Vol. 154: 1-16, 1997 | MARINE ECOLOGY PROGRESS SERIES
Mar Ecol Prog Ser | Published July 31

A virtual mesocosm with artificial salps for exploring the conditions of swarm development in the pelagic tunicate *Salpa fusiformis*

Philippe Laval[*]

Observatoire océanologique, Station zoologique, Université Pierre et Marie Curie - CNRS URA 2077, BP 28, F-06234 Villefranche-sur-Mer cedex, France

ABSTRACT: A virtual mesocosm is a software framework simulating a controlled oceanic environment. Artificial salps and their food are introduced into the mesocosm where their development can be watched on the computer screen. The artificial salps are software entities mimicking closely the life-history and behavior of their real counterparts (the tunicate *Salpa fusiformis*): they swim in the artificial space, where each individual feeds on the local resources; they metabolize, accumulate reserves, and grow; their reproduction alternates between asexual and sexual generations. Plausible values are given to 21 parameters characterizing their life-cycle and metabolism. A virtual mesocosm is in fact a model, focused upon individuals and their spatial interactions. The relationships present in the conceptual model are more completely and more precisely stated in a software design than in a mathematical model. Many computer experiments are possible with the artificial salps, without the difficult problems associated with the sampling and laboratory maintenance of the real, fragile, planktonic organisms. An example is given showing the effect of different spacing between food patches.

KEY WORDS: Computer simulation · Artificial life · Conceptual model · Object-based design

INTRODUCTION

Salps are macroplanktonic marine tunicates that may become very abundant at certain periods. Their so-called blooms, or swarms, result from their ability to multiply asexually. These swarms can cover very large areas; it is estimated that they could clear 100% of the upper 100 m of the water column in a few days (Nishikawa et al. 1995).

In salps, the organism emerging from the egg, or oozooid, gives birth by asexual multiplication to a chain of identical individuals, or aggregates. Each aggregate in the chain reproduces sexually and emits an oozooid (see for example Alldredge & Madin 1982, for a description of the tunicate reproductive cycle). In this text, when a distinction is not necessary, I will call both oozooids and chains 'zooids'.

Modeling the development of tunicate swarms is worthwhile because they can have critical effects on the pelagic ecosystem. A model was attempted by Andersen & Nival (1986) for *Salpa fusiformis* Cuvier, a tunicate that forms large populations in the Mediterranean. These authors formulated a mathematical model of the population dynamics of *S. fusiformis*, using several differential equations. This model grossly reproduces the large population increase of the species following a phytoplankton bloom. A more detailed model, taking space, more physiological parameters, and individual histories into account, would be mathematically intractable.

In this article, I propose a different approach, making use of what I call a software model that is in essence a model, but not one based on mathematical equations. This approach consists of building entirely in software an artificial world populated with artificial creatures mimicking closely real salps. A convenient metaphor which will appeal to oceanographers is that of a virtual mesocosm. In biological oceanography, a mesocosm is a large enclosure where selected planktonic organisms may be sampled at intervals, while the

[*]E-mail: laval@ccrv.obs-vlfr.fr

external conditions are monitored. But mesocosms would be difficult to build for salps. A software model does not have the physical limitations of mesocosms.

METHODS

Software development. Building a mesocosm in software is a large endeavor. Writing software in this context means much more than simply coding algorithms in some computer language like FORTRAN. What is required is a working knowledge of software engineering (particularly object-based design). Because many objects in the design should execute concurrently (i.e. the creatures, the simulation clock), there are additional difficulties in comparison with the design of sequential programs.

To develop the software I followed a methodology in use in the aerospace industry, called HOOD, Hierarchical Object-Oriented Design (Delatte et al. 1993). HOOD prescribes a very formal set of rules, in order to ease project coordination by teams of programmers, which aim to arrive at automatic code generation from a graphical and textual description of the model. Being an individual programmer without industrial support, I am content to adhere closely to the spirit of the rules: I always strive to craft software 'objects' corresponding to ecological entities and with relationships having an ecological meaning. In a software design, it is easy to introduce artefactual objects which permit unnatural relationships with no logical significance to apparently 'work'. In a technical system, it is generally possible to determine if the software corresponds correctly to what has been designed. In an ecological model there is no such luxury, unless precise data are available, a very rare situation in plankton ecology.

In essence, the HOOD methodology consists of the decomposition, in a hierarchical manner, of the system to be modeled into software objects corresponding to the domain entities. The top level, or root, contains only 1 (abstract) object, which is the entire system. To build the next level, one has to find objects which, together, will provide the same functionalities as the root object. These 'child' objects are then decomposed in turn at the next level, and so on until 'terminal' objects, needing no further decomposition, are found. These steps should be performed very carefully, because a 'wrong' object, i.e. an object unnaturally aggregating functionalities pertaining to heterogeneous entities, will carry its bad design to the lower level(s), until it is discovered that it has become impossible to connect some other child objects which are needed to provide a service. The design at this level should then be rearranged in another way, with profound repercussions for consistency at higher levels.

It is possible to code a design in several languages, but Ada is the only one providing both concurrency and object-based programming. Moreover, Ada was designed from the onset to support software engineering principles, which makes it the language of choice to set up a sound design. HOOD was first aimed at an Ada translation (other languages, like C++, were later added for industrial reasons, but the mapping is not so natural). Ada is a strongly typed language (much more so than Pascal): checks made during a compilation are much more stringent than the ones made in FORTRAN or Pascal programs. They allow one to detect very subtle errors. An Ada compiler also controls the inter-module consistencies, a feature which greatly helps to verify the design of a program.

The software for the model referred to in this article was first named CALIFE (for Computational Artificial LIFE); I have stuck to this name, even if it now seems too general. It comprises, in its present form (version 8.0), more than 14000 lines of code — including utility routines and comments (about one third). Hallam et al. (1996) pointed out the difficulty in explaining individual-oriented simulation software to the readership of an ecological journal. I encountered similar problems for CALIFE. A mathematical model can be summarized by writing down its major equations and presenting the conceptual model. This is enough information to allow an appraisal by the scientific community. In contrast, to disclose a software model, the source code along with a detailed documentation should be available. The Ada 83 source code of CALIFE is stable enough and well documented in many sections, but it is still evolving. Before its re-design in Ada 95 (Intermetrics 1995), I intend to make it available in a copyrighted medium. Readers wishing to examine how some sections are coded may contact me directly. Like Hallam et al. (1996), I will only try in this article to describe the present state of the software in general terms, with few computer details.

Object-based modeling. Several recent software engineering textbooks explain how to build a detailed conceptual model of a domain, which can be translated to executable code (for example Booch 1991, Rumbaugh et al. 1991, Coleman et al. 1994). The same techniques used to design models of manufactured systems may benefit ecosystem models (Laval 1995a). The components of such software systems are software 'objects'. These objects often directly correspond to real objects in the domain, but may also represent some high-level abstractions. In a software design of an ecosystem, one finds environmental objects (for example, food) as well as objects corresponding to (artificial) organisms. Software objects have behavior, and thus may mimic the dynamics of real organisms in their environment. With this kind of model it is possible to go down to the individual level. Each individual object remembers its position in space, its local state, and behaves according to its current parameter values.

Of course a real ecosystem cannot be specified with only a few objects. However, it is possible to carefully select key components of the system. A hierarchical design method is very useful here (Müller 1992, Laval 1995a). Of course mathematical modelers building a conceptual model also strive to express key abstractions. But instead of dealing with theoretical relations (which frequently are approximations), the ecologist building a software model can work with more concrete representations and may more easily add new expertise in the domain (Coquillard & Hill 1997).

Object-oriented languages are tools of choice to develop this kind of model. In fact these languages seem so powerful that it is tempting to try to set up a generic object-oriented, individual-based ecosystem model, composed of ecologically significant building blocks, which could be tailored to particular ecological problems. Olson & Sequeira (1995) made such an attempt. Making a general object-oriented ecosystem model is the aim of several other authors (Baveco & Smeulders 1994, Hill et al. 1994, Maxwell & Costanza 1994, Wenzel 1994, Bindingnavle et al. 1995, Mooij & Boersma 1996). The potential of such a tool in ecology would be considerable. However, crafting a multiform and flexible tool will require a great amount of work, owing to the diversity of life-histories and ecological strategies of biological organisms (a point often underestimated by computer scientists). Even with these considerations in mind, it seems unlikely that the same system could be tailored, for example, for terrestrial vegetation studies and for plankton ecology.

To study the development of salp swarms, I wrote a specific software model aimed at the design of artificial salps in their environment. The experience gained from this first step will later be generalized to include other organisms. In this specific model, I combine my programming experience with my biological background on gelatinous macroplankton (Laval 1980, Laval et al. 1989, 1992).

