METHODOLOGY IN SYSTEMS MODELLING AND SIMULATION
B.P. Zeigler, M.S. Elzas, G.J. Klir, T.I. Ören (eds.)

HETERARCHICAL, SELFSTRUCTURING SIMULATION SYSTEMS: CONCEPTS AND APPLICATIONS IN BIOLOGY

P. Hogeweg and B. Hesper
Bioinformatica
University of Utrecht
de Uithof, Utrecht
Netherlands

1. INTRODUCTION

In this paper we outline a framework for modelling and simulation, which emphasises local specification of entities, dynamic generation of entities, heterarchical control and self structuring properties of simulation models. Two examples of programs using these principles are discussed in some detail. They are:
- MIRROR, a program for modelling Moving, Interacting, Reproducing and Retiring Organisms, and
- MICMAC, a program, exploring "micro/macro" relations among processes for self structuring.

In the discussion on MIRROR we emphasise the local specification and heterarchical control using DEMONs; in the discussion on MICMAC we employ self structuring properties through the use of EXPERTs.
Both programs are implemented in SIMULA/67 (Dahl, Myhrhaug and Nygaard, 1970). We found the CLASS concept of this language useful, although the strict hierarchical structure of CLASSes is less suitable for our purposes.
In the more technical part of the paper, familiarity with SIMULA/67 is assumed, but the concepts introduced should be understandable without such background knowledge.

"global state considered harmful"
C. Hewitt (1973)

2. LOCAL SPECIFICATION OF ENTITIES

2.1 Parts and wholes

Biology, like physics, is simple if and only if viewed locally. This implies that an entity is to be described in terms of information available to the entity itself, without reference to an outside observer. For example, "a free falling particle is to be observed from the inside of a free falling rocket with the windows closed" (Misner, Thorne and Wheeler, 1973).
Likewise, the behaviour of a cell is to be viewed in relation to its neighbours only, not in terms of the global position it occupies in an organism.

Even in models which represent such a "localness" explicitly there often remains some implicit global variable or control structure. This was true for Newton's description of moving bodies (absolute space and time remained global variables). A rather similar situation is seen in contemporary cellular models for biological systems. While the system is subdivided into cells, supposedly autonomous units, time remains a global variable: cell transitions are globally synchronised in cellular space systems (von Neumann, 1960; Ulam, 1962; cf Burks, 1970) and L-systems (Lindenmayer, 1968a,b; Herman and Liu, 1973; Herman and Rozenberg, 1975)(See also Herman et al, 1974; Lindenmayer and Rozenberg, 1975). This global synchronisation not only affects drastically the results in all practical applications as shown by Hogeweg (1978a,b), but also lacks any biological rationale.

The artificiality of the global synchronisation of the transformations of the cells is at once apparent if viewed from the standpoint of a cell as an autonomous unit. However, from the standpoint of an outside observer, who seeks to model the transformations of an entire system (subdividing it into (arbitrary) units) imposing a global synchronicity seems quite acceptable. Indeed all differential and difference equation models, as well as all automatic theoretic models employ such global synchronicity implicitly. Likewise most simulation strategies for such models enforce global synchronicity (on a sequential computer).
Such simulation models typically contain a global control structure (monitor) to run the program and use global data structures (arrays, etc.) to store the variables and parameters. Thus, although, for example in continuous system languages (CSMP, LEANS, etc.), the user decomposes the system into a set of standard blocks, and specifies the interconnections, the system translates this into the global transformation of a vector of system variables.

Implicitly global formulation can be very dangerous, in particular if the behaviour of the entities is not known (or not modelled) in detail. This danger lies in the fact that the entities are formulated as 'parts of a whole' instead of as autonomous units. Simulation models make it feasible to study how a set of (dynamically) interrelating entities can generate seemingly (i.e. in the eye of the observer) 'emergent' properties. This obviates metaphysical discourses on emergence, unless entities are implicitly controlled by the 'whole' whence all this metaphysics 're-emerges'.

In many interesting problems in biology, both the overall system and the subsystems are supposed to be an incomplete representation of parts of the biological system studied. If such is the case the subsystems should be regarded as autonomous units, and should be represented in the (simulation) model as such.

