their timestamp;
- as a list of files; or
- by a filename which is determined by the state of a time series agent.

Except in the case of the simple file, which remains static for the duration of the simulation, the data associated with the agent changes as the run progresses. For directories of files and lists of files the processing is straightforward – at initialisation a priority queue of the filenames is built and sorted on the time the data is to be swapped in. When an agent is queried by another, it is a simple and relatively fast matter to hunt through the queue to find the appropriate name and to transparently replace out of date (stale) data before the request is answered. For surfaces that change according to the state of another (governing) agent, the process is only slightly more awkward. The first step is to compare the subjective time of the two agents. If they are the same, there is no need to move to a new table entry, and no action need be taken. Otherwise a new filename is generated and again the stale data is replaced before the request is answered.

B.4.2 Derived surfaces

In contrast to data that are indexed by values returned from another agent, some of the environmental attributes are generated using values obtained from other agents. The best example of this type of environment agent is the ocean current model. By using a current derived from the wind and tides, a broader set of consistent environmental conditions can be generated, and can examine the effect of different weather regimes on the state of the system.

The basic data for the current agent is a large table of spatially referenced coefficients. These data were generated by locally fitting the coefficients to minimise the error of a set of forth degree polynomial models of the currents from MECO (Condie et al. 2006) at specific locations using wind and tidal data (Fulton et al. 2006a).

The effects of wind and ocean currents is integrated into the model at a very low level to ensure that their effects on modelled entities is consistent.

B.4.3 Vertex-based environments

One of the properties of vertex based Environment agents is that they may be advected and diffused, and they may exhibit self motility. This is a fundamental property of every class derived from the `Tracer` class, which includes our representations for pipelines, benthic organisms, larvae and bycatch. `Tracers` may be used to represent both contaminant plumes and, via derived classes, most of the `Environment`-represented biological entities.

The inclusion of the animate models in this class hierarchy may seem a bit odd, but the reason is that in the open water, clouds of larvae (for example) behave in much the same way as plumes of contaminants. Since the physics of advection and diffusion operate on the base class, `Tracer`, its child classes are automatically subject to the same processes. This is particularly important when trying to simulate the organisms suspended in the water column associated with an outflow from some stationary source.

B.5 Utility code

The *InVitro* framework incorporates a body of "utility" code to simplify the process of implementing sub-models or controlling their interactions. Some of this has been discussed in chapter 1 of the main text (e.g. the A* search algorthim), but the rest of the utility code is briefly outlined below.

B.5.1 Geometric projections

At the beginning of the project it was unclear which data would be available and what forms the data would take. The study region encompasses a large area and the data sets anticipated to be of interest were quite disparate. Clearly there was a need to make the model as flexible as possible in what data it could use. Where the conversion of a file from one particular format to another is not overly onerous or likely to have an impact on the nature of the model, the geographic representation of the data may have farther reaching implications.

Many of the entities modelled would be best suited by running them in a three dimensional euclidean space based on the SI units (i.e. metres). Fish, seagrass, and contaminant dispersal are all good examples of these sorts of entities. Fishing vessels are a different matter, however. They simultaneously operate in this Euclidean space (the global model space, incidentally) and in a space where the coordinates are longitude and latitude (LL). All their historical catch data is recorded in LL coordinates as are their reports and control zones that constrain their activity.

To deal with the plurality, each agent that has a "physical presence" in the model has a projection associated with it. This projection maps from its native metric space to the global model space (in our case an MKS system in three dimensions) using the proj4 libraries which are freely obtained from the USGS. Once specified, the projection for a given model transparently converts the data to and from the global model space whenever the agent interacts with another agent. In this way, locations, distances, areas, and velocities are all converted to a common basis so that agents need not know what the internal representation of the data is in other agents.

B.5.2 Priority queues to reduce search time through lists

Priority queues are used for a variety of reason in the model. Their main use is in the implementation of the scheduler. The entity at the top of the run queue is the one that should run next. The structure of the queue ensures that when the entry at the top is deleted, the correct replacement is in place. Priority queues are also used by the navigation algorithm (discussed in chapter 1), in the `Blastula` agent (chapter 6) which maintains the cohorts of spawned animals, and in the system which maintains environmental data that is replaced at particular times from data stored on disk. The use of the queues eliminates the need to search through lists in some of these niches of the model.

