Cranfield University

Uwe Neumann

An Adaptive Agent-based Multicriteria Simulation System

School of Industrial and Manufacturing Science

International Ecotechnology Research Centre

PhD Thesis

Cranfield University

School of Industrial and Manufacturing Science

International Ecotechnology Research Centre

PhD Thesis

Academic Year 1997 / 98

Uwe Neumann

An Adaptive Agent-based Multicriteria Simulation System

Supervisor: Professor Peter M. Allen

15 July 1998

This thesis is submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy (PhD)

Abstract

It is argued that traditional models of urban development are characterised by an aggregate mechanistic description of statistical units. Furthermore, important aspects of transportation are not included in these models, but urban development can be regarded as a combined process of land use change, transportation system and lifestyles. New developments in evolutionary theory provide a new paradigm for a microsimulation approach on the level of individuals, which accounts for diversity, learning and change in the population of the modelled system.

In this thesis a framework for agent-based simulations will be presented for which this new evolutionary theory provides the theoretical background. The essence of the approach builds on the mutual interdependencies between all system elements, in this case inhabitants and their environment. This principle is extended to change in the interactions of the system over time, leading to an adaptive system that mutually specifies all its elements over time.

On this framework an adaptive agent-based model for the use in urban simulations is built. The agents are equipped with a set of intrinsic needs, the satisfaction of which is expressed through a set of corresponding budgets. The budget state is fed into a Fuzzy Logic rule base for decision making. As opposed to many existing approaches to microsimulation, the agents are designed to change their behavioural rules during run time according to experience. Different adaptation strategies are tested and compete with each other.

The results of the model vindicate the conceptual framework. The essence of the underlying theory - mutual specification based on satisficing as opposed to optimisation - leads to a cognitive approach to the simulation of socio-natural systems. Microsimulation based on adaptive agents can help integrate many aspects of urban models, which are conventionally treated by separate models and can help clarify the implications of change for the inhabitants of an urban system.

Acknowledgements

Above all to my supervisor Peter Allen for his enthusiasm, guidance and support for the project - without which it would not have been possible.

To everybody at IERC for all the fruitful discussions and help I experienced during the project.

To my parents and grandmother for their generous spiritual and financial support.

To Nick for proofreading the thesis draft (at least the first half).

Contents

Figures

Equations

Tables

1 Introduction

This short introduction into the topic of this thesis - and the underlying motivations to carry out the research in the first place - will take the reader through a brief outline of shortcomings of existing work. These will be treated in more depth in Chapter 2. The next section then defines the scope of the project. Here the aims and objectives derived from the original motivation will be described in some detail. An itinerary through the rest of this thesis concludes this introduction.

1.1 The Motivation for this Project

The motivation for this research was borne out of the conviction that many of the existing models of urban land use and transportation leave out crucial areas, which can influence the outcome of a modelling exercise considerably. For a start both issues - transportation planning and urban planning - are dealt with by different disciplines both in the academic world and in practice. Transportation planning would consider the built environment and the inhabitants of a city as a given constraint and would work out solutions based on this proposition. Only recently more research into the reasons for traffic and how to influence the volume of traffic has started. On the other side urban planning is usually concerned only with the built environment and the number of inhabitants as well their needs for infrastructure. In the case of transportation however, the solution is left to transportation planning. Models, which explore possible futures built by either of the two disciplines, suffer from the same division of the subjects, and are restricted to either of the views. In reality traffic has to be regarded as a consequence of both the built environment and the distribution of land uses. In reverse, available transportation determines to a good part where new developments are built. For instance, greenbelt developments would not exist if the motorcar were not widely available, because these areas are not accessible with other means of transport. This mutual dependency reaches even further, as people's lifestyles play a great role in this development. Here the mobility, which has been made available by the motorcar, has changed people's perceptions about what commuting distances are acceptable, and where residential areas should be located. The resulting dependency on the motorcar as the only mode of transport adequate to the resulting city structure is widely regretted, but it appears to be inevitable if people's lifestyles lead to the ideal of living in residential areas in suburban locations.

These interactions have been omitted in most models, which are in use to date. Furthermore, many of the current models use approaches based on a mechanistic worldview using optimisation or equilibrium approaches. Recent developments in other areas of science have produced modelling approaches which can deal not only with quantitative, but also qualitative long term change of systems, but which have not been widely applied to the fields of urban and transportation planning.

Even empirical models in transportation planning - which at least feature accurate starting conditions - are admitted by their builders to be able to cope only with working / shopping trips (E. Kutter, personal communication). This was sufficient when the main concern of planners was about getting people to and from work. Recently the growing sector of leisure traffic has gained in importance enormously and its volume is about to surpass the one of traffic to and from work. The effects of leisure traffic on urban settlements cannot be underestimated as leisure facilities are built in locations which are not accessible by public transport, and the resulting (road) traffic has a severe impact on the quality of life in formerly quiet areas. The feedback of this development into human behaviour is considerable. On the one hand side it has become acceptable to drive to Greenfield sites, which diminishes the quality of life for people living in these areas. In return local people who like quiet residential areas will consider moving even further outside the city, because the motorcar is available to them, making trips over large distances possible.

The most important shortcoming of transportation as well as urban models is however that these models deal with spatial zones (in urban models) or traffic flows. In reality zones are populated by people and it is people who decide where to go and how to do this. People are following rules, which cannot be described with the approaches used by most models. Furthermore, people change their behaviour over time, because they adapt to changing circumstances, and people's behaviour has a huge impact on their environment. Hägerstrand (1970) was the first to mention that regional science ignored the fact that the subject is dealing with people and not with spatial aggregates, but during the past 28 years very little has happened to change this in transportation and urban modelling.

In this light the idea to integrate an urban model with a transportation model and to base this on the behaviour of an artificial population was born. The rationale is clear: Because urban development and transportation are defining each other mutually and all this depends on the behaviour of people, an integrated approach can account for these interdependencies. This integrated approach can as well reduce the need for models dealing with partial aspects of what can be considered to be one problem. A model of people's behaviour in an everyday situation can provide traffic flows as well as migration movements and changing land uses, if run over a longer periods of time. This means that such a model has to incorporate adaptation of the inhabitants of the model city, as these change their behaviours and life styles over time.