Thus we envisage a system in which the behaviour of a subsystem is entirely specified within the subsystem itself, using only its own local variables (no global data structures) and those of other subsystems with which it is acquainted. Thus the information and the information processing capabilities of the subsystem are stated explicitly in the model. It is important to endow the subsystem only with capabilities and concerns which are reasonable with regard to the object it is supposed to model (Note that we do not shun from using anthropomorphic terminology in describing our entities: it protects us from the far more serious fallacy of implicitly global control).

Such a simulation methodology is envisaged by us in the first place because of the usefulness in biology (our own research is mainly in modelling morphological development and modelling ecosystems). Besides it has many attractive properties from the simulation point of view: complete local specification of the modules increases the flexibility, extendability, transferability and last and least the readability of the model.

The maxim above this paper is "global state considered harmful". Indeed in our type of simulation methodology, the global state is not readily available even to the user (or output unit). In fact an output unit is, like the rest of the system, a locally defined entity, observing the system from its own local viewpoint through inspection of its acquaintances. As is the case for other units, the set of acquaintances may change dynamically also for the output unit, thus extracting the output on different parts of the system. Only as a special case can we endow the output unit with the power to observe all the local variables.

"an ant viewed as a behaving system is quite simple, the apparent complexity of its behaviour in time is largely a reflexion of the complexity of the environment in which it finds itself ....."
"a man viewed as a behaving system is quite simple, the apparent complexity of his behaviour in time is largely a reflexion of the complexity of the environment in which he finds himself ...."
H.A. Simon (1969)

2.2 Sample model: Moving, Interacting, Reproducing and Retiring Organisms (MIRROR)

Features of the foregoing simulation methodology will be elaborated further in the context of a specific example: the behaviour of individual organisms in a plane. The organisms are viewed as information processing entities whose behaviour is determined (possibly stochastically) by features of the environment they can directly observe, and by their memory structure. Because they can move about, are born or die, the set of entities with which they interact varies with time. We endow our organisms with only the information actually available to them at any time. This makes it possible to study the amount and kind of information needed to perform a certain behaviour.

Examples of simple problems we can tackle with the system are:
- Formation of spatial patterns due to interaction between organisms. For example patterns due to density dependent influence on germination, growth and death of plants, or due to changes in moving direction of animals in meeting other animals etc.
- Influence of behaviour parameters of the prey on the density of prey needed by various typed of predators to survive. For example, the influence of trooping of prey on predators hiding in a bush, or predators moving about in certain ways.

In short we want to be able to model the structuring of the environment by physical processes, plants and animals, and the influence thereof on the behavioural patterns of the organisms. Thus we hope to gain further insight into the simplicity and complexity of behavioural patterns following the paradigm of Simon as stated above this chapter.

We desire that the simulation system be structured so as to allow for easy modelling of quite simple problems (such as those mentioned above) and of much more complex models, while providing a smooth path from the former to the latter.

In designing such a system it becomes apparent that the most obvious choice of information processing primitive, i.e. individual organisms, does not suffice. Organisms should be rather modelled as dynamic colonies of information processing units. This is so because:
1. We are interested in modelling a number of partially independent processes taking place in an individual. Different information is relevant to each of these, they take place in different time scales, and they interact only once in a while. Even in the simplest cases, we might need an information processing entity which handles movement of the organism step by step, one which handles the daily cycle of waking and sleeping and one which handles the reproduction cycle. These processes are partially independent (an animal does not have to check at every time of the day whether the night is falling), but also should modify each others behaviour at certain times.
2. In order to obtain the relevant information for its behaviour, each information processing unit may moreover need to 'expand' itself to check on potential interaction partners for important events which trigger its own activity. For this purpose we implemented DEMON-like structures in our system (Charniak, 1972; see also Bobrow and Winograd (1977) on "active programming").

This twofold need for organism decomposition is again a manifestation of the maxim above: describing an organism by its global state is harmful.

Moreover such a decomposition amplifies greatly the strength of Simon's paradigm.

2.3 DEMONs

Activities of animals may have to be triggered by other events. Take as an example an ambush predator (LION) hiding in a bush, waiting for a prey to come along. It may have to wait for a long time, but when the prey arrives it should be fast catching it. Implementing this by frequent checking of the environment would be very inefficient. Instead the LION should be triggered by the arrival of the prey. However the prey is prohibited to call the LION explicitly upon entering its surrounding by our demand that the knowledge of each entity is to be confined to knowledge it may reasonably possess. Instead it should ward the LION without knowing it (compare the snapping of a branch). Such a trigger can be achieved by using DEMONs.