SOFTWARE CHARACTERISTICS

Overview of the structure. Higher-level modules in the HOOD hierarchy represent the ecological environment, with the necessary artefacts to conduct a simulation. In the ecological environment a predominant role is played by a Space object. The Space object contains the resources in a Salp_Food object. The simulation context adds a global clock, a means to display the individuals on the computer screen, and counters to record the numbers of zooids created. The lower-level objects are the artificial organisms mimicking tunicates. In the computer implementation of the tunicate reproductive cycle, a chain (composed of several aggregates) is represented by only 1 object, because all the aggregates constituting the chain are clones attached together, and all have the same behavior, the same biological characteristics, and the same overall position.

Space. Space and time in an individual-based simulation obviously present scaling problems. In a proportional representation of the space occupied by each individual, only a very small spatial area may be displayed. On another hand, if some spatial scaling is implemented, the scaled-down spatial area may contain an unmanageable number of individuals. Scheffer et al. (1995) proposed using a super-individual concept to alleviate this problem. This is an ingenious solution to reduce calculations in a numerical model. However, it has the disadvantage of blurring the individual characteristics, each super-individual being only a mean individual of its category. In CALIFE, I wanted to retain the advantages of individual-based models, so I chose another way to compromise. I did not use super-individuals (except in a certain sense for chains, where each individual stands for several identical aggregates). Individual zooids stand for themselves; it is their behavior which is averaged over time periods.

Space in CALIFE is a 2-dimensional grid of 318×222 positions, representing a vertical slice of the ocean. To be able to estimate filtered volumes, each cell of the grid is considered as having a small extent in the third dimension (not seen in projection).

This Space object is made consistent with the time scale in the following manner. Each grid position represents the mean distance an individual can travel in a day (the time unit). With this convention, each occupied 'position' is in fact a mean position over a day spent in some volume of water (a cell). *Salpa fusiformis* has a mean swimming velocity of 1.3 to 6.6 cm s^{-1} (Bone & Trueman 1983). If moving from one grid position to the adjacent position represents a (mean) day of swimming, the horizontal distance between 2 grid positions would represent about 3400 m. However this species, whether solitary individuals or chains, does not swim in a straight line (and probably not at constant speed). Thus its mean daily transference in x-projection may be much less than 3400 m. A rather arbitrary, but convenient, value of 1000 m will be retained as the horizontal spatial unit.

In the vertical, it is important to represent the first 100 m, where blooms usually occur. Therefore, the 222 positions were each made 0.5 m high. The virtual third dimension can be assigned a thickness of 0.5 m. The grid thus corresponds to 318×222 cells of size $1000 \times 0.5 \times 0.5$ m.

A comparison with field data shows that a 250 m^3 cell volume is not unrealistic. The density of oozooids in the sea depends widely on the time of the year. I have an unpublished series of samples obtained in the morning between January 1, 1963 and May 5, 1964 in the Bay of Villefranche (Mediterranean Sea) at a depth of 30 m with a 1 m plankton net (filtering about 300 m^3). Of these samples, 156 were negative for *Salpa fusiformis* oozooids. During bloom periods, the density of oozooids varied from 1 to 45 per sample (mean 5.4, n = 26). With the impossibility of representing the real numbers of individuals, the compromise of considering a mean individual filtering a mean daily volume of 250 m^3 does not scale up too badly.

Time. CALIFE is a concurrent software where each artificial individual executes its own actions autonomously. Often parallelism is only introduced as a means of taking advantage of a multiprocessor architecture to reduce computation time (see, for example, Foster 1995). However, even on a monoprocessor, concurrency may also be used to implement conceptual parallelism. With conceptual parallelism, processes which are logically concurrent are programmed using modules which have their own threads of control (on a monoprocessor, the threads are interleaved by slicing the processor time). This is possible in Ada, where these modules are called tasks. Conceptual parallelism results in programs which are easier to understand. Ada programs may also implement other forms of parallelism which, on a multiprocessor, benefit from an increase in speed.

In CALIFE, each individual is driven by an internal clock, with a mean 'daily' period. With a 90 MHz CPU monoprocessor, 1 d of a tunicate life in fact lasts for 1.5 s of computer time. For each individual the actual computation time is on the order of 2 ms. As there may be hundreds of individuals executing concurrently on the computer's processor, during the idle time the control is given (by the Ada runtime) to the other individuals. Within 1.5 s all the individuals will have carried out their daily computations. A shorter 'day' duration cannot be safely used with the hardware available.

There is also a global clock, which is only used to set the duration of the simulation, and is only consulted by the individuals (in the same way that real creatures look at the sun). This astronomical clock is a scaled down version of the system clock, with a 1.5 s period. This global clock is graduated in mean days, and its time is displayed on screen to reflect the progression of the simulation.

Real-time visualization. When simulating organizational or manufacturing systems, animation is an effective means of showing the results of simulations (Verbraeck & de Vreede 1993). In CALIFE, the final counts at the end of the program could be obtained without displaying the individuals. However, showing on-screen what is happening has 2 distinct advantages. The first is related to program design. Mapping the Space object to the computer display makes readily apparent that space should be designed as a resource shared by all the individuals. Second, displaying the individuals is invaluable in program development, because concurrency is notoriously the source of hard-to-find bugs in software. Displaying individuals is not without problems. Individuals are represented by small colored dots 2×2 pixels wide. There are 10 different color codes distinguishing oozooids and chains as live feeding or non-feeding and as mature or non-mature, and also oozooid and chain cadavers. Even on large monitor displays, their different codes are difficult to distinguish. For the software development, another version of CALIFE was derived in which categories of individuals are represented with different text characters. Using the screen in character mode limits the spatial grid to a dimension of 78×26, but the different kinds of individuals are clearly apparent. This character mode version cannot be used for realistic simulations, because the border of the spatial grid (which acts as a reflecting boundary) is encountered by the outer individuals before the end of a simulation. It was nonetheless very useful for program development. In pixel mode, the outer individuals in a typical simulation (less than 200 'days') do not have time to reach the grid limits, so the space may be considered unlimited.

THE ARTIFICIAL ORGANISMS

Structure and dynamics. Each artificial salp is implemented with an Ada record with a component containing an access value (a pointer) to a task (Laval 1995b). The other components store the individual's birth date, weight, actual position, etc. The task allocated to an individual starts executing when the individual is created. It runs in a timed loop, simulating the individual's biological clock with each period corresponding to an average day spent by the organism.

During each clock cycle, the individual moves, feeds, metabolizes, and grows. This is accomplished by calling corresponding procedures. After a certain time, if its reserve level is sufficient, the individual reproduces. This is accomplished by allocating a new location in the computer memory to another individual, and linking this new individual to its parent via an access value. This creates a linked list of individuals starting from the first individual. Oozooids produce 1 or more generations of chains; chains produce the number of oozooids corresponding to their constituent numbers of aggregates (Laval (1995b). The linked list is thus a multibranched tree, alternating oozooids and chains.

Because the tasks belonging to the individuals are independent processes, at a given moment 2 individuals having different birth dates may perform different actions. The repertoire of these actions is outlined in Table 1, using pseudocode. Reproduction and movements constitute large amounts of the code.

Reproduction. With their asexual multiplication phase, the number of tunicates can grow explosively. One oozooid gives birth to a chain composed of a number NA of aggregates, which a generation later gives NA oozooids. Therefore, in the absence of mortality (an unrealistic hypothesis), 1 oozooid would give NA^n oozooids after n generations. For *Salpa fusiformis*, NA is about 125; an oozooid produces a chain in 8 d, and a chain emits its NA oozooids in 14 d (Braconnot et al. 1988). After 4 generations (about 88 d), 1 oozooid would have the potential to produce 2.4×10^8 oozooids. Moreover, a chain emits several generations of chains (a number NG of generations, up to 4). Thus NA oozooids are produced NG times at 3 d intervals, leading to still potentially higher numbers. Of course such numbers are not reached in the ocean, because there is not enough food, so that many chains do not reproduce, or they produce fewer oozooids. Salp swarms are nevertheless prominent events in the open sea (see for example Alldredge & Madin 1982,