Priority queues are usually sorted on a key which is based on the time at which some event is to happen. In order to ensure that we may correctly compare times and keep them ordered correctly, times in the model are all represented as seconds past a nominated "start of the epoch" which typically occurs before any of the data to be used in the model. By mapping times into a number of seconds, agents can usually progress through the entire run without dealing with time issues, and the correct ordering within the queues is ensured. Routines are also provided to convert between the internal representation of dates and times and a calendar representation for output. Special attention has been paid to ensure that we are also able to schedule events that occur periodically based on calendar dates.

APPENDIX C: MAJOR ASSUMPTIONS

A number of assumptions have been made within the sub-models which comprise *NWS-InVitro*. These assumptions are typically taken from literature or expert judgement where there is a lack of data, although assumptions are also made about the way aspects of various sub-models behave. To the extent possible, these assumptions are described and briefly discussed below. We apologise for any assumptions which we have inadvertently omitted from this appendix.

C.1 Physical characteristics

C.1.1 Advection and diffusion

The drift of organisms, polygons, drifters and vessels (including boats) due to currents and wind is generated using a simple model. While a more accurate model could have been implemented which integrated the drift vectors through the path of the entity, this would have been at a computational cost which was deemed to be far too high. Where the time-step of an agent is such that the effects of current or wind would be unduly magnified, the agent's susceptibility to these sources of drift has been correspondingly reduced. The proportions assigned to current and wind for the various agents were arrived at through *ad hoc* questions to senior scientists in CMAR.

The diffusion of larval polygons (and in fact any agent derived from Tracer) is generated as a weighted combination of a radial (linear) diffusion and an areal diffusion. This was seen as a simple means of rapidly approximating a range of diffusivities from footprint with a bounded radius (areal growth) to a footprint with an area which increases geometrically (radial growth). While this may not be a wholly appropriate means of modelling fates of contaminants, as a means of introducing "slicks" of larvae, produced formation water (PFW) or some other material into the water, it does not appear to introduce major anomalies and it has the extreme advantage of a very low overhead.

The way the agents maintain the data concerning the proportions of wind and current which are applied to when advecting permits us to modify these values within the model. In practice, the only time this facility is used is to introduce directed passive movement (due to vertical migration). Thus, a living prawn and a dead prawn experience essentially the same advection regime.

C.1.2 Physical disturbance

The physical effects associated with cyclones, dredging and trawling incorporate several assumptions. Most significantly, there are no turbidity or "burial" effects. In practice, this means that the effect of the footprint is limited to that footprint, with no ancillary damage to the region immediately adjacent to the footprint. The footprint was inflated to cover the area of highest impacts immediately surrounding the swept area, but did not include the tails of any potential plumes. The assumption is then that the damage can be treated within the footprint as homogeneous, and that all significant damage and the most substantial direct impacts are contained within that footprint area

(and are thus accounted for). Another implicit assumption is that the damage done at any spatially displaced spoil or dumping ground is independent of the damage at the main site. There should be a tighter link between the swept area and the spoil grounds, so that the makeup of the sediment in the spoil grounds reflects the impact (e.g. dredging) history. This extension is planned for subsequent models.

C.2 Biological assumptions for animals and populations

C.2.1 Relative importance of modes of behaviour

The basic modes of behaviour evinced by the agents in the "Animal" class are designed to maximise the success of spawning and to minimise fear, hunger and discomfort in that order. In this line of reasoning, spawning takes complete precedence over all other behaviour. The implicit ranking of fear, hunger and discomfort does not mean that extreme discomfort will not overcome some fearsome influence – all the weightings are mapped into the half-open range (0,1), and a sufficiently strong stimulus will overcome any of the other motivations. The relative weightings (2 000, 1 000 and 100, for fear, hunger and discomfort respectively) are, beyond their *order*, admittedly arbitrary, but based on Delphic rankings (CSIRO Marine and Atmospheric Research (CMAR) personnel pers. comm.).

C.2.2 Spawning and recruitment

Animals in poor condition are less likely to spawn than animals in good condition. In *NWS-InVitro* the success of spawning is dependent on the extent to which an animal achieves or exceeds its "nominal" mass (due to increased fecundity). Animals which are below 90% of their nominal mass are prohibited from reproducing. This assumption was based on exploration of the effect of condition on fecundity in other population and ecosystem models (e.g. *Atlantis*, Fulton et al. 2004). The assumption was made that individuals drop to 75% of their nominal body mass after spawning was also made based on discussions with fisheries biologists at CMAR and the dynamics of other models (including *Atlantis*). The specific factors this loss attempts to capture include: the loss of the mass of the offspring; loss due to fasting; and loss due to any other stressors such as migration and site competition.