DEMONs provide the time driven heterarchical interconnection and activation structure of our simulation system. They may be seen as generalised and localised 'wait until'. They combine the features of process (time) oriented activation systems and (conditional) event oriented systems. They reduce to either of these for particular parameter settings, and go beyond these for other parameter settings.
In most applications DEMONs are entirely transparant except at the time of their creation.
DEMONs have four parameters: TIE, TARGET, DT and FLAG. TIE is a reference to the entity which is to be 'revived' by the DEMON (which is often but not necessarily the same as the one which created the DEMON); TARGET is a reference to an entity to be watched by the DEMON; DT the time delay after which the DEMON activates itself and FLAG is a parameter to check for obsolescence of the DEMON in relation to its TIE.

REVIVAL of an entity involves:
1. providing the entity with a reference to the cause of its revival, i.e. to TARGET.
2. providing the TIE with a MESSAGE[A].
3. revival of the DEMONs associated with the entity to be revived.
4. reactivation of the entity to be revived.

REVIVAL is only executed if the DEMON is not obsolete, otherwise the DEMON is deleted from the system.

All entities in a DEMON-driven system possess a list for DEMONs and a pointer called REVIVER to receive the cause of the revival. Entities include INTEGER,REAL, BOOLEAN,ARRAY1,ARRAY2.....ARRAYn,LISTS and DPROCESS and any user defined entity which is a subCLASS of DLINK or DPROCESS. All entities can be member of a list. Convenient side-effects of the representation of the variables in such a form include stacks and dynamic allocation as standard facilities for all variables.

DEMONs take the role of REACTIVATION clauses in SIMULA/67 (there implemented through entities calles EVENT NOTICES) if TARGET is NONE and the FLAG of an entity is increased upon DEMON creation to a level above the one given to DEMONs up to that time (a REACTIVATION clause in SIMULA supersedes all previous ones whether

---

[A] MESSAGES are an important concept in Hewitt's ACTOR systems, which are related to our approach (Hewitt, 1973). A MESSAGE mechanism is included in our system (each entity has a SCRATCHPAD for this purpose) but is so far little used.

the scheduling is for earlier or later times). If on the contrary the FLAG is increased upon REVIVAL of the entity beyond the value given to all previously generated DEMONs the scheduling for the earliest time supersedes all the others. Likewise other FLAG manipulations can provide us with the latest scheduling or the first generated scheduling. Moreover any amount of interaction among DEMONs of the same TIE or the same TARGET may be achieved.

We will use the above mentioned LION as an example of a DEMON-driven animal in our implementation of a system for modelling moving, interacting, reproducing and retiring organisms.
The space in which the organisms live is subdivided into PATCHes, discrete spatial units within which the environment is supposed to be homogeneous. Consider the space divided in a fixed set of PATCHes, of fixed size (actually we implemented an extended patch structure in which the number of PATCHes and their sizes are determined by the needs of the system itself, using a way of structuring similar to the one discussed in sections 3.2 and 3.3). PATCH possesses a number of environmental variables and a list of organisms inhabiting it (REF (DHEAD) BIOTA). The distinction between environmental variables and organisms is defined in such a way that the set of environmental variables for each PATCH is fixed during the simulation, while the set of organisms will vary.

ANIMAL is a subCLASS of ORG which possesses, besides the observation procedures which are shared by all ORGs, procedures to move about. The moving procedures enter the ANIMAL in the appropriate BIOTA list. In such a context LION is defined as a subCLASS of ANIMAL. The LION sets up DEMONs watching over the surroundings (i.e. the BIOTA lists of nearby PATCHes). Moreover the LION sets up a DEMON to warn it when it is hunting time (i.e. when it has grown so hungry that it has to go searching for its prey). An outerloop of the CLASS body of LION defines the hunting behaviour, i.e. moving about and setting up new DEMONs in its new surroundings; it increases the FLAG of the LION to render obsolete old DEMONs watching over areas now out of view. The innerloop of the CLASS body defines the behaviour while hiding in the bush: upon REVIVAL, when the REVIVER is the surrounding area, it tries to find a prey there; if it finds a prey it kills it, goes fast asleep and wakes up again to watch over the surrounding area; if no prey is found (false alarm by entry of a non-prey species into its surroundings) nothing happens. The LION goes hunting when the hunger level is high (REVIVED by the hunger-DEMON); if the hunt remains unsuccessful for too long it dies from starvation.