Table 1. Algorithmic structure underlying the behavior of an artificial tunicate (simplified). The number in square brackets following an ellipsis (...) indicates how many Ada statements make up the action. [Ooz/Ch] means that specific code exists for both the oozooid and the chain generation

```
[Ooz/Ch] Ada 'rendezvous' with the parent (transmission of the pointer on the child structure)
    ... [7]
Initializations
    ... [13]
Wait the duration of the embryonic phase
Find an empty position close to the parent (abort if not possible)
    ... [58]
Main_Loop : internal cycle
    [Ooz/Ch] if conditions to pass to the Mature stage are fulfilled, do so; if longevity is exceeded, change the appearance to
    Cadaver
        ... [23]
    if Cadaver then
        if inside area limits then go down
            ... [23]
        else continue until complete decomposition (no display)
        end if (inside frame)
        [Ooz/Ch] if Cadaver 'longevity' is exceeded (decomposition) then disappear (and exit Main_Loop)
            ... [27]
    else if Live then
        [Ooz/Ch] Determine the food requirements
            ... [13]
        Try to move to an empty position (x, y)
            ... [193]
        if there is food at (x, y) then subtract up to the food requirements
            ... [3]
        [Ooz/Ch] if food obtained then grow (increase the current weight — only if not Mature, and increase the reserves)
            ... [33]
        [Ooz/Ch] Decrease the reserves by the amount corresponding to metabolism
            ... [20]
        if the reserves go below the Reserve_Min value, make the age equal to the longevity (will cause death at the next cycle)
        Every 4 moves, change randomly the swimming direction
            ... [27 + library package]
        [Ooz/Ch] if conditions (age, reserves, etc.) allow, reproduce (this decreases the reserves)
            ... [55]
    end if (Live)
    Compute the time remaining until the next cycle, and wait for this duration
    (if not enough time available, raise exception Timing_Error)
        ... [11]
    Increment the age
end loop Main_Loop
```

Laval 1995a). In the software, reproduction follows the following algorithm:

```
Each oozooid produces up to NG generations of
chains until it dies.
The size S (number of aggregates) of the first chain
is initialized to NA.
For Generation_Index = 1 to NG loop
    Emit 1 chain
        For Oozooid_Index = 1 to S loop
            Produce 1 oozooid. Each oozooid grows
            and resets S (≤NA) according to the
            amount of reserves (which depends on
            the food encountered)
        end loop;
    Continue if there are enough reserves
end loop;
```

It is not possible to simulate in the CALIFE software the multiplication of chains with 125 aggregates. Machine considerations (memory size, concurrency on a monoprocessor) limit NA to a small number, not exceeding 5 or 6. This small number is nevertheless sufficient, owing to the exponential growth, to produce peaks of 600 to 800 individuals living concurrently. With this number it is possible to simulate the multiple spatial interactions existing in a tunicate swarm.

This scaling of the number of individuals should be unfolded to interpret the numbers of individuals produced by the software. If the number of individuals constituting a chain is artificially limited to 5 (for example), it may be considered that the number of individuals produced by the CALIFE model is scaled down by a factor 25. That is, chains of 5 aggregates in the software correspond to chains of 125 aggregates in the ocean.

Because these numbers grow exponentially, logarithms must be used to compare the numbers of individuals created in a simulation with the numbers found in the ocean. The proportionality factor, however, may be much less than the theoretical value of 25 (for NA = 5), because of the unknown mortality in the field. Moreover, sampling problems make such comparisons problematic.

The oozooid of *Salpa fusiformis* emits the first chain when the stolon has differentiated enough. This differentiation time depends on the food available and its quality, and on physical factors like temperature. In the software these conditions are translated into the following tests: the first chain is emitted only if 3 conditions are fulfilled:

(1) the age of the oozooid is greater or equal to the minimum for the first chain emission;

(2) the weight has reached the minimum adult weight; and

(3) the reserves are greater or equal to the minimum reserves necessary for reproduction.

For the subsequent chains the conditions are the same, but with a minimum age augmented by the interval between chains multiplied by the number of chains already emitted. Comparable conditions are set on chains, this time for the oozooid emission by each aggregate.

Movements. After a 'day' spent at a grid position, each individual moves to another position. This position is the next one along in the present swimming direction, unless the position is occupied (by another individual, or by an obstacle like the surface). In this case the zooid searches the nearest free position by changing its present direction, as described in Laval (1996). The actual direction is also changed randomly if the reserves of the individual stay below one fourth of the adult weight (an arbitrary quantity) during 4 successive moves. This was necessary to increase the spatial dispersion of the bloom. Otherwise the food, which is not renewed, rapidly becomes exhausted in the area. The algorithm for random changes is programmed using a machine-independent pseudorandom generator (Harmon & Baker 1988).

During reproduction, the oozooids issued from a chain must each be emitted at an empty position next to the parent chain. This is done in mutual exclusion so that a position cannot be occupied by a nearby individual while the oozooids are emitted. In order to distribute the offspring around the parent, each emitted oozooid tries consecutive swimming directions until a free position is found.

In a rectangular grid there are 8 possible directions. I experimented with this number, but I eventually suppressed the 4 diagonal directions to keep the space homogeneous in each direction (see 'Food distribution' below).

Growth. Coding a growth function revealed unexpected difficulties. Classical growth functions from the literature are too general to be translated to an individual basis. They express length (only) as a function of time, after fitting a general equation to a sample of individuals of different sizes; the resulting equation only summarizes the growth of an idealized mean individual. This is of little help in CALIFE, where it is necessary to compute the new weight of a given individual eating some amount of food.

The dynamic nature of the software makes it necessary to introduce the notion of reserves. In the newborn individual, these reserves are localized in a special organ, the placenta (see Bone et al. 1985). They allow the newborn individual to live when it is not yet able to filter food. Later on, growth and production of new individuals are so intense that the food assimilated is rapidly invested in production of new tissues. The notion of reserves is then a convenient abstraction for managing a temporary storage for the food assimilated.

The growth routine has thus to compute a new weight and a new level of reserves. At the individual level, the production of new tissues is the result of a balance between the amount of food assimilated (anabolism) and the amount of reserves consumed (catabolism) (von Bertalanffy 1964). In an individual-based software, it is possible to go further: the notion that the reserves act as a 'buffer' can be implemented at the level of each individual. If food is lacking, or simply insufficient, new tissues can be built by decreasing the reserves (down to some point). The reserves are at each cycle decreased by metabolism. For each zooid emitted, reproduction also decreases the reserves by a fraction of the adult weight.

Measurements shown in Madin & Deibel (1997) make clear that, in every species of salps, growth increments diminish with age. In a mature salp, growth becomes null and the food assimilated is used for metabolism and reproduction. In CALIFE, an individual is mature when it has reached the minimum age for reproduction and its weight is greater or equal to the minimum adult weight. The conjunction of these 2 conditions ensures that small immature individuals, which have not found enough food to reach a critical weight, do not become reproductively mature.

Because of the very different adult weights of oozooids and aggregates, different growth coefficients are needed for these 2 categories. These coefficients are not classical allometric coefficients (which, as we have seen, do not apply to individual growth), but ad hoc coefficients used to exponentially decrease the assimilation efficiency of the growing individual. They are named T_Ooz_Growth_Coeff (for the oozooid) and T_Bl_Growth_Coeff (for the aggregate, or blastozooid).

The growth routine works as follows. FR, the food requested (in computer jargon, a client 'requests' something from a server) by a zooid arriving at a given location, is equal to its current weight W × T_Food_Coeff. The zooid obtains FO $\leq$ FR (possibly 0.0), according to the amount available at the location. If the zooid is not mature:

(1) The Food Intake, FO, is transformed to Food Assimilated, FA = FO × T_Food_Conversion_Coeff.

(2) An Age Index is computed, which is simply the current age relative to the age at maturity (giving 0.0 at birth and 1.0 at maturity). A Weight Index is computed similarly for the current weight. These 2 indexes are averaged, giving a Growth Index GI (constrained to $GI \leq 1.0$).

(3) The proportion of assimilated food which produces new tissues is computed using GI so that it decreases as age and weight increase. This gives the actual growth increment: $W_{INC} = FA \times (1 - e^{-k \times GI})$, with k = either T_Ooz_Growth_Coeff or T_Bl_Growth_Coeff.

(4) If no or little food is found in the environment, the above formula would give a zero weight increment. A real salp may still grow in these conditions, owing to the use of its reserves. To account for the reserves, a potential growth increment P_{INC} is computed according to the same formula, with FR instead of FO. P_{INC} represents the growth which would be achieved if the food requested was obtained.

(5) If $W_{INC} < P_{INC}$, the missing amount $P_{INC} - W_{INC}$ is subtracted, if possible, from the current reserves. Otherwise the current weight is increased by W_{INC}, and the difference $FA - P_{INC}$ is added to the current reserves.

If the zooid is mature, its weight does not change, and 20% (an arbitrary percentage) of FA is added to the reserves.