The final spawning related assumptions are to do with the recruitment for Population agents. This is calculated using a modified Beverton-Holt recruitment function, which is based on the population available to recruit from the Blastula/Larva agents and assumes all relevant mortality, habitat-dependent and compensation factors have occurred within the time the spawned biomass was resident in a Blastula/Larva agent.

C.2.3 Metabolic rates and growth

In the NWSJEMS implementation of *InVitro* we did not explicitly model trophic interaction (largely due to the computational cost). As a result, it was assumed that if the habitat was of sufficient quality then the agents representing organisms in the model were able to get enough food to grow to maturity and reproduce, so the explicit metabolic rates are not used. The explicit metabolic routines were used to condition the FalseMetabolism used in *NWS-InVitro*, though they were not held as an overwhelming

constraint. Consequently, the growth ogive associated with BasalMetabolism() is a simple curve chosen to exhibit the property that growth is fastest in young animals and it tails off to zero in adulthood.

Population agents are assumed to grow according to a von Bertalanffy growth equation. The cohorts which are tracked within a Blastula agent are also assumed to grow according to a variant of the von Bertalanffy equation. This assumption must be kept in mind when considering whether these representations are appropriate for the species in question – in particular, Blastula are used as the principal means of managing juveniles before adulthood, this assumption can therefore also impinge on other class groups (such as Benthics, and Polyorganisms in general).

C.2.4 Mortality associated with local populations over capacity

This is represented as a mortality which is four times greater than the usual natural mortality in the population. This fairly arbitrary value was chosen to bring the population down to the carrying capacity quite quickly – the value was informed by consideration of some observed population recovery trajectories, but as these were for pinnipeds they may not be appropriate for all species.

C.2.5 Decay rates

Dead animals are arbitrarily assumed to decay at a rate of 0.01% of their body mass every 600 seconds. This rate represents the loss of body mass due to decomposition alone, and is probably not aggressive enough. The issue which was foremost when the rate was chosen was that much of the removal of dead material in the ocean is through scavenging, and premature removal of what may be an important source of food was not desirable. Since scavenging was not explicitly modelled (nor was detritus tracked), this rate was not calibrated further for the North West Shelf system.

C.2.6 Navigation

As a general rule, if an agent is attempting to move to a specific location (say to spawn), there is a degree of randomness with respect to exactly where the target is. When fish move toward a spawning bed, they initially select a location with an associated radius. Whenever they set their velocity (usually once a time-step) they direct themselves directly toward a point randomly located within the radius (though agents pursuing prey are not subject to this mechanism as they have a specific location to target). For reasons of computational efficiency, when simulated organisms are purposefully moving (seeking a particular location or target) on “gross” space and timescales (i.e. not finescale feeding interactions where exact spatial intersections are crucial) it is assumed that once they get within 10cm of their objective that this is close enough and can be considered to be coincident for the purpose of the movement. The basis for this assumption is that all of the relevant organisms in the model have a maximum speed which allows them to “correct” for errors in trajectory within one second’s movement.

C.2.7 Movement

Adult movement

When agents like fish or turtles are not actively seeking particular habitat, prey or avoiding something, they usually move using a correlated random-walk with a variable step-length. This is not strictly a correlated random-walk in the usual mathematical sense since the step-length is variable. This approach has been used in Lyne et al. (1994) for tuna (and other pelagic species), and the simulated tuna tracks compared well with the tracks of tagged tuna.

Movement while recruiting

When Blastula recruit a cohort to the adult population, it is assumed that the cohort is able to make its way from the juvenile habitat to the adult habitat with some (parameterised) level of loss. This clearly assumes that some path is available and that there are no insurmountable obstacles preventing the juveniles reaching the adult habitats. This would be inappropriate in cases where juvenile habitats may be intermittently (or permanently) closed (e.g. coastal lakes or dammed freshwater rivers etc). In those cases the explicit larval and Animal agents would need to be used or the recruiting site for the Blastula would have to be defined at the centre point of the blastula and movement to adult habitats attempted explicitly.