2.4 Some simple simulation results from this system

We will mention some results which became immediately apparent while working with the system, but which, at least to us, were not obvious beforehand.

- In case of the above mentioned LION its viability is crucially dependent on the social behaviour of its prey: if the prey animals have only a slight tendency to clump (i.e. moving preferentially in the direction in which a fellow-species is observed) a much larger prey population is needed to maintain the LION, especially if it is not to go out searching when hungry. While such an effect was expected, its size was rather surprising to us.
- If entities interact by changing direction of movement when meeting, this only leads to specific spatial distribution patterns, if an entity "can see beyond its nose". By this we mean that its sensors provide it with information about a larger area of space than it covers in one 'step', i.e. the area it covers without updating its information.
  This result seems to have profound implications for the simulation methodology which we advocate.

3. THE MICMAC STRUCTURE OF MODELS

3.1 Discrete event formalism and the selection of variables and events

In formulating a simulation model we have to select 'relevant' features of the system. What is considered to be a relevant feature of a subsystem is dependent on other subsystems, which observe some features (and react on them) and do not observe or react on other features. The user, as output unit, is just one such a subsystem, as argued above, and selection of relevance by the user is analogous to selection by other subsystems. Selection of interesting features occurs very markedly in discrete event modelling, in which the underlying conceptualisation of continuous simulation models "everything is changing all the time" is replaced by "once in a while something interesting is changing somewhere in the system". The latter reduces, in theory but never in practice, to the former if "everything is considered interesting all the time". Obviously in any implementation on a digital computer some selection of interest is necessary: in fixed time step simulations the global state if considered interesting at fixed regularly spaced points in time; in variable time step integrators the selection is on the basis of calculability of the next global state. The selection of interesting features of the system on the basis of calculability is also important in discrete event systems, but the calculations are for local states as opposed to global states, and interesting points in time will be different for different subsystems. Moreover the selection of an interesting point for a subsystem A may be due to the fact that some other subsystem B needs A's local state to compute its own next state (with respect to timing or value or both). For example A may be a 'micro' entity operating on a fast time scale and B a 'macro' entity operating on a slower time scale (or vice versa). In fact, if no part of the model is directly or indirectly interested in the state of a subsystem, there is no purpose in computing it.

It should be emphasised that the selection of interesting features of the model by other components of the model contrasts with the global control denounced above. It involves only selection, not control: the 'micro' entity does not know the 'macro' entity. All that happens is that it receives a request to provide it with its state at a certain time, or to provide it with the time in which it has a certain state. Recall the LION requesting to be warned when a prey enters its range of vision; as programmed, using DEMONs, the prey warns the LION without being aware of it itself. In the case of the LION this resulted only in a reduction of the amount of calculation done by the LION; the prey kept moving step by step, choosing its moves on the basis of local circumstances. Below we describe a system in which such requests reduce the amount of computation more drastically.

3.2 Selfstructuring in simulation models

It is well known that formulating the dynamics of a model in terms of small changes in small time steps, obviates the incorporation of indirect interactions between variables. However, the goal of compounding these direct interactions postulated for the model in a simulation run is exactly to gain insight in the indirect dependencies between variables on a longer time scale. Aids for obtaining such an insight are: simulation of the entire system and inspection of the output, analytical solutions of subsystems, simulation of subsystems in isolation, fixing of variables etc.

Zeigler (1977) showed how longer time scale information gathered for subsystems once and for all, could be repeatedly used to improve the efficiency of the simulation of the composite system. Thus it is useful to incorporate partial knowledge in the model itself. Furthermore it is useful to endow the simulator with "self awareness" so that it can itself gather such partial knowledge about itself and use it to improve the simulation. However, because such knowledge is partial, it is bound to be wrong once in a while. Thus the usage of partial knowledge should be accompanied by self criticism, and the ability to retreat to the basic formula-

tion of the dynamics supplied to it. As an example of such a simulation methodology we discuss our implementation and elaboration of the model for predator prey interactions in a patchy environment proposed by Zeigler (1977).