Experiments with the model using different food conditions showed that the growth in weight of the artificial oozooids and aggregates did not reach unrealistic levels.

Metabolism. Cetta et al. (1986) measured the oxygen consumption of *Salpa fusiformis* individuals of different weights; the relation seems to be linear on a log-log scale, but important variability is apparent. To provide a metabolic function for an individual in CALIFE, it is necessary to know precisely how metabolism changes with age in one individual. In the software, the daily metabolism of oozooids and aggregates simply consumes a proportion of the reserves. Metabolism includes excretion, which is not coded as a separate function.

Death and decomposition. An individual dies either when its natural longevity is passed, or if its metabolism consumes more than its reserves. The death date is stored in a reserved field of the record, but the internal clock still loops during a decomposition duration (no actions are of course accomplished within the loop, except downwards passive movements to simulate the sinking of cadavers). When the decomposition duration is over, the individual disappears. The production of cadavers was implemented in the software because it is an important parameter for ocean flux studies.

PARAMETERIZATION

Reduction of the variability. To demonstrate the resultant effects of various parameter values, very little 'biological' variability has been introduced in the software. For example, the parameters setting the durations of the different phases of the development are only maximum values. The actual length of these phases may be shortened because of death by starvation, or prolonged if maturity is delayed because the adult weight is not yet reached. It would have been easy to add normally distributed stochastic delays during each developmental phase, to mimic more closely the variability found in nature. But this variability would have obscured the basic functioning of the program when everything should be thoroughly checked.

Table 2. Parameters used in the software. Values in the right column are those used in typical simulations (Figs. 3 to 5). They are based on the available data for *Salpa fusiformis*, or else are the best guesses from several experiments with the software

Simulation parameters	
1. Period of the zooids internal clock: duration, in seconds, corresponding to 1 cycle of a zooid (= 1 average 'day')	1.5
2. Duration of the simulation (in cycles or 'days')	
3. First zooid position: line (2 .. 27) in character mode or (2 .. 223) in pixel mode	112
4. First zooid position: col. (2 .. 79) in character mode or (2 .. 319) in pixel mode	160
5. First zooid swimming direction: (N, E, S, or W)	N
Life cycle parameters	
For oozooids and chains:	
6. T_Embryonic_Delay: duration of the phase between the zooid birth and its first move. If (7) = yes, this duration (in days) will be random and (uniformly) distributed between 0.0 and T_Embryonic_Delay; otherwise its (constant) value will be T_Embryonic_Delay	0.2
7. Random_Delay (yes or no)	No
For oozooids:	
8. Number of chains emitted during the life	2
9. Number of aggregates (= number of individuals in a chain)	5
10. First chain emission, duration of blastogenesis (d)	8.0
11. Between-chains interval, for subsequent chain emissions (d)	3.0
12. Maximum age, i.e. oozooid longevity (d)	18.0
13. Duration of a cadaver before total decomposition (d)	6.0
For an individual in a chain (aggregate):	
14. Oozooid emission, duration of oogenesis (d)	12.0
15. Maximum age, aggregate longevity (d)	20.0
16. Duration of a cadaver before total decomposition (d)	6.0
Food and metabolic parameters	
17. Size of square food patches or 'pavements' (0, 1, 2 or 4). If (17) = 0, food will be present at each position (except an empty centering area around the frame). If (17) > 0, food will be distributed in a (centered) checkerboard pattern, with full pavements of size (17) alternating with empty pavements. Whatever the value of (17), a central area of size 8 × 8 contains food in every position	Variable
18. Food value at each full position (µg-at. N, 0.0 .. 2000.0)	Variable
19. Scaling factor, F_Unit, for output of food values	1.0
20. T_Ooz_Adult_Weight: dry weight of an adult oozooid (in µg-at. N)	52.0
21. T_Bl_Adult_Weight: dry weight of an adult aggregate (in µg-at. N)	26.3
22. T_Ooz_Initial_Weight: dry weight of a newborn oozooid when it leaves its parent chain (in µg-at. N)	3.0
23. T_Bl_Initial_Weight: dry weight of a newborn aggregate when it is emitted by the oozooid (in µg-at. N)	1.5
24. T_Ooz_Growth_Coeff: an empirically determined growth coefficient for the oozooid	1.70
25. T_Bl_Growth_Coeff: an empirically determined growth coefficient for the aggregate	0.80
26. T_Reprod_Cost_Coeff: proportion of (20) or (21) subtracted from the reserves during reproduction	0.20
27. T_Res_Min_Coeff: proportion of (22) or (23) corresponding to the reserves at birth. The minimum level of reserves will be set to minus the amount of reserves at birth	5.00
28. T_Food_Conversion_Coeff: proportion of the daily ration transformed into reserves	0.60
29. T_Metabolic_Coeff: proportion of the reserves consumed each day by the metabolism	0.06
30. T_Food_Coeff: proportion of the current weight, which, if food is non limiting, but non clogging, could be filtered in 1 d; may be > 1.0	1.60
31. Constant_Food: if yes any food removal by a zooid is replaced in the environment	Yes/No

Suppressing most of the variability has an undesirable effect; because many individuals move at the same time, the scheduler (the element of the runtime system responsible for the management of concurrency on a monoprocessor) is sometimes overwhelmed, temporarily halting some individuals when they should have moved. To obviate this effect, a small stochastic delay may be introduced just after the birth of every individual (parameter 6, Table 2). This is not unnatural because not all offspring are emitted at the same time during reproduction. This very slight desynchronization of each individual at birth increasingly separates the generations of zooids as the simulation progresses. Moreover, competition is slightly lowered, because there are fewer individuals trying to occupy the same area quasi-simultaneously.

Biological and life-cycle parameters. The life cycle stage durations of *Salpa fusiformis* (illustrated in Fig. 1, and shown in Table 2) may be inferred from laboratory rearing. Some caution should be exercised because salps are very fragile creatures, difficult to keep in an aquarium: even very small perturbations may prevent the deployment of their feeding mucous net (Madin 1974). The durations established in Braconnot et al. (1988), taking the median when a range is indicated, are used to choose basic values. Shorter or longer durations obviously accelerate or delay the development of the swarm.

Physiological parameters. A consistent unit is used throughout the software, from the food found in the environment to the amounts of reserves in the organisms. The units of measurement found in the literature often differ between authors. They were converted to a common unit, microgram-atom of nitrogen (µg-at. N). The C:N ratios used in the conversions are the ones established for *Salpa fusiformis* by Cetta et al. (1986).

The names and initial values of the parameters are shown in Table 2. These values represent the most likely values resulting from a great number of assays with the software. They may be considered as starting points to refine a particular setting of a parameter, to test a new hypothesis, or to re-evaluate another parameter in the light of new experimental evidence.

Initial and adult weights of oozooids and aggregates (parameters 20 to 23) are values from Andersen & Nival (1986), after conversion to µg-at. N. These authors give only weight classes. For the initial weights, I took the lower limit of their class I; adult weights correspond to the mean value of the limit between their classes IV and V. However, the lower limit of class I for the aggregate appeared far too low; using their data, the ratio adult-weight:initial-weight would be about 17:1 for the oozooid, and 239:1 for the aggregate. The initial weight of the aggregate was corrected to 1.5 µg-at. N to give roughly the same proportion to the adult weight as for the oozooid.

Parameter 26, T_Reprod_Cost_Coeff, expresses the metabolic cost of reproduction. The tentative value chosen, 20 % of body N, is comparable to that found by Madin & Purcell (1992) for *Cyclosalpa bakeri*. Increasing this value would first lead to chains with fewer aggregates and then may suppress the production of chains, depending of the current level of reserves.

Parameter 27, T_Res_Min_Coeff, determines the amount of reserves present at birth. An initial value of 5.0 (5 times the initial weight — not including the reserves) was found adequate to represent these reserves. This parameter is also used to manage starvation, without introducing a new parameter. If there is no food in the environment, the organism first consumes its reserves, which may eventually become 0.0. Rather than declaring that the individual is then dead, the reserves are allowed to become negative, up to minus T_Res_Min_Coeff (which implies that they still may be replenished if some food is again found). If this point is passed, the individual dies.