C.2.8 Eating range and perception range

Little data is available to quantify either the range at which an organism (like a shark) cab be considered to be able to take an incapacitating bite out of something. Similarly it is difficult to actually nail down a range at which we can generally say something is perceptible. These values are multiples of the maximum speed and cruising speed (respectively) of the organisms concerned. With the eating range, the linkage is reasonably clear, and the scalar is kept relatively small. For perception it is somewhat more complex, since the general mechanisms of perception vary both in kind and in range. Thus the decision to link the sensible area to the arclength an agent may explicitly cover within a given time was made in the interests of simplicity.

C.2.9 Flight range

The existence of a zone of discomfort and a flight zone in prey animals is well documented. The zone of discomfort is implicitly a part of the weighting of behavioural imperatives whenever a predator is present. The flight range of the simulated organisms was chosen with regard to the speed of the organisms and their usual predators to produce a distribution of success that matched available observations reported in the general ecological literature.

C.2.10 Contaminants

There are no sub-lethal effects modelled in *NWS-InVitro*. This is primarily due to a lack of data for the modelled organisms, apart from some data of limited application for larval stages (available from laboratory studies on very young larvae these data are insufficient to give insight of the effect of adults and juveniles in turbulently mixed

open environments). The lethal effects model primarily deals with chronic effects, although separating acute effects from this would be reasonably straightforward. The outfalls present multiple contaminants, and the assumption is made that the actions of the contaminants are response-additive (mathematically independent). This assumption is made since no data was found prior to the study that allowed for quantification of any interactions amongst the contaminants, and in some sense independent action is the "middle-ground".

For computational efficiency it has also been assumed that the plume makeup can be modelled by generating the contaminant concentrations using a plume index with the rate of contaminant flow at the outfall. The uptake curves and (for prawns) used to model the transport of contaminants into the tissue were calibrated using data from a paper by Hashmi et al. (2002), though there was insufficient data to test the calibration. The expulsion/metabolisation rates for contaminants are arbitrary and chosen to represent the gross character of the contaminant, this is again largely due to a lack of appropriate data.

Lastly for the contaminants, the mortality surface used to assign a level of mortality to a tissue load are based on LC-centiles from the literature; usually LC50 values for some nominated time period. It was necessary to make arbitrary guesses for LC100 values as they are not typically reported (or determined).

C.3 Assumptions for Benthic agents

C.3.1 Spawning and recruitment

Self-seeding and spreading was not sufficient to capture the observed recovery dynamics in the benthic community in the North West Shelf, so a constant recruitment term was incorporated. This term accounts for larvae which enter the study area from outside the region and was tuned to give the same temporal span for small benthos recovery (under moderate productivity trajectories) as observed empirically.

C.4 Fisheries operations

C.4.1 Commercial fisheries – spotter planes

The prawn fisheries in the North West Shelf region use spotter planes to locate boils near the surface. Modelling these planes was a crucial part of the fleet operations of the fishery. The planes are implemented as a Monitor agent, and are able to locate every prawn school in the indicated depth range. The assumption is that this perfect knowledge (which could readily degrade) does not introduce unreasonable efficiency within the fishery. This was considered to be acceptable given that the plane agents fly intermittently, and that their information quickly becomes outdated and matches the grade of information available in reality given the frequency of coverage of planes (or more typically these days with satellite images) in shallow water areas like Exmouth Gulf.

C.4.2 Recreational fisheries

The mortality attributed to recreational fishing is relatively novel, so the precise formulation is assumed to apply some reasonable level of mortality disaggregated spatially. More importantly, it is also assumed that catchability can be carried across for Population agents from the commercial fisheries (scaled to account for the different gear types used in the different sectors) for those species.

APPENDIX D: DECISION TREE FOR REGIONALLY COORDINATED MANAGEMENT

The following decision tree was used to implement the regionally coordinated (cross sector) management in *NWS-InVitro*. Refer to chapters on the EPA (chapter 19) and FMA (chapter 17) agents for how the tree was tied into the rest of the model framework.