3.3 Example: predator/prey in a patchy environment

The model is about a field of discontinuous patches in which predator and prey species interact. The interaction between predator and prey is supposed to be governed by Lotka-Volterra dynamics with a self-limiting term for the prey. Such dynamics give rise to a stable equilibrium; if however the population numbers in this equilibrium (or anywhere along the trajectory to the equilibrium) are very small, extinction will occur in practice. Such extinctions are often observed in homogeneous experimental settings of predators and prey. The model is designed to discover whether in an environment of extinction prone patches, both predator and prey may survive. In the model proposed by Zeigler, migration between patches occurs if and only if food shortage occurs, i.e., the prey migrates if it has reached its carrying capacity in a patch devoid of predators, and the predator migrates if it has exhausted the prey in a patch. It is shown by Zeigler (1978) that this system will indeed give rise to continued existence of predators and prey for a reasonable large number of parameter settings.

Straightforward simulation of this system consists of continuous simulation of the predator/prey interactions of all the patches, checking all the time for the conditions of migration, and, if these are fulfilled, establishing the migrations as discrete events. This exhaustive approach is quite unfeasible if we use many patches. However, we are interested only in occurrence of both populations and not in exact population numbers. Moreover migrations take place only at specific stages of the within-patch system. Thus we can simplify matters considerably by selecting interesting events only. Such a simplification can be carried out if we know the following partial solutions of the model: 1) how long it takes a prey to reach carrying capacity after it has colonized a patch, and how large the population is at that time; 2) how long it takes for a predator to exhaust its food, when migrated to a patch with a certain number of prey, and how many predators there will be at that time; 3) how long a skeleton predator population survives in a patch devoid of prey. This knowledge can be obtained partly by analytical solution and partly by simulating the trajectories of the predator/prey system in isolated patches. However this knowledge is not always applicable because intermediate migrations may disturb the path which was considered to be isolated in these calculations. Zeigler implemented the model providing it with the above mentioned knowledge, approximating the effect of intermediate migrations in the predator/prey system by interpolation, and changing the scheduled events for emigration from the prey patch.

We implemented Zeigler's model within the above simulation framework, emphasising local specification of the units of the system. Moreover because we wanted the system to be applicable (or at least easily extendible) to different models of interaction of populations in patches (e.g., competition or symbiotic relations) and to different numbers of interacting populations (e.g., two predators predating on one prey species), it seemed to be begging the question to require an extensive a priori knowledge of the isolated patches. Therefore we designed the system so as to gather the necessary knowledge by itself, while it could use the basic formulation of the model when no specialised knowledge was (yet) available.

3.4 EXPERTs

The selfstructuring properties are implemented using EXPERTs. EXPERTs have expertise in carrying a process from a specified initial state to a (partially) specified next state or over a specified time stretch. In the former case an EXPERT supplies the complete specification of the next state and the time at which it occurs; in the latter case it supplies the next state. Thus an EXPERT is a triple (INISTATE, NEXTSTATE, DT). EXPERTs exist in a local state space of initial conditions. EXPERTs have to know when their expertise is applicable. For this they have

to know the similarity structure (metric) of the surrounding state space. This similarity structure may be quite different in different regions of the state space and in different directions. Thus such knowledge is only locally applicable by the EXPERT itself. The EXPERT has to build up this knowledge from experience within the framework of its data structure. EXPERTs may consult other EXPERTs and the knowledge of nearby EXPERTs may be used as first approximation to its own knowledge. EXPERTs are eager to learn and apply their knowledge. As soon as a situation arises within their capabilities, they apply their knowledge to it. As soon as a situation arises which is just outside their capabilities they generate hypotheses about it. They may also be so doubtful about their capabilities that they do not solicit for a job, but are nevertheless consulted. In both the last mentioned cases they play it safe: they test their hypotheses by direct simulation.