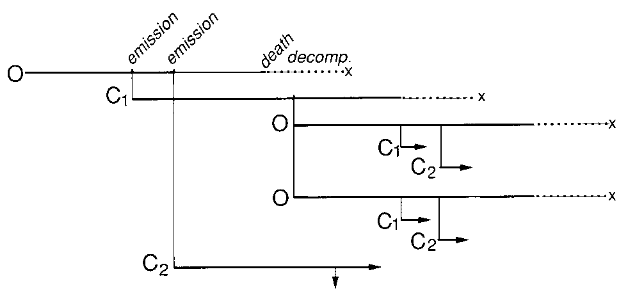

Fig. 1. Life-cycles of artificial tunicates. The horizontal direction represents time. Each horizontal line segment represents a specific process associated with an individual. The diagram shows an oozooid O which emits only 2 generations of chains, C_1 and C_2. Each chain, composed of only 2 aggregates, emit 2 oozooids (this is shown only for C_1). Real *Salpa fusiformis* emit up to 4 generations of chains, each composed of about 125 aggregates. The lengths of the different line segments are proportional to the durations given in Table 2

Parameter 28, T_Food_Conversion_Coeff, is an assimilation rate, the proportion of ingested N which is transformed into reserves. This ingested N is proportional to the current weight. Andersen (1986) found that the assimilation efficiency of *Salpa fusiformis* depends upon the nature of food, and is very variable, from 28 to 81 %. A value of 0.60, intermediate between the mean values found for diatoms and flagellates, was tentatively set for this parameter.

Parameter 29, T_Metabolic_Coeff, is the proportion of the N weight which is consumed daily by metabolism. In the absence of an experimental value for *Salpa fusiformis*, it was set to 0.06, the value found for *Cyclosalpa bakeri* by Madin & Purcell (1992). Experiments showed that it is a very sensitive parameter.

Parameter 30, T_Food_Coeff, is an ingestion coefficient, the proportion of the current nitrogen weight filtered daily. The value 1.60, the mean value for oozooids and aggregates, was estimated from the ingestion rates given in Andersen (1985), after conversion to µg-at. N.

Food distribution. In this version of the software, food is initially distributed in space either uniformly, or in a checkerboard pattern with alternating full and empty patches. The geometrical checkerboard pattern was chosen because it allows an easier assessment of the possible values of the parameters. For calibration, a random distribution of food patches of various sizes would give too much importance to factors such as the distance from the first created oozooid to the nearest patch, the probability of any zooid finding a patch, or the patch size. For the same reason, food is not renewed during the simulation, and it has no internal dynamics. The initial amount of food can only be diminished by the zooids.

The diet of salps consists of living or detrital particles in the range 1.0 µm to 1.0 mm. However, particles smaller than 0.2 µm are retained with low efficiency (Madin & Kremer 1995). It is difficult to set a realistic value for the food amount in the software, because the exact diet of *Salpa fusiformis* is not known. Food in the software is expressed in µg-at. N. In spring in the Ligurian Sea (Mediterranean), an averaged integrated value (on the 0 to 75 m column) of 23.3 mg of chlorophyll *a* per m^2 is observed (Bustillos-Guzmán et al. 1995). This corresponds to about 5000 µg-at. N in 250 m^3 (the volume of a grid cell). This value should be divided by 25 to account for the scaling factor for chains, giving 200 µg-at. N for parameter 18 in the software.

Given the physiological and life-cycle parameter values in Table 2, is this amount of food sufficient to result in a healthy production of artificial salps, say, an amount resulting in each oozooid emitting 2 chains per generation, each with 5 aggregates (corresponding to 250 aggregates in the sea) per chain for most of the zooids? Experiments performed by slightly changing this food amount have shown that with the 2 × 2 checkerboard pattern, 200 µg-at. N is too high (the computer memory is exhausted before 200 d), and with the 4 × 4 pattern it is too low (the population is extinct before 200 d). A medium food level, giving a moderate swarm development, corresponds to 170 µg-at. N for the 2 × 2 pattern, and 225 µg-at. N for the 4 × 4 pattern.

STOCHASTICITY OF THE MODEL

If 2 individuals head simultaneously for the same position, the rules built in the Ada language lead to the arbitrary selection of 1 of them. This is a fundamental requirement of concurrent systems (Agha 1990, p. 132), but it introduces indeterminacy in the program. This indeterminacy is of course increased by the random changes in direction of the individuals. However, even if changes in swimming direction are made deterministic (for example by not using a random sequence and making each individual turn clockwise every n moves), the indeterminacy due to concurrency alone is sufficient to make the location reached after a certain time by a given individual unpredictable. If the simulation is repeated with identical values of the parameters, the detailed distribution of the individuals in space will not be identical, because after repeated competition for positions the individuals' trajectories differ. Also, the individual obtaining a position may be a near-mature one with a lot of reserves, or a young one that has not found much food previously.

This variability of individual paths is the only source of variability in the model: all individuals of the same kind (oozooids or chains) are created equal, with the same initial weight and amount of reserves. Of course a random normal distribution (or any other statistical distribution) of initial weights and reserves could be established. But the software model needs first to be calibrated, and this added variability would obscure the influence of the other parameters. This natural variability may be introduced later. With fixed initial weights and reserves, all the variability is a consequence of the food distribution in space.

Even if the only source of variability is the random encountering of food, it would be difficult to verify and understand the basic model. Every modification of the level of a parameter would need to be replicated a large number of times in order to assess its corresponding variability. This is not very practical for software in which time is an incompressible number of seconds elapsed. I eventually found an artifice to optionally suppress the variability of the results, while retaining the variability of the paths of the individuals in space. If the 'Constant Food' option (parameter 31, Table 2) is selected, the amount of food an individual has removed for its daily ration is automatically replaced in the environment. If 2 individuals occupy the same position in quick succession, they each leave the position with their ration, but the food remains artificially at the original level. The Constant Food option initially distributes the food evenly over all the area, so that the individuals find their daily ration every time they move, irrespective of the actual positions they pass through. The unavoidable variability of individual paths is still present, but it has no effect on the food obtained. The same amounts of offspring are produced, and the evolution of the population in time does not depend on food availability. This Constant Food option is useful to verify that the program operates as specified, and to get the appropriate responses to parameter changes without stochasticity.

CALIBRATION

The calibration of this kind of model presents the same difficulties as do mathematical models with many parameters. A reasonable range can be established or guessed for some key parameters (by inferences from rearing experiments, data reported in the literature, or some deductions). For the other parameters, it is only possible to try some values and see what happens in the simulation.

To determine a plausible value for each parameter, the Constant Food option of the software is used first, with the Constant Food level set in excess of the daily ration (except for testing the food level itself). This gives an upper limit for each parameter.

For the number of aggregates in a chain, and the number of chains per generation emitted by an oozooid, the Constant Food option allows us to check

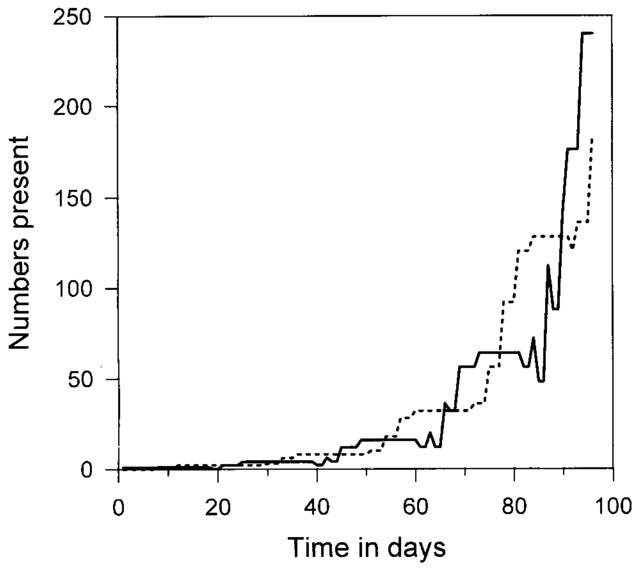

Fig. 2. Unlimited growth of the artificial tunicate population when food is replaced as soon as it is eaten (Constant Food option set to Yes). Parameters of Table 2, except that each chain emits only 2 oozooids, and each oozooid produces only 2 chains per generation; food is present everywhere with 170 µg-at. N in each position. The simulation ends after 96 d because the computer memory is exhausted. 'Days' are simulated days, each lasting 1.5 s of computer time. (——) Number of live oozooids; (---) numbers of live chains present each day