1.2 The Scope of this Project

The vision outlined above leads to a very complex model, for which the theoretical justification has to be found, and which has to be implemented and validated as well. For this purpose, existing knowledge from a multitude of scientific disciplines has to be integrated in a theoretical framework and put together in a single model. To achieve all this, e.g. to arrive at a complete urban model, much more has to be done than can be dealt with in a single project.

The scope of this project had therefore to be restricted to a contribution to an integrated model of urban systems. The area chosen was to develop and implement an adaptive agent-based model representing an artificial population in an urban model. This particular area had been chosen, because the environment (the city) has already been treated extensively by modellers, so that working models of this part are already in existence. The new part was to develop a model of human acting, which is then to be embedded into a model of the environment.

A number of benchmark objectives were derived for this above aim of the project. As the use of individual actors in an urban model is a novel approach to the problem, methodological as well as practical aspects had to be considered. This led to the following four objectives:

- To develop a methodology for modelling urban systems accounting explicitly for the interactions between inhabitants and their environment as well as the change occurring in these interactions (learning and qualitative system change).
- To test this methodology using a computer model in order to assess its validity.
- To assess in which area the use of this methodology is adequate and appropriate.
- And to gain more insights on the qualitative nature of the modelled system.

Quantitative aspects of the resulting disaggregated model as well as a real-world case study were not considered, as the focus of the project was shifted to the methodological end of the modelling exercise. In any case the quantitative validity of the model could only be tested in the case of a complete implementation, which was considered unachievable within the framework of this project.

It will be shown that most urban models do not account for individual decision making, and prefer top-down approaches based on analogues with physical phenomena, whilst other models use statistical descriptions of past behaviour, which is extrapolated into the future without allowing for change in lifestyles. A gap can be identified for there are no dynamic models of urban systems built on individual decision making in existence. However, there exist examples of simulations of artificial social systems, and the methods for transferring these to the urban domain are in existence. The concept of individual needs is introduced as a driving force for behaviour. The coexistence of several needs results in a multicriteria approach. Special consideration is given to the aspects of change in behaviour.

The computer model built displays a variety of possible modes of behaviour for individual agents, which can be interpreted as different lifestyles. These lifestyles are in existence for only limited time periods as the individuals inflict change on their shared environment, which in return requires others to adapt to this change. Although the model presented works only on a small scale, the developed methodology is considered very useful for the implementation of simulations of social systems, as it can be used to explore the implications of change on individual lifestyles.

1.3 Itinerary

In the following a brief overview of this thesis is given. In Chapter 2 a more in-depth critique of the conventional methods used in urban and transportation modelling will be presented to complement the motivation for the project given in Section 1.1 above. It will be shown how these conventional methodologies lead to the results observed. From this critique a catalogue of requirements on an integrated model of urban change and spatial interaction is derived.

The gap identified in existing work is the basis for the investigation plan proposed in Chapter 3. From here more evidence on the implementation methods for a model of individual behaviour is presented in Chapter 4, which will show that a model fulfilling the defined requirements is feasible.

This evidence leads to a conceptual framework, which is presented in Chapter 5. As this theoretical framework has to be tested in practice (practice here meaning a computer model) a computer model had to be built. The implementation of this computer model is shown in Chapter 6. Chapter 7 then summarises the results generated by the model during the experimentation period. These results and conclusions on the validity of the conceptual framework are then discussed in the last chapter. Finally, the implications of this project for possible future work are outlined.

Figure 1 shows the itinerary taken in this thesis as well as the progress of work in this project.

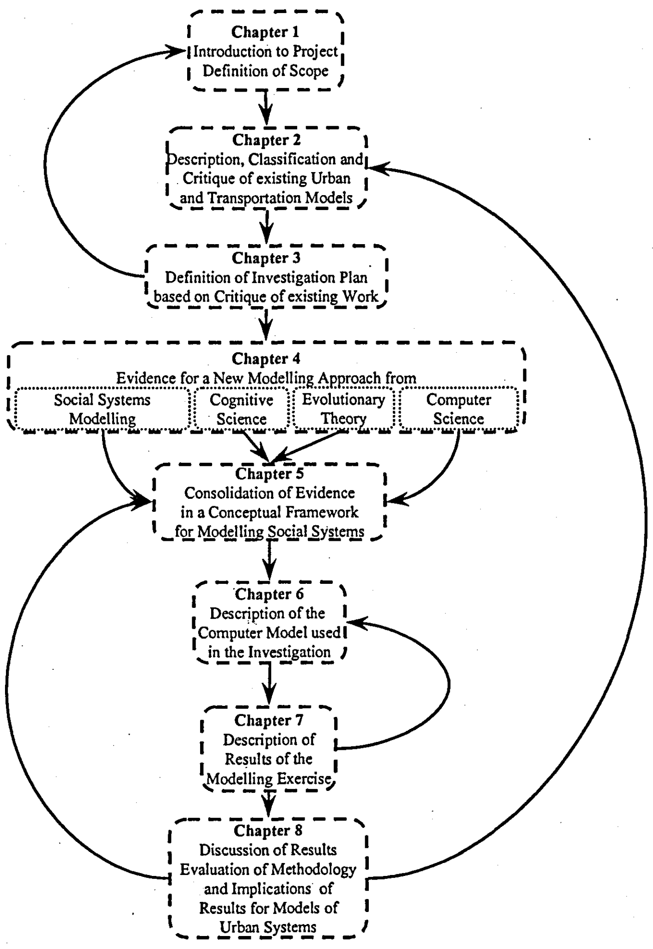
Chapter 1
Introduction to Project
Definition of Scope
Chapter 2
Description, Classification and Critique of existing Urban and Transportation Models
Chapter 3
Definition of Investigation Plan based on Critique of existing Work
Chapter 4
Evidence for a New Modelling Approach from
Social Systems Modelling
Cognitive Science
Evolutionary Theory
Computer Science
Chapter 5
Consolidation of Evidence in a Conceptual Framework for Modelling Social Systems
Chapter 6
Description of the Computer Model used in the Investigation
Chapter 7
Description of Results of the Modelling Exercise
Chapter 8
Discussion of Results Evaluation of Methodology and Implications of Results for Models of Urban Systems

Figure 1: Itinerary of this Thesis

2 A Taxonomy of Existing Work in Urban Modelling

The field of urban and transportation modelling contains a multitude of approaches to the simulation of the development of settlements and their internal structure. As these models have been developed for a number of purposes and by different disciplines, it is considered necessary to propose a classification scheme. This can clarify the main directions of these approaches. A classification makes it also easier to identify gaps in the existing work, and to formulate the novel approach taken in this project.