An EXPERT is endowed with a number of alternate hypotheses it may generate. Example hypotheses are: "my own NEXTSTATE may be used for initial conditions sufficiently like my own INISTATE" (and what is sufficiently alike, it will learn), or, "the NEXTSTATE for initial conditions intermediate between my own INISTATE and that of other EXPERT(s) can be derived by linear interpolation of our NEXTSTATEs" (weighted for different directions in state space for higher dimensional state spaces). The hypotheses are tested and given a confidence rating. This rating extends to all situations closer to the EXPERT than the ones which were successfully tested. If the confidence rating is high enough the hypothesis is applied. If an EXPERT is not consulted often enough, or its knowledge is never successfully applied, it is deleted from the system. EXPERTs react on REQUESTs. REQUESTs exist in the same state space as EXPERTs, and a REQUEST is also a triple (INISTATE, partially specified NEXTSTATE, DT). It requests the missing information not given in the triple. When a REQUEST enters the state space, the EXPERTs nearby examine it; if they can handle it, they supply it with the needed information and reactivate it at the appropriate time to report back to the process which generated the REQUEST. If the REQUEST is not handled thus, it resorts to direct simulation. Upon reaching the partially specified NEXTSTATE, the EXPERTs which generated hypotheses for the REQUEST, test these and finally the process which generated the REQUEST is reactivated. If an interrupt occurs before the condition on the NEXTSTATE is fulfilled, the REQUEST deletes the previously generated hypotheses: failure should in this case not decrease the confidence rating. The REQUEST then checks whether EXPERTs are available to handle the new situation (etc.). If upon reaching the NEXTSTATE, none of the hypotheses proves to be correct (or no hypotheses were available), the REQUEST generates a new EXPERT, supplying it with the now attained missing information. This newly generated EXPERT is initialised with a region in state space, about which to generate hypotheses, of the same shape as that of its nearest neighbouring EXPERT. Otherwise, if a correct hypotheses was generated upon arrival of the REQUEST, and is now validated, no new EXPERT is generated but the system relies in the future on this hypothesis.

3.5 MICMAC, program for simulating interacting popoulations in a patchy environment

MICMAC (micro/macro) explores the use of EXPERTs in the context of the patch structured model of interacting populations. The basic entities distinguished in MICMAC are:
CLASS PATCH(X,Y);
PROCESS CLASS MICDYN (micro dynamics);
PROCESS CLASS MACDYN(PTCH) (macro dynamics); see fig. 1.
The patches are hexagonal and arranged in a toroidal FIELD, by a procedure local to the CLASS PATCH, which provides the pointers to neighbouring PATCHes. MICDYN possesses the definition of the population interactions within a PATCH, in the procedure DYN(T,POP,DPOP) and the associated parameters. MICDYN takes care of direct simulation of the population interactions as defined in the differential equations of the procedure DYN. It uses variable time step integration using the extrapolation method of Bulirsch and Stoer (1966). The timestep is adjusted on the basis of calculability (as predicted by the extrapolation method) and on the basis of interest: a check is performed to see whether an overshoot would occur with regard

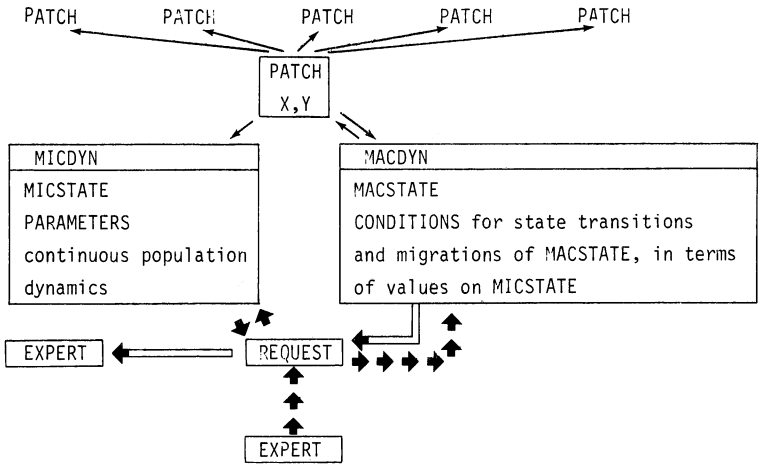

Fig. 1 Relationships among entities in MICMAC

→ pointer

⟹ generation of a new entry

➡ ➡ activation of an entity

For further explanation, see text.

to an interesting state using the predicted time step. If such an overshoot would occur, the time step is repeatedly halved (down to a certain minimum), using the intermediate calculations of the integration until the exact timing and the value of the interesting state are determined. After computing timing and next state, the process (MICDYN) is delayed for the calculated time stretch and afterwards assumes the new state. If this new state is equal to the goal state requested by MACDYN then REQUEST (and subsequently MACDYN) is reactivated and MICDYN is suspended until reactivated by a new REQUEST from MACDYN. However, MICDYN may be interrupted during its idle time (after computing timing and value of the next state) by an unexpected new REQUEST from MACDYN, which is sent because of migrations. Therefore MICDYN has to check whether it is revived on schedule (i.e, by itself) or by such an interrupt. In the latter case it has to recompute its next state after readjusting the time step according to the interrupt time.