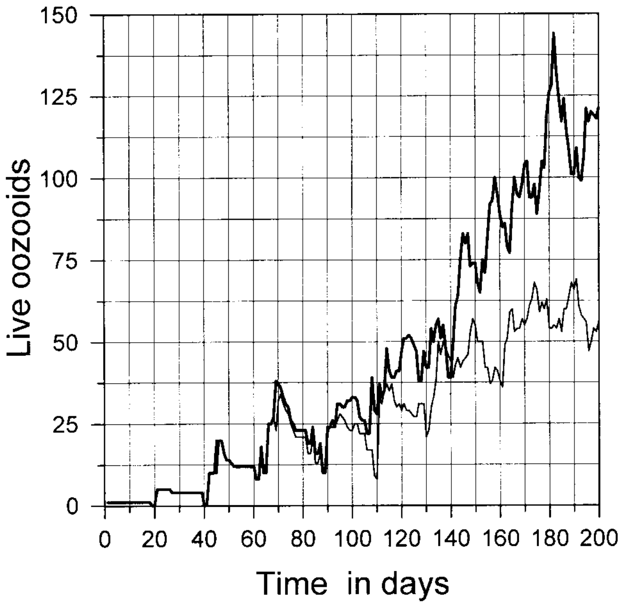

Fig. 3. Results produced when food is not replaced. Food is initially distributed in a checkerboard pattern (full patches of size 2 × 2 positions alternating with empty patches of the same size), with an initial value of 170 µg-at. N of food at each full position. Parameter values of Table 2. Two 200 d simulations made with exactly the same parameters are superimposed on the graph. Only the oozooid counts are shown

easily the algorithm given above for the number of oozooids produced. The emission of more than 1 chain per oozooid generation has the effect of superimposing a new exponential curve every time a chain is emitted. Fig. 2 shows the exponential growth of a population when there are 2 chains emitted by the oozooid, with only 2 aggregates per chain, and an unlimited food supply. Plateaus and dips are due to the asynchrony of births and deaths of the individuals.

In the normal situation (when Constant Food = no), the curves depart considerably from this theoretical population growth: food varies in space, so the reserves of a zooid may at times be so low that reproduction (asexual or sexual) of the zooid does not occur. If chains are emitted, they may be composed of fewer aggregates than the specific maximum (parameter 9, Table 2). The number of chains emitted by the oozooid may also be less than the maximum possible. The resulting variability thus reflects not only the basic alternation of zooid generations, but also the fortuitous encounters with food patches. Even if the food patches have strictly the same position and contents, different simulations with identical parameters still give different numbers of zooids produced during a bloom (Fig. 3). This is due to the indeterminacy of the paths followed by the zooids.

The variability of the simulation runs was assessed by doing 20 simulations with identical parameter values (Table 2), for patches of size 2 × 2 containing 170 µg-at. N at each non-empty position. After 170 d, the numbers of zooids produced, and the amount of food consumed, show rather large variations (Table 3). Again, since no individual variability of the zooid characteristics (initial weight, metabolic parameters) is introduced in the software, only the paths followed by the individuals in space differ between runs.

In Fig. 3 another aspect of the variability is apparent. Because the zooids must reach a minimum weight and a minimum amount of reserves to reproduce, they do not reach this condition after exactly the same duration. This difference in timing accumulates over the generations. In Fig. 3, the 2 curves (corresponding to initial oozooids created at time 0 in 2 different simulations) are desynchronized after about 150 d, despite the fact that both simulations have identical parameters and food distributions.

The sensitivity of the model to small changes of the metabolic parameters indicates that this is an area where the functions coded in the model would need to be improved. Real salps are adapted to a range of conditions around optimal values, so that physiological parameters should behave correctly within tolerance limits. More laboratory experiments should be done to find these limits, but to keep salps in a large tank poses many practical problems.

Table 3. Variability of 20 runs made with identical initial parameters. Parameter values shown in Table 2, with parameter 17 = 2 (food patches of size 2 × 2) and parameter 18 = 170.0 (µg-at. N). Statistics are computed for the cumulative numbers of zooids created, and the percentage of food consumed after 170 d

	No. of oozooids	No. of oozooid cadavers	No. of chains	No. of chain cadavers	% of food consumed
Mean	415.9	324.4	333.4	246.2	4.914
Standard deviation	41.305	30.699	30.983	21.796	0.4031
Minimum	353	266	276	204	4.193
Maximum	503	369	413	278	5.637

RESULTS

Spatial heterogeneity of food

With the various parameters set to plausible values, the reproductive rhythm of the artificial salps produces a swarm which spreads in space (Figs. 4 & 5). As every individual moves autonomously in parallel with the other individuals, screenshots made at different time steps would not give a true idea of the dynamics of what is displayed on the screen.

Fig. 4 shows the evolution of the salp population in the presence of different sizes of the checkerboard pattern of food. The advantage of this pattern is that the overall amount of food is the same whatever the pavement size if the same initial amount of food (parameter 18, Table 2) is set at each non-empty position. Food thus differs only in the size of the food patches.

With patches of size 1 cell × 1 cell (Fig. 4A), the interpatch empty areas (corresponding to periods of starvation) are small enough to not significantly deplete the individuals' reserves. In the presence of such abundant food supply, the population rapidly expands. Few individuals die during the traversal of the empty areas; when they become mature, the other individuals reproduce, and their offspring colonize new areas. The computer memory is exhausted after about 130 to 140 d because too many individuals are created.

With patches of size 2 × 2 (separated with gaps of the same size), the population increases steadily; at each generation a few more zooids colonize new patches (Fig. 4B).

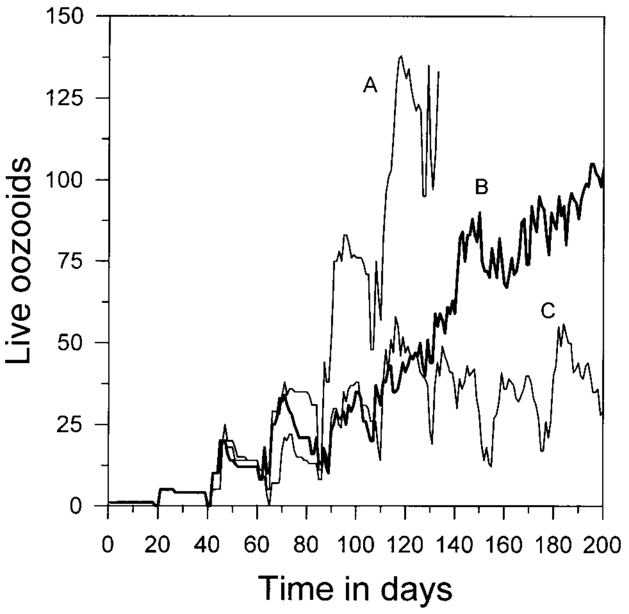

Fig. 4. Evolution of the artificial salp population (oozooids only) with different food levels. Food is initially distributed in a checkerboard pattern, with equal full cell alternating with empty cells. (A) Cells of size 1 position × 1 position, initial food value = 170 µg-at. N; memory is exhausted after 133 d. (B) Size 2 × 2, 200 µg-at. N. (C) Size 4 × 4, 200 µg-at. N. Other parameters as in Table 2

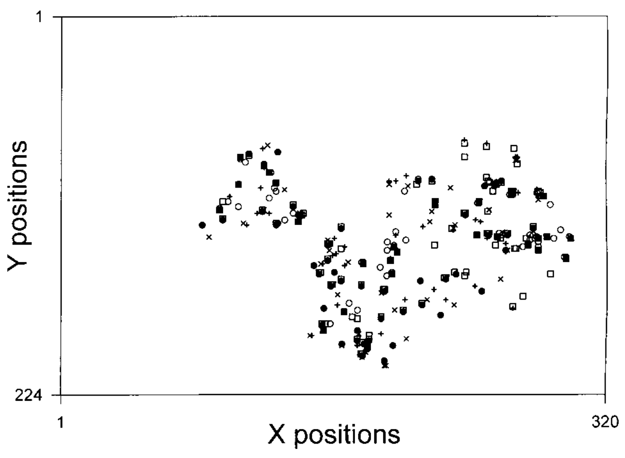

Fig. 5. Example of spatial repartition of the zooids after 200 d of simulation, corresponding to the final state of the simulation shown in Fig. 4B. On the computer screen the actual size of the zooid symbols is much smaller: each zooid is represented by a colored dot 2 × 2 pixels wide (the zooid positions cannot overlap). Surface is on top and depth toward the bottom. (o) Non-mature live oozooids; (●) mature live oozooids; (+) oozooid cadavers. (□) Non-mature live chains; (■) mature live chains; (×) chain cadavers

With patches of size 4 × 4, the situation is more critical. In some simulations the initial individuals do not encounter enough food patches, and the population becomes extinct. More often some critical mass is reached, and there are enough successful individuals to restart the development of the population (Fig. 4C).

These experiments show that, for an identical total amount of food, small food patches separated by small empty gaps are more likely to sustain salp swarms than large but distant patches. Once a salp has found a food patch and accumulated reserves by feeding on it, it has to live on these reserves until it finds another patch. With the parameter values of Table 2, the corresponding optimal inter-patch distance is on the order of 2 positions, corresponding to about 2 km (horizontally). An example of spatial distribution after 200 d is shown in Fig. 5.