The classification takes place in two steps. At first, the basic classification scheme is outlined. In a second step, the taxonomy is applied to a representative sample of existing work. These approaches are further differentiated by the methods used to build the model. The methods are then explained and analysed using the example models chosen.

2.1 Definitions

In order to set up a taxonomy of urban models, the categories of classification have to be defined. The classification contains four basic categories, which will be explained in detail below. They are concerning the basic aim of the model, e.g. whether the model is attempting to explain a given phenomenon with a theory, or whether the model is built in order to describe phenomena and possibly extrapolate their tendency into the future. Secondly, the level of description determines the resolution of the results of a model. The third category is classifying how time is treated in a model. This is resulting in either a static or a dynamic model. Finally, social systems such as cities do not only contain the physical environment, such as houses, roads and other infrastructure, but they are populated by people, whose actions have a major impact on the way the system is developing.

2.1.1 Area of Interest

In the field reviewed there exist two broad areas of models. The first category is concerned with urban development, usually in the form of land uses, whilst the second area of interest rests with traffic patterns and the forecast of future traffic volumes. Urban models can be divided into four sub-categories: Economic models, models of settlement structures, models of population dynamics and land use models. Economic models have their roots in location theory, and the main objective of these models is to determine the optimal place for a company in order to minimise transport costs and to serve markets best. Although there exists a vast body of literature in this area, only some classic theories and models will be referred to in this place, as most of these models do not aim at the simulation of urban systems, but are better placed in the realms of micro-economics.

Models of settlement structures are quite similar to those of land use, the main difference between the two areas being that the former focus on the topological aspects of urban systems, whilst the latter are concerned with the future development of land uses. Finally, population dynamics models deal with the development of the population a settlement as well as the dynamics of migration processes.

The other broad area concerned with urban systems are transportation models. These can as well be divided into four main aspects. The classic area of transportation models is dealing with the projection of future demand for transportation, resulting in origin/destination volume models. The second area builds on the origin/destination models and aims at assigning the traffic volume to a transportation network. In this context, modal-split models become important, as this class of models tries to determine which mode of transport will be used to make a trip. The last aim of transportation models concerns the flow of road traffic. Traffic flow models aim at calculating the capacity and the congestion level of road networks.

A number of approaches in the past combined some of the subjects of these above models, thereby creating hybrid models. Most of these take into account only a few of the above areas, which is due to the aggregation level of these models. On the other hand, for a model of land use and population dynamics taking into account the implications of the existing transportation network it would probably make little sense to describe the exact patterns of traffic flow in the modelled city. Here the aim of the model would be to give a broad overview of possible developments without being concerned about the precise quality of the traffic flow where approximate measures of congestion would give a precise enough information on the system state.

2.1.2 Explanatory and Descriptive Models

Models can be built for two very different purposes. One class of models is aiming at explaining observed phenomena. For this purpose a theory is formulated according to which the model is built. If the model's results match reality on either a qualitative or quantitative level, the theory can be assumed to be justified. In this sense the model helps to clarify properties of the system under consideration. Furthermore, this kind of model can potentially help to explain the reasons why the system is behaving in the observed way. This class of model can be regarded as a tool to learn about the modelled system.

This objective is quite different from a descriptive model. A descriptive model is aiming at reproducing the observed behaviour a system as closely as possible. This class of model has often been used to extrapolate past behaviour into the future in order to make forecasts about system states. The methods used are in line with the different objectives of these two classes of models. While descriptive models usually use methods relying on statistics, which can give a reliable description of past behaviour, explanatory models can be purely conceptual (e.g. not using simulation methods at all) or rely on theoretical assumptions captured in differential equations, for instance.

However, a theory-based model, which according to the definition proposed here would be classified as explanatory, can in certain circumstances reduce to being simply descriptive. This is the case for theories, which are used as analogues for phenomena encountered in different domains. One example for this will be encountered in Chapter 2.3.2. The so-called gravity model is built on an analogue between the gravitational force in physics and the distribution of travel distances and trip frequencies. While the law of gravitation can explain planetary motion, for instance, and deliver a reasonable explanation for this phenomenon, the gravity model cannot deliver a reason why people choose to behave in traffic in the observed way. The gravity model can therefore only deliver a description of phenomena and not an explanation, although it is based on a theory.

2.1.3 Aggregation Level

The aggregation level chosen in a model determines how detailed the phenomena created by the model are. Low aggregation levels lead to models that can deal with individuals or social groups when dealing with a population or with individual plots in a real estate market. High aggregation levels simplify the description of the system. This can be very important if the system is big or complex, so that significant gains in the speed of a computer model can be made. The other danger with low aggregation levels lies in the reproduction of unimportant detail, which can obscure other important processes, thereby contradicting the model's aim of providing information about the system.

Highly aggregated models in land use are usually working on the level of city quarters, whereas high aggregation levels in transport models mean that the traffic volume is computed on the level of flows without referring to individual participants. Low aggregation levels mean that the model is based on socio-economic groups in the population, who are described using statistical methods. In land use models, however, low aggregation levels lead to smaller spatial units of description, without necessarily referring to the population.