The structure of the PROCESS CLASS MACDYN is quite different from MICDYN because it has no definition of a time dependent next state function. Instead the next state function is expressed in terms of conditions on the state of MICDYN. MACDYN possesses the procedure CONDITION which defines the conditions to be fulfilled for transition to the next state given a certain initial state. For example, if the present state is GROWTH PHASE the next state is the PREY MIGRATION PHASE and is entered upon reaching the carrying capacity (i.e., a certain number of prey). MACDYN should be warned at the time that these conditions are fulfilled. For this purpose MACDYN files a REQUEST, and then is passivated. Upon reactivation it sets its next state and files a new REQUEST, etc. Without self organising properties the REQUEST simply reactivates MICDYN and attaches itself as a DEMON to the FLAG COND, to be reactivated if this is set true, and reactivates MACDYN in its turn. The importance of the REQUEST structure lies in its use in the self organising properties of the model: EXPERTs may take over control and delay the REQUEST, so that it does not reactivate MICDYN. The EXPERT supplies in that case the REQUEST with the missing information, so that it knows when to reactivate MACDYN and what should be the next state of MICDYN and MACDYN at that time (see section 3.4).

3.6 Discussion of MICMAC

We will examine the flexibility of the above described model, and its rigidity and specificity in turn.

1. Flexibility

MICMAC easily adapts to changes in the model, which can be made by very local modifications. For example the population dynamics of the patches in isolation can be changed by modifying the procedure DYN. Likewise the dynamics of the macroprocess can be changed by local modifications: for example by modifying the conditions for migration or the migration process itself.
Moreover the model can be generalised to any number of interacting populations. In this case both the micro and the macro-dynamics are changed, but the changes are confined to model-specific changes. A very interesting generalisation of the model seems to be to change the definition of the PATCHes.
The model may be embedded in MIRROR (universe of mobile interacting ANIMALs), see section 2.2. The PATCHes are then identified with such ANIMALs and the migration neighbours are determined as the neighbours of ANIMALs of the appropriate species using the procedure which ANIMALs have for this purpose.

2. Rigidity and specificity

The system as it stands, although flexible (i.e. able to support a large class of models), is limited in several respects, which are not inherent in the framework of modelling and simulation which we have in mind.
In particular, it is simplified in the sense that there is a sharp and fixed distinction between micro and macro processes. The micro process has a time controlled next state function (here even with infinitely small time steps in the definition), while the macroprocess has an entirely condition controlled next state function and the time to reach the next state always exceeds the time step of the

microprocess. Although the terms micro and macroprocess may suggest such a relation, it is not what we have in mind: MICMAC is a relation which can exist between processes at certain times, a relation however which may change dynamically during model simulation (it may for instance be reversed). Generalisations in that direction are presently being investigated.

4. DISCUSSION AND CONCLUSION

1. "Parts and wholes"

In this paper we outlined a framework for modelling and simulation stressing as desiderata local representation and heterarchical control.
The implementation of the models here described has indeed achieved these desiderata to a considerable extent. Although the two systems were developed independently, they may each serve as a subsystem for the other.
The way in which MICMAC can be a subsystem of MIRROR has already been mentioned: the fixed patches of MICMAC can be replaced by the mobile organisms of MIRROR. As in the original MICMAC the surrounding of the unit (patch or organism) accessible for migration is given in this unit. This generalisation is useful for modelling epidemic processes in relation to interaction patterns. The other way around, MIRROR can serve as a subsystem of MICMAC. In this case the local dynamics of the interaction within a patch is modelled by a MIRROR-type system of interactions between individuals. In this case, such a MIRROR-like system operates in each patch of MICMAC. Note that the MICDYN is not necessarily a continuous formulation as seen in this example. EXPERTs may again find regularities in the so modelled dynamics to speed up the simulation of the distribution of populations over the field. Moreover, both these generalisations may be true in the same system, i.e. MICMAC serves as a subsystem of MIRROR and MIRROR serves as a subsystem of MICMAC at the same time (and these patterns may be nested). In this case we are modelling individual moving organisms which interact with each other and which are infected by several other species. The interactions between these species are again modelled in term of the behaviour of the individuals. This mutual embedding of the two systems in each other is the best proof for achieving heterarchical control.