DISCUSSION

An agent-based model

The CALIFE software is a kind of individual-based model (Huston et al. 1988), but one where no differential equations are computed for individuals with a given state. It belongs to a category of agent-based models where the individuals are autonomous entities concurrently executing simple behaviors. They are reactive agents (see Müller 1996), rather than intelligent agents (Steels 1995): they have no plan, no goal except the satisfaction of their immediate needs. They move randomly; when they discover food, they feed; otherwise they grow, metabolize, and reproduce. These simple reactive agents are appropriate to model invertebrates which do not cooperate.

Space

The spatial extent of the salp swarm is first related to the swimming capabilities of the individuals, the way they change their swimming direction, and also to the persistence of their reserves when they are starving. The software model may give some clues on these factors.

In CALIFE space is only represented with a 2-dimensional grid of cells. The extension to a 3-dimensional volume is not basically a difficult programming problem (just add a third coordinate, and modify accordingly the procedures related to moving). The resulting huge memory requirements could be reduced with a sparse matrix approach. A special graphic hardware would be necessary to visualize in real-time the positions of the zooids in perspective view, with hidden parts removed. In 3 dimensions, an individual would have 26 possible directions instead of 8. It is not obvious whether this 3-dimensional view would provide a much better understanding of swarm development, but with the necessary hardware this approach can be tried.

Food

Food in the ocean is distributed in patches. Salps are very effective filter-feeders (Madin & Kremer 1995). Swarm production is their response to favorable trophic conditions. Because salps are explosive reproductive machines, they are able to quickly exploit any food patch they may encounter. These encounters are probably random, but any food patch encountered by a 'lucky' individual is fully exploited by its offspring. This explains why the swarm extent may be so variable (see Fig. 3).

Because salps survive in their unpredictable environment generation after generation, it may be inferred that their biological characteristics have been tuned by evolution to cope with this changing environment. However, the swarm contains the germ of its extinction because, unless food is continuously regenerated (as in an upwelling area), the population must reach another rich patch when the current one is exhausted. To exploit an oceanic area successfully, individuals should be able to survive between patches. CALIFE does not presently model the temporal aspect of food renewal, but its layout makes it adequate to illustrate the spatial aspect.

In this version of CALIFE, I decided not to implement food renewal dynamics. Not because of technical problems (a periodic task may update the food content of each cell, according to some function of the amount of food present, the date, or the depth), but because this would quickly lead to a model of a precise ecological situation. For example, food could be unevenly distributed, matching some spatial pattern. This spatial heterogeneity would, in turn, demand a knowledge of the fluxes from and to the adjacent areas. The parameters of the food renewal function could incorporate the rate of cell divisions of phytoplankton, etc. In the presence of renewed food, the individuals would proliferate exponentially, so their metabolism needs to be very precisely adjusted. For this version, it was judged simpler to depict a homogeneous area, with a fixed initial mean amount in each elementary volume corresponding to 1 position.

Simulation software

CALIFE is not strictly speaking a simulation software in the sense of this term in engineering. In industry, simulations are replicated runs of a model where events are generated according to some adequate statistical distributions. The aim of such simulations is to obtain an estimate of the variability of the parameter values of a model under some constraints, to see the outcome of the input values. In CALIFE the emphasis is more placed on the design of the model. The runs of the software serve mainly to see if the conceptual model 'works', or whether something needs to be modified to reflect more faithfully salp swarms in the ocean. When some confidence is obtained that artificial salps are behaving like real salps, it will be possible to customize the software to answer precise ecological questions, making it a true simulation software with statistical distributions of the parameters.

CALIFE in its present state could be tailored to answer questions such as: are salps able to detect food at distance (and, if yes, at what maximum distance?); what happens if they are able to modify their swimming direction toward food; how long are they able to swim without food, and so on.

Validation of the model

A model consisting entirely of a software architecture is not amenable to the same validation techniques as models which can be validated against field data (Rykiel 1996, Coquillard & Hill 1997). What should be validated first is the conceptual model at the basis of the software design. Field data are usually not adequate. In oceanic samples, it is not possible to be sure that the same population was sampled each time; there are some missing data due to bad weather (even in such a protected sampling place as the Bay of Villefranche); this is further worsened for salps by the fact that chains are not well sampled as their length is of the order, or greater, than the plankton net opening. Moreover, chains break naturally when they age, increasing their spatial dispersion (a phenomenon not simulated in the software). They also break in the net and when formalin is added to the sample. All that can be counted in samples is the number of aggregates, not the number of whole chains (whereas the program gives the number of chains).

One of the first requirements for comparison with field data would be to find a high-frequency time series of *Salpa fusiformis* counts (oozooids and aggregates). Such small-scale data are not available in the literature, and even the unpublished series referred to in the 'Space' section above is too coarse. Moreover, the corresponding prevailing food conditions are not known.

However, as Rykiel (1996) has shown in an illuminating discussion, when field data are extremely difficult to obtain with the required precision, a computer model may still be very useful. What is important is to see if the purpose of the model is fulfilled. The purpose of the CALIFE software model is to allow a biological oceanographer to experiment with artificial salps, taking space and time into account, as well as a great deal of biological information. The visual display of the developing swarm helps the oceanographer to examine hypotheses such as what happens if there is one more generation of oozooids, if aggregate maturation is shorter, or if food is distributed in distant patches. Of course, as in any model, many variables are only sketched in or are completely missing. But if, for example, the influence of clogging of the salp mucous filter is the subject of interest, this new parameter can be added to the feeding algorithm with a little programming effort, and different levels of clogging may be simulated.

The 'lucky individual' effect (see the 'Calibration' section) is amplified by the explosive reproductive potential of salps. Swarm development is strongly contingent. It depends on an opportune coincidence between favorable local trophic conditions and a few individuals in an adequate state. The model can be very useful at determining what should never be observed, given some parameters: if such a situation occurs in the ocean, then the model is clearly invalidated, and the software must be modified.

Artificial salps are not the real creatures. They are only model organisms, containing selected features deemed pertinent in our current knowledge about the biological reality. This knowledge is put into action by running the software. Because the algorithms execute concurrently in all individuals, it is possible to observe the results of individual dynamic interactions in time and space. This is something which cannot be disclosed with mathematical equations.

CONCLUSION

As experimental devices, mesocosms have many advantages (Drake et al. 1996); care should however be exercised when extrapolating to the field (Carpenter 1996). I cannot resist quoting a concluding remark from Drake et al. (1996): 'Finally, we issue a call to ecologists to consider all avenues of inquiry, whether those avenues require a laboratory coat, a computer, or a pair of boots.'

In a virtual mesocosm there is no space limitation. This extension in space may easily induce one to think of the virtual mesocosm as a virtual ecosystem. However, this is a big conceptual leap. I prefer to regard this software as modeling a very large container, filled with artificial seawater seeded with axenic food, where a salp swarm develops. It simulates a kind of large laboratory apparatus, where time runs faster. It is convenient to tune physiological parameters, to explore different generation times hypotheses, to see the effect of starvation, or how food may be exploited. In the ocean, the dynamics of salp swarms depends on local spatial conditions (currents, hydrological boundaries). To test hypotheses for a given oceanic area, the virtual mesocosm should be turned into a scale model of this area. Precise food values may be very difficult to guess, but different scenarios may be tried.

In its present state, CALIFE is not written to solve a specific problem about tunicate ecology. Further adaptations, for example to study questions such as how a population of *Salpa fusiformis* may be regulated in the frontal zone of the Ligurian Sea, are a further endeavor, which asks for the insertion of domain features (like a 'hydrological front' object). The CALIFE basic structure is flexible enough to incorporate these specializations.

The main use of a software model like CALIFE is, in my opinion, the observation of the behavior of the conceptual model once translated to software specifications. An unexpected behavior may either indicate an inadequacy in the conceptual model, or may lead to new experiments or field observations. This is not different from what is done with mathematical models. The advantage of a software architecture is that it incorporates more biological knowledge, and it is free from the simplifications of mathematical models (limitations to a small number of parameters, sequential processes only, no complex spatial factors).

Work is under way to add an artificial crustacean to CALIFE, simulating the amphipod *Vibilia armata* which lives on tunicates as a parasitoid. Future developments include: rewriting the software in Ada 95 to take advantage of the truly object-oriented features and the improved concurrency facilities of the language; making the environment and the food dynamic to simulate what happens in a frontal zone; and adding another tunicate species *Thalia democratica* to study its competition with *Salpa fusiformis*.