2.1.4 Static and Dynamic Models

There are two basic approaches to the treatment of time in modelling, leading to very different models. The static approach applies the model's mechanisms to a start state and computes the end state on this basis without explicitly referring to the process between these points. Constraints do not change during the model's operation. It is possible to run static models for consecutive time periods, but in this case all constraints which were valid for the start state will have to be extrapolated to match the new start conditions of the model. This requires either the modeller to do this "by hand", or sub models have to be built to do this. Dynamic models on the other hand are built on descriptions or theories on the processes taking place in the modelled system. This approach can be run for infinite time periods without requiring adjustment of constraints, because these have to be treated as part of the model as they are part of the processes taking place.

In a computer model the continuous flow of time has to be divided into discrete time steps, which - if chosen small enough - approach the continuous description given by, for instance, a differential equation at the cost of increased computational effort. For most applications of dynamic models a rather coarse time step might be good enough as many processes in urban systems take place on time scales of months or years. However, it has to be determined for each application what step width is taken, as the representation of time can have grave consequences for the model's results. Too wide a step width might cause the model to skip turning points in development, while a very narrow step size increases computational effort. In general it can be said that small increments are to be preferred over large ones, but the trade-off with the required computer power has to be considered at all times.

2.1.5 Representation of Individual Decision Making

Another issue in the representation of social systems (and cities have to be regarded as such) concerns how its inhabitants are incorporated into models. Most urban models account for the behaviour of inhabitants in the form of constraints as their aggregation level is set at a high level. The precondition for an explicit account of individual decision making is however a low aggregation level. The lowest aggregation levels to be found are socio-economic groups and very rarely individuals, which are a statistical description of lifestyles based on age, status and income levels. This means that individual action is considered at a level of behavioural observation, and no reference is made to the underlying motivations for this observed behaviour.

There are some approaches to individual decision making in existence, though. Most of these try to explain behaviour in respect to economic measures or the specific utility of behavioural alternatives. Only very recently approaches to decision making based on research in cognitive science have been made. These approaches are very promising in respect to the representation of individual decision making, but lack other aspects of a holistic approach to the representation of an urban system.

As most urban models describe the development of urban systems from the top down by defining interactions between land uses and resulting migratory and traffic flows, the impact of cumulative individual behaviour and its changes cannot be accounted for. On the other hand an approach to the simulation of individual behaviour from the bottom up can account for changes on higher levels of observation, thereby reversing the traditional modelling hierarchy. Instead of assessing the implications of urban change on individual behaviour, the implications of changes in individual behaviour on the urban system are modelled. This can give further insight into the actual processes of urban change and possibly influence the way future aggregate models are built.

2.2 The Basic Classification Scheme

The above definition of classification categories forms the basis for a taxonomy of existing work in urban modelling. Figure 2 shows the entire scheme. It has to be noted that only 24 out of 128 possible classifications have been identified. In the case of urban economic models only the traditional approaches have been incorporated, and there exists further work, which is considered outside the scope of this project. Earlier approaches are almost invariantly static with the dynamic approaches appearing later on. The majority of models works on a high aggregation level as well.

Whereas there exist numerous hybrid approaches, for instance combining models of land use with population dynamics or traffic volume with land use, there are only two categories in this classification which indicate that individual decision making has been accounted for in a model. This is extraordinary, as it would appear logical at first sight to base a model of a social system on a model of its inhabitants' actions. However, it has to be taken into account that this leads to a very large model as a city can have up to several million inhabitants. This was not feasible in the past when available computer power was very much limited. In addition to this such a model would generate huge amounts of data that would be extremely difficult to analyse.

From a systemic point of view is appears desirable to integrate as many areas as possible into a model of urban development, but this will results in a very large model accounting for all possible influences. This aim will almost certainly result in a compromise on model generality, precision and realism as outlined by Levins (1966) for the field of modelling population biology. Traditional approaches to urban modelling have either focused on the precision of results, when the aim of the modelling exercise was geared towards forecasting future development, or they have tried to be as realistic as possible when describing the behavioural patterns of socio-economic groups.