2. Mixed mode, continuous and discrete event simulation

Mixed mode continuous and discrete event simulation systems proved to be very useful if continuous simulation is employed in those cases in which the discrete event simulation breaks down because of unexpected interrupts. Continuous simulation should then be employed until the system arrived at a recognizable state from which the larger time scale discrete event simulation can be picked up again. In the worst case this strategy results in a simulation similar to an analogous, entirely continuous, simulation (if interrupts occur very often). This manifests a use of mixed mode simulation opposite to the one usually advocated. In the latter use (e.g. Cellier, this volume) continuous simulation is employed unless it breaks down because of sudden changes in either the structure of the model or the parameter settings.

3. DEMONs, EXPERTs and self structuring

DEMONs and EXPERTs, here introduced in diverse contexts, are closely related entities. They both are autonomous entities, generated by the system, with own local knowledge and concerns, which take over control and restructure the system.
DEMONs proved to be a very helpful construct to augment the individual's information processing without violating local constraints.
DEMONs serve as links between potential interaction partners. They are used by an individual to handle 'expected' interrupts, i.e. situations which it knows may happen and which it knows to handle. In our implementation they in fact take the role of 'search images'. DEMONs may however outlive the situation which generated them.
EXPERTs likewise handle expected situations and they act as DEMONs on incoming REQUESTS. They differ from the prototype DEMON in building up their own expectations, and in containing more knowledge. Moreover, contrary to the prototype DEMON, they live in state-space rather than space-space. These differences are not pro-

found, however, and many such expansions are conceivable (and useful). Opposite to the MIRROR system, EXPERT (DEMON) control tends to be the norm in MICMAC. Therefore one is inclined to say that MICDYN, i.e. the basic continuous simulation, handles unexpected interrupts, rather than to say that EXPERTs handle the expected interrupts.

References

Bobrow, G.B. and Winograd, T. (1977), "An overview of KRL, a knowledge representation language", Cognitive Science, Vol 1, pp3-46.

Bulirsch, R. and Stoer, J. (1966), "Numerical treatment of ordinary differential equations by extrapolation methods", Numerische Mathematik 8, pp1-13.

Charniak, E. (1972), Toward a model of children's story comprehension, AI-TR-266, MIT.

Dahl, O.J., Myhrhaug, B., Nygaard, K. (1970), Simula information, Common Base Language, Norwegian Computing Center.

Heistad, E. et al (1975), NDRE SIMULA implementation user's manual, FFI-Mat Teknisk Notat S-370, Reference: Job 271/17-, Kjeller, Norway.

Herman, G.T., Arbib, M.A. and Schneider, R.E. (1974), Biologically motivated automata theory, IEEE, New York

Herman, G.T. and Rozenberg, G. (1975), Developmental systems and languages, North Holland/American Elsevier, Amsterdam.

Hewitt, C. (1973), Oral presentation at the Fourth International Joint Conference on Artificial Intelligence.

Hogeweg, P. (1977a), "Locally synchronised developmental systems, conceptual advantages of discrete event formalism", in Zeigler, B.P. (ed.), Frontiers in systems modelling, in press.

Hogeweg, P. (1978), "Simulation of the growth of cellular forms", Simulation.

Lindenmayer, A. (1968), "Mathematical models for cellular interactions in development. I: Filaments with one-sided input.
II: Simple and branching filaments with two-sided inputs".
Journal of Theoretical Biology, Vol 18, pp280-312.

Lindenmayer, A. and Rozenberg, G. (1975), Formal languages, automata and development, University of Utrecht, Netherlands.

Misner, Ch.W., Thorne, K.S. and Wheeler, J.A. (1975), Gravitation, Freeman and Co, San Francisco.

Neumann, J. von (1960), Theory of self reproducing automata, University of Illinois Press, Urbana.

Simon, H. (1969), The sciences of the artificial, MIT press.

Ulam, S.M. (1962), On some mathematical problems connected with patterns of growth of figures, reprinted in A.W. Burks (ed.), Essays on cellular automata, University of Illinois Press (1970).

Zeigler, B.P. (1977), "System theoretic description of models: a vehicle for reconciling diverse modelling concepts", in: Proceedings of the NATO conference on trends in applied general systems research, ed. G.J. Klir, Plenum Press.

Zeigler, B.P. (1978), "Multi-level multi-formalism modelling - an ecosystem example", in: Theoretical ecological systems, ed. E. Halfon, Academic Press.