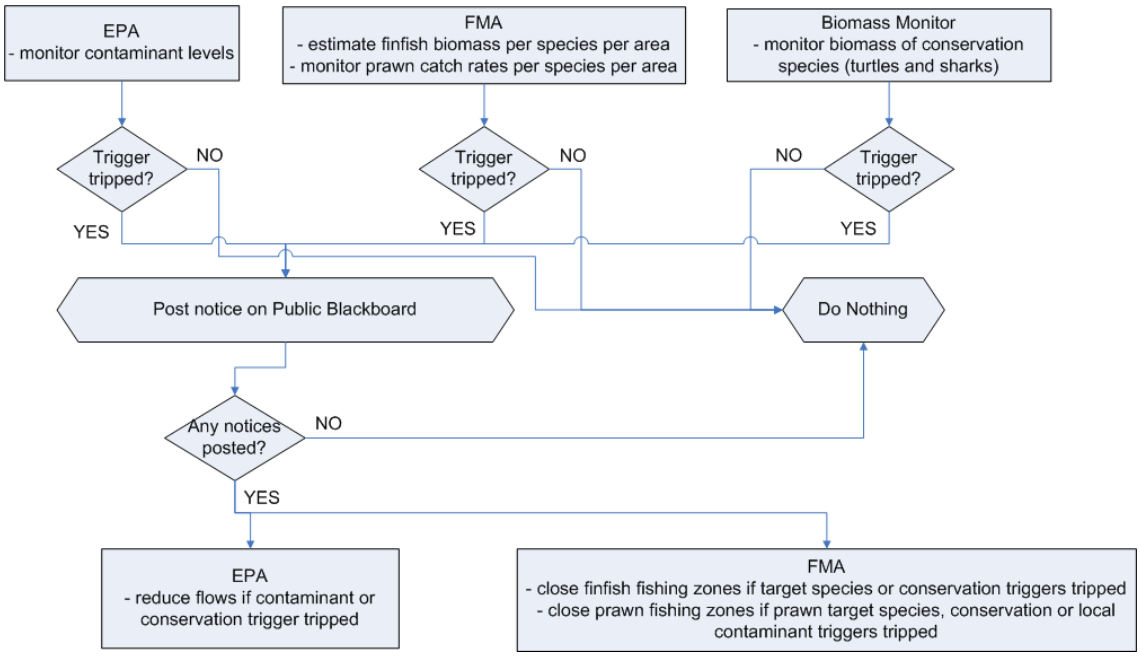

Figure D.1: Regionally coordinated management decision tree.

ACKNOWLEDGMENTS

The following people and agencies have contributed significantly to the Study through the provision of technical expertise and advice, and historical data and information. The Study partners gratefully acknowledge their contribution.

Western Australian State agencies
Department of Environment and Conservation (Department of Conservation and Land Management and Department of Environment)
Department of Fisheries
Department of Industry and Resources (Department of Mineral and Petroleum Resources)
Department of Land Information
Department for Planning and Infrastructure (Department of Transport)
Pilbara Tourism Association
Shire of Roebourne
Town of Port Hedland
Tourism Western Australia
Western Australian Land Information System
Western Australian Museum

Commonwealth agencies
Australian Institute of Marine Science
Geoscience Australia (formerly Australian Geological Survey Organisation)

Consultants
Cognito Consulting
David Gordon International Risk Consultants
METOCEAN Engineers (formerly Weather News International, Perth)
Oceanica (formerly DA Lord and Associates)

Industries
Australian Petroleum Production Exploration Association (APPEA)
Apache Energy
BHP Petroleum
Chevron Australia
Dampier Salt
Hamersley Iron
Mermaid Marine
Woodside Energy

Individuals
Clay Bryce
Graham Cobby
Nick D'Adamo
Mike Forde
David Gordon
Andrew Heyward
Barry Hutchins
Bryan Jenkins

Di Jones
Ian LeProvost
Ray Masini
Mike Moran
Steve Newman
Eric Paling
Kelly Pendoley
Bob Prinz
Chris Simpson
Shirley Slack-Smith
Di Walker

Reviewers
Malcolm Haddon

Editorial and publishing
Louise Bell – Graphics/cover design
Lea Crosswell – Webpage design
Rob McKenzie – Editor
Diana Reale – Webpage design
Linda Thomas – Editorial consultant/layout and design
Helen Webb – Editorial consultant/Project Manager

Front cover photos courtesy of:
Centre – Coral reef ecosystem, WA Museum, Clay Bryce
Aquaculture pearls, Department of Fisheries WA
Recreational fishing, Department of Fisheries WA, Jirri Lockman
Offshore petroleum platform, Woodside Energy Ltd
Commercial Fishing, Department of Fisheries WA
Tourism, CSIRO
Coastal development aerial photos, Hamersley Iron Pty Ltd