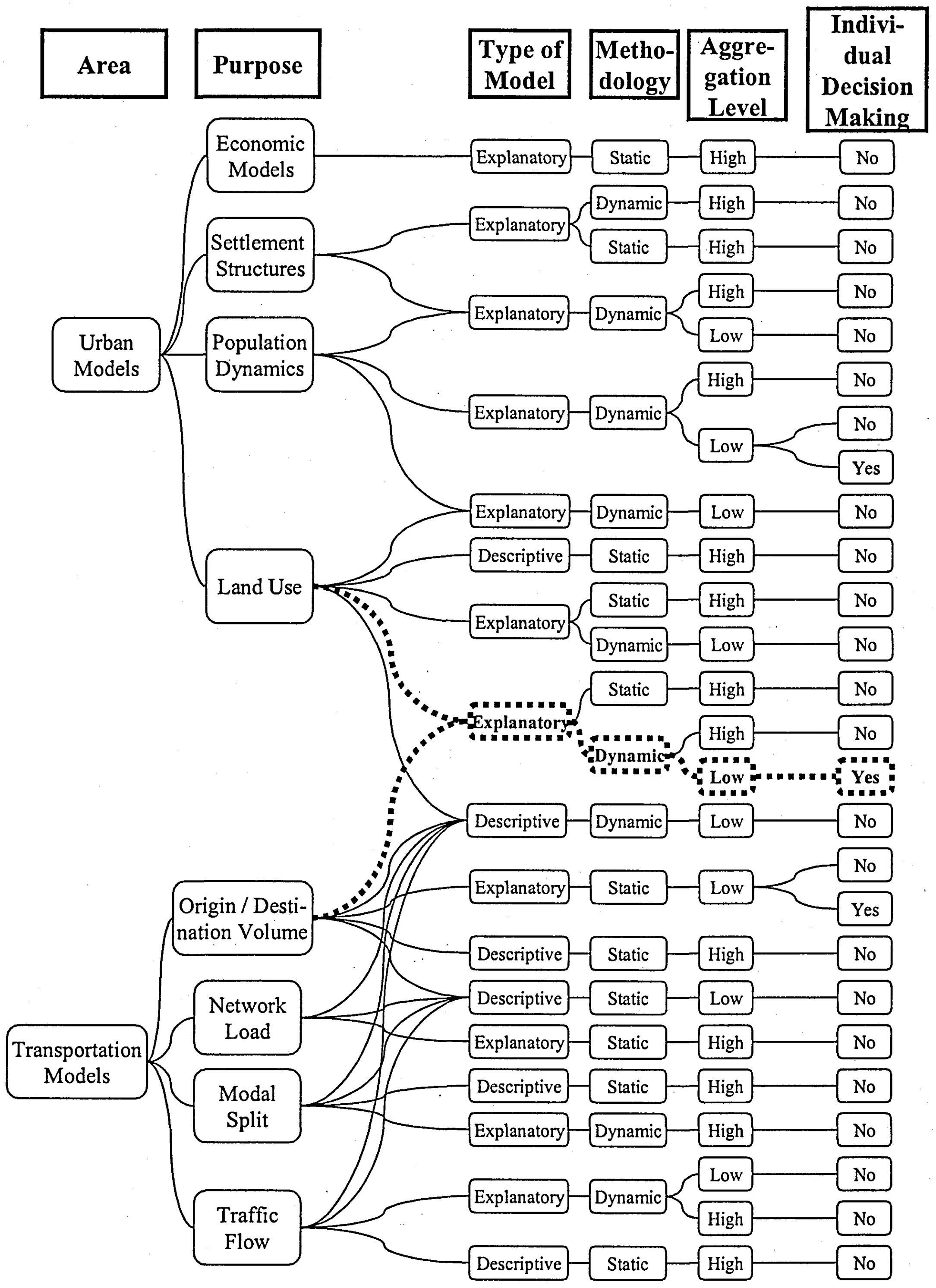

Figure 2: A Taxonomy of Urban Models

The gap identified in the field of representation of individual decision making can be filled with a model which would be geared towards generality, as the multitude of facets of individual motivations and resulting behaviour will clearly be difficult to capture with precision. Realism as well would have to be sacrificed to a good part in order to account for an explanation of the most basic trends of motivation, but this kind of model might be able to highlight some general trends in individual decision making. For a start, such a model could incorporate the development of land uses and the traffic volumes between these. The model should aim at explaining the processes taking place on a dynamic level. The place of such a model in the proposed classification scheme is indicated in bold in Figure 2.

	Urban Models											
	Economic	**Settlements**		***Settlements and* Population Dynamics**		**Population Dynamics**			**Population Dynamics *and Land* Use**		**Land Use**	
Type	Explanatory	Explanatory	Explanatory	Explanatory	Explanatory	Explanatory	Explanatory	Explanatory	Explanatory	Explanatory	Descriptive	Explanatory
Static / Dynamic	Static	Dynamic	Static	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Static	Static	Dynamic
Aggregation Level	High	High	High	High	Low	High	Low	Low	Low	High	High	High
Representation of Individual Decision Making	No	No	No	No	No	No	No	Yes	No	No	No	No
Conceptual Model	von Thunen, 1826	Parr, 1981	Christaller, 1933							Burgess, 1927; Hoyt, 1939; Harris / Ullman, 1945		
Gravitation Model											Lowry, 1964	Batty, 1976
Entropy Maximisation											Wilson, 1970	
Equilibrium Methods											Lowry, 1964	Batty, 1976
Time Use/ Time Budgets										Chapin and Logan, 1970		
System Dynamics						Forrester, 1969						
Cellular Automata		Couclelis, 1985; Batty and Xie, 1994					Portugali and Benenson, 1995					*White and* Engelen, 1993; Engelen, White and Uljee, 1995
Differential Equations						Dendrinos and Mullally, 1981; 1985						
Master Equation				Haag et al., 1992			Weidlich and Haag, 1988					
Fractals		Batty and Longley, 1986; Longley, Batty and Fothering-ham, 1992										
Self-Organisation		Allen and Sanglier, 1981							Allen, 1997b			Beaumont, Clarke and Wilson, 1981
Agent-based					Sanders et al., 1994			Portugali and Benenson, 1998				

Table 1: Classification of Urban Models (Examples)

	Hybrid Models				Transportation Models								
	Land Use and Traffic Volume			Land Use, Traffic Volume, Network Load, Modal Split and Traffic Flow	Traffic Volume			Traffic Volume, Network Load, Modal Split and Traffic Flow	Network Load	Modal Split		Traffic Flow	
Type	Explanatory	Descriptive	Explanatory	Descriptive	Descriptive	Descriptive	Descriptive	Descriptive	Explanatory	Descriptive	Explanatory	Explanatory	Explanatory
Static / Dynamic	Static	Static	Dynamic	Dynamic	Static	Static	Static	Static	Static	Static	Dynamic	Dynamic	Dynamic
Aggregation Level	High	Low	High	High	Low	Low	High	Low	High	High	Low	Low	High
Representation of Individual Decision Making	No	No	No	No	No	Yes	No	No	No	No	No	No	No
Conceptual Model			Pred, 1973			Hagerstrand, 1970							
Entropy Maximisation				IRPUD (Wegener, 1983; 1985)	Wilson, 1970								
Optimisation (Linear Programming)	TOPAZ (Brotchie 1969, Sharpe 1983)												
Equilibrium Methods	Echenique et al., 1974; Batten, 1983			IRPUD (Wegener, 1983; 1985)					Wardrop, 1952; Smith, 1979; Fisk, 1980				
Discrete Choice				IRPUD (Wegener, 1983; 1985)			Recker and McNally, 1986; Kitamura, 1984; Pas, 1988			Sinha, Khanna and Arora, 1983			
Statistical/ Probabilistic Methods					Clark, Dix and Goodwin, 1982			Kutter, 1973; 1984					
Time Use/ Time Budgets		Chapin, 1968											
Cognitive Approaches						Garling et al., 1993; 1998							
Cellular Automata												Rasmussen and Nagel, 1994	
Differential Equations			Cordey-Hayes and Varaprasad, 1982								Kahn, Deneubourg and de Palma, 1981; 1985		Lighthill and Whitham, 1955

Table 2: Classification of Hybrid and Transportation Models (Examples)

2.3 An Extension of the Classification Scheme to Incorporate Methods

The above classification of models in urban and transportation planning can only give a broad overview on the concepts used in the past. For a more thorough assessment of existing work it appears to be necessary to review the concepts and methods used in these approaches as well. For this purpose the classification scheme is reformulated in a matrix which shows the types of models in the columns and the methods used in rows (Table 1 and Table 2). Examples for applications are indicated in the matrix itself. The methods will be explained below using the existing work for clarification.