Acknowledgements. I am indebted to Charles B. Miller (Oregon State University), who pointed out some needed clarifications and corrected the English version. The comments of several anonymous referees were very helpful to improve the text. This paper also benefited from numerous discussions with D. Hill (ISIMA, Aubière, France) and P. Coquillard (Laboratoire d'Ecologie, Clermont-Ferrand, France). D. Sud helped to experiment with the parameter values. This work was supported by the CNRS (URA 2077).

LITERATURE CITED

Agha G (1990) Concurrent object-oriented programming. Commun ACM 33:125–141

Alldredge AL, Madin LP (1982) Pelagic tunicates: unique herbivores in the marine plankton. BioSci 32:655–663

Andersen V (1985) Filtration and ingestion rates of *Salpa fusiformis* Cuvier (Tunicata: Thaliacea): effects of size, individual weight and algal concentration. J Exp Mar Biol Ecol 87:13–29

Andersen V (1986) Effect of temperature on the filtration rate and percentage of assimilation of *Salpa fusiformis* Cuvier (Tunicata: Thaliacea). Hydrobiologia 137:135–140

Andersen V, Nival P (1986) A model of the population dynamics of salps in coastal waters of the Ligurian Sea. J Plankton Res 8:1091–1110

Baveco JM, Smeulders AWM (1994) Objects for simulations: SMALLTALK and ecology. Simulation 62:42–57

Bertalanffy L von (1964) Basic concepts in quantitative biology of metabolism. Helgoländer Meeresunters 9:5–37

Bindingnavle U, Knox R, Kalb V (1995) An object-oriented environment for re-use of ecosystem models. In: Roberts CA, Beaumariage T, Herrings C, Wallace J (eds) Proceedings of the 1995 Western MultiConference on Object-Oriented Simulation. Society of Computer Simulation, San Diego, p 123–128

Bone Q, Pulsford AL, Amoroso EC (1985) The placenta of the salp (Tunicata: Thaliacea). Placenta 6:53–64

Bone Q, Trueman ER (1983) Jet propulsion in salps (Tunicata: Thaliacea). J Zool 201:481–506

Booch G (1991) Object-oriented design with applications. Benjamin/Cummings Publ Co, Redwood City

Braconnot JC, Choe SM, Nival P (1988) La croissance et le développement de *Salpa fusiformis* Cuvier (Tunicata, Thaliacea). Ann Inst Oceanogr 64:101–114

Bustillos-Guzmán J, Claustre H, Marty JC (1995) Specific phytoplankton signatures and their relationship to hydrographic conditions in the coastal northwestern Mediterranean Sea. Mar Ecol Prog Ser 124:247–258

Carpenter SR (1996) Microcosm experiments have limited relevance for community and ecosystem ecology. Ecology 77: 677–680

Cetta CM, Madin LP, Kremer P (1986) Respiration and excretion by oceanic salps. Mar Biol 91:529–537

Coleman D, Arnold D, Bodoff S, Dollin C, Gilchrist H, Hayes F, Jeremaes P (1994) Object-oriented development: the fusion method. Prentice Hall, Englewood Cliffs, NJ

Coquillard P, Hill DRC (1997) Modélisation et simulation d'écosystèmes. Des modèles déterministes aux simulations à événements discrets. Masson, Paris

Delatte B, Heitz M, Muller JF (1993) HOOD Reference Manual 3.1. Masson, Paris and Prentice-Hall, London

Drake JA, Huxel GR, Hewitt CL (1996) Microcosms as models for generating and testing community theory. Ecology 77:670–677

Foster IT (1995) Designing and building parallel programs. Concepts and tools for parallel software engineering. Addison-Wesley Publishing Company, Reading, MA

Hallam TG, Trawick TL, Wolff WF (1996) Modeling effects of chemicals in a population: application to a wading bird nesting colony. Ecol Model 92:55–178

Harmon MG, Baker TP (1988) An Ada implementation of Marsaglia's 'universal' random number generator. ACM Ada Lett 8:110–112

Hill D, Pastre J, Coquillard P, Gueugnot J (1994) Design of an ecosystem modelling environment: application to forest growth simulation. In: Halin J, Karplus W, Rimane R (eds) Proceedings of the Conference of International Simulation Societies. Society for Computer Simulation, Zurich, p 538–542

Huston M, DeAngelis D, Post W (1988) New computer models unify ecological theory. BioSci 38:682–691

Intermetrics (1995) Ada 95 Reference Manual, revised international standard ISO/IEC 8652: 1995 (E). Intermetrics Inc, Cambridge, MA

Laval P (1980) Hyperiid amphipods as crustacean parasitoids associated with gelatinous zooplankton. Oceanogr Mar Biol Annu Rev 18:11–56

Laval P (1995a) Hierarchical object-oriented design of a concurrent, individual-based, model of a pelagic tunicate bloom. Ecol Model 82:265–276

Laval P (1995b) Implementing self-reproducing artificial organisms with Ada. ACM Ada Lett 15:46–53

Laval P (1996) The representation of space in an object-oriented computational pelagic ecosystem. Ecol Model 88: 113–124

Laval P, Braconnot JC, Carré C, Goy J, Morand P, Mills C (1989) Small scale distribution of macroplankton in the Ligurian Sea (Mediterranean Sea) as observed from the manned submersible *Cyana*. J Plankton Res 11:665–685

Laval P, Braconnot JC, Lins da Silva N (1992) Deep planktonic filter-feeders found in the aphotic zone with the Cyana submersible in the Ligurian Sea (NW Mediterranean). Mar Ecol Prog Ser 79:235–241

Madin LP (1974) Field observations on the feeding behavior of salps (Tunicata:Thaliacea). Mar Biol 25:143–168

Madin LP, Deibel D (1997) Feeding and energetics of thaliaceans. In: Bone Q (ed) The biology of pelagic tunicates. Oxford University Press, Oxford, p 43–64

Madin LP, Kremer P (1995) Determination of the filter-feeding rates of salps (Tunicata, Thaliacea). ICES J Mar Sci 52: 583–595

Madin LP, Purcell JE (1992) Feeding, metabolism, and growth of *Cyclosalpa bakeri* in the subarctic Pacific. Limnol Oceanogr 37:1236–1251

Maxwell T, Costanza R (1994) Distributed object-oriented spatial ecosystem modeling. In: Herring C, Wallace J, Beaumariage T, Roberts C (eds) Proceedings of the Object-Oriented Simulation Conference (OOS'94), Jan. 24-26, 1994, Tempe, Arizona. Simul Ser 26:85–90

Mooij WM, Boersma M (1996) An object-oriented simulation framework for individual-based simulations (OSIRIS): *Daphnia* population dynamics as an example. Ecol Model 93:139–153

Müller F (1992) Hierarchical approaches to ecosystem theory. Ecol Model 63:215–242

Müller JP (1996) The design of intelligent agents, a layered approach. Lecture notes in artificial intelligence, 1177. Springer-Verlag, Berlin

Nishikawa J, Naganobu M, Ichii T, Ichii H, Terakazi M, Kawaguchi K (1995) Distribution of salps near the South Shetland Islands during austral summer (1990-1991), with special reference to krill distribution. Polar Biol 15:31–39

Olson RL, Sequeira RA (1995) An emergent computational approach to the study of ecosystem dynamics. Ecol Model 79:95–120

Rumbaugh J, Blaha M, Eddy F, Lorensen W (1991) Object-oriented modeling and design. Prentice Hall, Englewood Cliffs, NJ

Rykiel EJ Jr (1996) Testing ecological models: the meaning of validation. Ecol Model 90:229–224

Scheffer M, Baveco JM, De Angelis DL, Rose KA, van Nes EH (1995) Super-individuals a simple solution for modelling large populations on an individual basis. Ecol Model 80: 161–170

Steels L (1995) When are robots intelligent autonomous agents? Robotics Autonom Syst 15:3–9

Verbraeck A, de Vreede GJ (1993) Animation as a communication vehicle in simulation studies. In: Pavé A (ed) Proc 1993 European Simulation Multiconference, ESM93 École Normale supérieure de Lyon. Society for Computer Simulation, Lyon, p 670–674

Wenzel V (1994) MOSES as an object-oriented ecology-specific modelling language. In: Herring C, Wallace J, Beaumariage T, Roberts C (eds) Proceedings of the Object-Oriented Simulation Conference (OOS'94), Jan 24–26, 1994, Tempe, Arizona. Simul Ser 26:91–96

This article was submitted to the editor

Manuscript received: February 26, 1997
Revised version accepted: May 22, 1997