2.3.1 Conceptual Models

One method of model building is to design a conceptual model of the observed system. A conceptual model is not a method, which can be directly used in a computer model, but a way of explaining phenomena of the natural world based on a theory. Conceptual models can therefore be classified as explanatory models. The first conceptual model in the context of urban development is von Thünen's (1826) model of land use around settlements. This model was of purely economic nature. Von Thünen related the transportation costs of agricultural products and their market price. He showed that there exist concentric zones around a settlement which are predestined for growing certain crops if a maximum profit respectively minimal transportation costs are to be achieved.

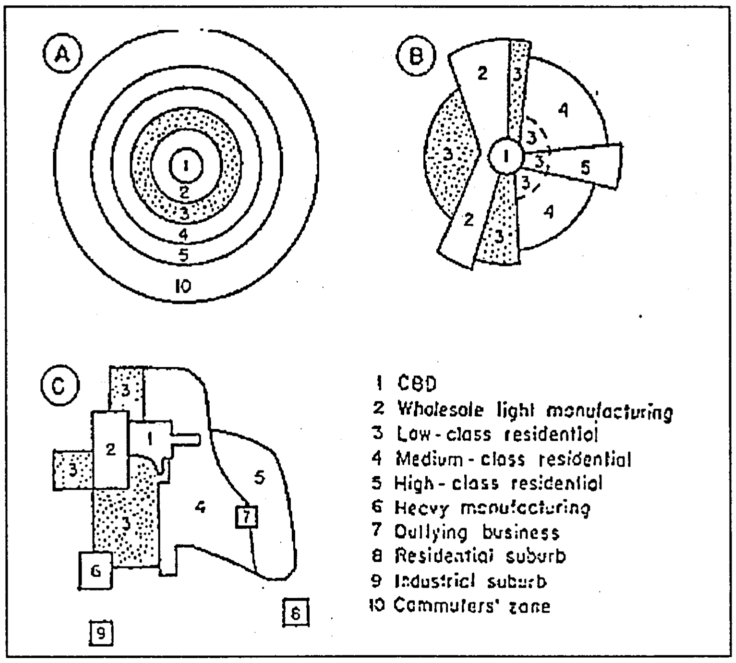

Figure 3: The Models of Burgess (A), Hoyt (B) and Harris/Ullman (C) (after Garner, 1967, p. 339)

The aim of a great number of urban models has since been to explain and possibly to forecast future land use. The best-known conceptual models in this area are Burgess' (1927) model of concentric expanding zones, which was later modified by Hoyt (1939) and Harris and Ullman (1945) (Figure 3). These models explain the growth of cities on a static basis without referring to processes of individual decision making, but refer to zones of different land uses.

The most prominent conceptual model of settlement structures is Christaller's (1933) theory of central places. This model postulated a hierarchy of settlements in a given region. On each level of centrality there exist a number of places complementing each other in respect to the level of services and goods offered in these places. Naturally, there exist only few places where rare services and goods are offered. Goods required on a daily basis like food are offered in nearly all places on the other hand. The level of services and goods available in a location determines the centrality of a place. This model is of static nature and can therefore not explain how these hierarchies come into existence. Parr (1981) provided a dynamical perspective on central place theory by introducing three patterns of temporal change into the basic theory. These are the formation of a new level, the modification in the extent of a level and the disappearance of a level of centrality. Although this theory does not describe the process of change in the central place system, which means that it is not a dynamic model in the strict sense, it can explain change in systems on a pre-change / post change basis, and it should therefore be classified as dynamic.

The last conceptual model to be mentioned in this place is Pred's (1973) city-system development based on the diffusion of information. Although not referring to the actions of individuals - information is treated as a flow between zones and not a result of communication - it can be regarded as one of the first theories, which relate the development of settlements to incomplete information of the actors in the system. Growth is seen as depending on the knowledge on innovations leading to new possibilities for economic activity. As discoveries are made in a specific place, the information about this has to get to other places before this new activity can be made available. Innovation in a given sector of the economy causes increased demand in other sectors as well, as the sectors of an economy are interdependent. This leads to multiple growth inducing feedback loops. A sub-model for transportation is developed on similar principles as well. Innovation in transport usually means a faster, more efficient service. This leads to a "space-time convergence", which simply means that more places than before are accessible. This leads to a concentration process of business in order to take advantage of economies of scale, which in return leads to more interaction between places. More interaction means that the relative advantage of the new transportation system is decreasing as congestion sets in. This creates demand for new transportation systems, as the new state has in the end brought no long-term improvement over the previous one. This model is very closely related to Time-Budget approaches, which will be discussed in Chapter 2.3.7.

2.3.2 The Gravity Model

One of the earliest computational approaches to modelling the interaction between two places is the so-called gravity model. This approach takes its name from the analogy with the physical law of gravitation. Here the urban structure is assumed to be static, and the distribution of employment, services (rarely taken into account in these early models) and residential areas and their distances is used to make forecasts on the traffic between those points. The primary concern was to give predictions of weekday traffic flows to work.

$$I_{ij} = G\frac{P_i P_j}{d_{ij}^b}$$

where I_{ij} = *the interaction between areas i and j*

P_i, P_j = *the size of areas i and j*

d_{ij} = *the distance between areas i and j*

b = *a power or exponent applied to the distance between the areas*

G = *an empirically determined constant*

Equation 1: Basic Equation of the Gravity Model (after Lee, 1973, p. 58)

The assumptions is that the traffic volume is reciprocal to the distance of two origin / destination pairs, whilst the number of employment / residential use (analogue to the mass of two bodies in physics) is determining the overall traffic volume. Alternatively, it is also possible to define "attractivities" as a measure of accessibility, economic activity or "personal" preference of areas instead of using the number of jobs or dwellings. The advantage of using attractivities over quantities is that it makes it easier to calibrate a model to survey data, which takes into account the perception of the population. This leads to an equation very similar to the physical law of gravity, hence the name of this approach.

The gravity model is static, and works on a high aggregation level without referring to the actual reasons for people to go from a place to another. Although the approach can be regarded as being explanatory on its own, gravity-type models are usually calibrated to observed levels of traffic. In this case the explanation of traffic is reduced to a description of past patterns, which might lose its validity if the activity (attractivity) levels are changed.

Another problem with the gravity model is that the physical law of gravitation is built on symmetric field forces, which is the case in physics, but extremely rare in urban systems. On this philosophical level it is questionable whether the use of the analogy can be justified. However, gravity models are widespread in urban modelling, because of their simple formulation. Example applications can be found in Lowry's urban model (see 2.3.5) or as sub-models of dynamic models, such as Batty's (1976) approach to land use modelling. As dynamic computer models rely on discrete time steps, it is possible to use static methods such as the gravity model in each time step in order to generate a pseudo-dynamic series of data points, which approaches the description by a dynamic function.

2.3.3 Entropy Maximisation

Wilson (1972) showed that a general form of the gravity model can be derived from applying Shannon and Weaver's (1949) information theory to the spatial distribution of traffic flows. This approach estimates the most probable matrix of trips between a set of given places by maximising the entropy of the distribution of trips. The model is in its original form restricted to estimating trips to work starting from a known number of jobs, number of workers living in defined zones, transportation costs and total expenditure for this segment of the transportation market. The final equations read as follows:

$$T_{ij} = A_i B_j O_i D_j \exp(-\beta c_{ij})$$

$$\textit{with}:$$

$$A_i = \frac{1}{\sum_j B_j D_j \exp(-\beta c_{ij})}$$

$$B_j = \frac{1}{\sum_i A_i O_i \exp(-\beta c_{ij})}$$

$$\textit{where}:$$

$$T_{ij} := \textit{number of individual s living in i and working in j (to be estimated)}$$

$$O_i := \textit{total number of workers living in i (given)}$$

$$D_j := \textit{total number of jobs in j (given)}$$

$$c_{ij} := \textit{cost of travelling from i to j (given)}$$

Equation 2: Equations of the Entropy Maximising Model (after Wilson, 1970, p. 5)

These equations return the most probable distribution of trips to work between all places i and j satisfying the following constraints:

$$\sum_i T_{ij} = O_i$$

$$\sum_j T_{ij} = D_j$$

$$\sum_i \sum_j T_{ij} c_{ij} = C$$

with:

C: = total expenditure on travel to work (given)

Equation 3: Constraints of the Entropy Maximising Model (after Wilson, 1970, p. 4)

While this formulation uses economic cost for the estimation of the trip matrix, it is also possible to use the trip time or a combination of factors instead, because the requirement to know the total expenditure on transport does not appear in the final equation satisfying the given constraints. The entropy model proves mathematically that the gravity approach derived from observation in fact is the most probable distribution of trips, when Shannon and Weaver's theory is applied to transportation.

The generality of this formulation has meant that this approach was featured in many urban models, such as Wegener, 1983; 1985. Other uses of the maximum entropy approach include modal-split models (Wilson 1972) and models of residential location (Wilson, Rees and Leigh, 1977). Although it gives credibility to the approach it still does not overcome the basic limitations of the gravity model, namely the static nature of the formulation and its high aggregation level. Reif (1973) already suggested either to introduce behavioural parameters into the entropy approach, or to disaggregate the model by using data from socio-economic groups as people's behaviour is only incorporated in an averaging way.

2.3.4 Optimising Models

Optimising models in urban planning have a different objective to most other approaches in the field. This class of model uses an optimisation algorithm such as Linear Programming (LP) in order to derive the most efficient configuration of a settlement as opposed to the aim of explaining or describing the development of an urban system. Only one approach will be outlined here. Sharpe (1983) proposes an optimising model based on three criteria using the TOPAZ (Technique for Optimum Placement of Activities in Zones) algorithm developed by Brotchie (1969):

1. Minimise the total combined energy use (or cost) of transport.
2. Minimise the total combined energy use of land use development.
3. Minimise the total combined cost of demolition of existing activity.

This model is designed to provide guidance for planners in respect to the development of future zoning plans for a given community. This overall objective, however, appears to be hard to reach with a global model. Sharpe points out: "The use of such an objective ignores any diversity that may occur in the trip distribution due to individuals pursuing their own separate objectives rather than complying with the objective to minimise total community cost." (Sharpe, 1983, p. 52). Energy efficiency is desirable for a community, but it cannot be determined with a macro scale model whether the required behaviour of the population can be attained in the context of current (or even future) lifestyles. Furthermore, the approach reaches a static solution for a given increase in population or economic activity. In reality, this solution would only be valid for exactly this state of which it is not known if it will be reached. Even if the projected state were attained, it would require all economic activity such as land use development to comply with an imperative planning law.

2.3.5 Equilibrium Methods

Many early urban models use the assumption that any developments in the modelled system reach equilibrium at some point in time. Equilibrium approaches are the most prominent feature of transportation network load models, and they underlie all iterative solutions. In land use modelling there are static as well as dynamic approaches which use the equilibrium assumption. Both will be treated by example.

The first - and still one of the most prominent - approach to modelling land use mathematically, e.g. using a computer model, was developed by Lowry (1964) for the Pittsburgh region. His model started the so-called "quantitative revolution", which provided researchers with the ability not only to assess urban growth qualitatively with conceptual models (2.3.1), but also to estimate the quantitative impact of change.

Lowry's model starts with the current number of employment in the in the so-called "base sector" and its spatial distribution. The base sector comprises the industries producing for the local area plus those exporting services and goods from the area. Then the existing employment is used to calculate the land needed for residential use. In the next step, the model calculates the number of resulting employment in the tertiary (service) sector, which is needed to supply the population employed in the base sector. This employment is then spatially allocated, thereby taking up land. In an iteration loop, the now increased population is again reallocated to the residential zones, feeding back into an increased demand for services and goods.

In Lowry's approach a gravity model (see 2.3.2) is determining the exact localities for residential and commercial / industrial use based on the accessibility (either in terms of geographical or time distance) of each place. Accessibility is in return a predetermined constraint of the transportation network, which is as well assumed as static over time. Mathematically, the model is iterating towards a static solution from a system of linear equations. Because there is only one input parameter (the employment in the base sector), it cannot produce any dynamics over time and the solution will always be at equilibrium of the equation system.

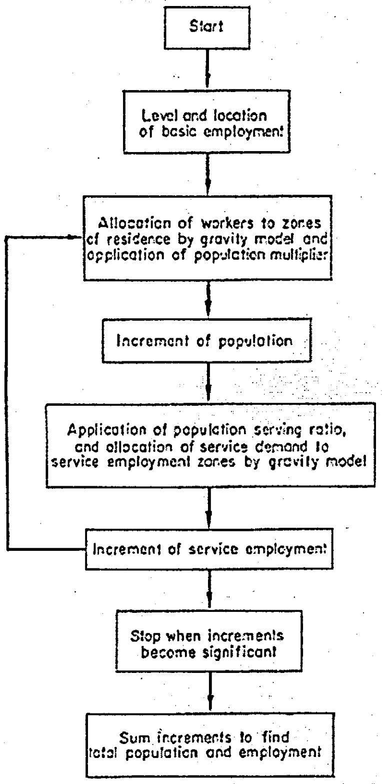
Start
Level and location of basic employment
Allocation of workers to zones of residence by gravity model and application of population multiplier
Increment of population
Application of population serving ratio, and allocation of service demand to service employment zones by gravity model
Increment of service employment
Stop when increments become significant
Sum increments to find total population and employment

Figure 4: Flow-chart of the Lowry-model (after Lee, 1973, p. 98)

This model is working on the aggregate level of zones, and its set-up can only reflect the average preferences of the population in their allocation to residential areas. The use of a gravity model means as well that the choice parameter for the allocation of population or land uses is economic (transport cost) or based on (time) distance to areas where employment can be found.

A dynamic formulation of an aggregate land use model incorporating also aspects of transportation based on equilibrium can be found in Batty (1976, p. 313 ff.). In this model the algorithm used in the Lowry model is extended to allow for relocation of residents and businesses in each time period. Supply and demand for residential land use is separated and the residential allocation module then assigns the demand to the supplied floorspace. However, the total floorspace available in the model is treated as an exogenous variable, so that this crucial parameter has to be fed from outside the actual model, which makes the dynamic approach weaker than it could be. In addition to this time lags are introduced for changes to have effects on the attractivities of certain locations. The model incorporates as well the resulting traffic between zones. For this the entropy approach is chosen which calculates a trip distribution matrix. This distribution is then assigned to the transportation network on the basis of the shortest routes between zones.

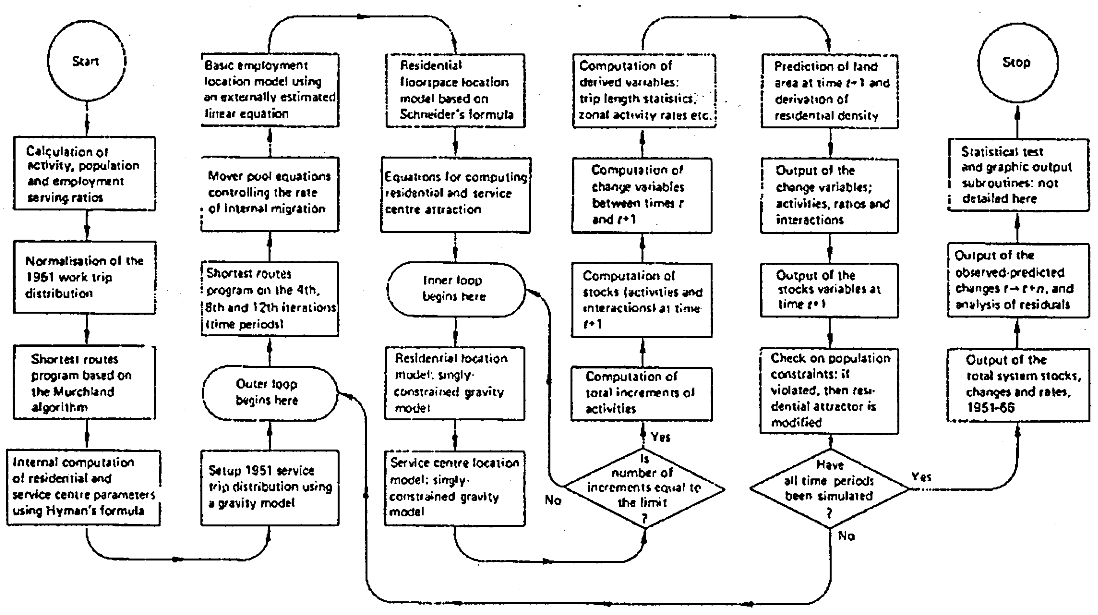
Start
Calculation of activity, population and employment serving ratios
Normalisation of the 1961 work trip distribution
Shortest routes program based on the Murchland algorithm
Internal computation of residential and service centre parameters using Hyman's formula
Basic employment location model using an externally estimated linear equation
Mover pool equations controlling the rate of internal migration
Shortest routes program on the 4th, 8th and 12th iterations (time periods)
Outer loop begins here
Setup 1951 service trip distribution using a gravity model
Residential floorspace location model based on Schneider's formula
Equations for computing residential and service centre attraction
Inner loop begins here
Residential location model; singly-constrained gravity model
Service centre location model; singly-constrained gravity model
Computation of derived variables: trip length statistics; zonal activity rates etc.
Computation of change variables between times t and t+1
Computation of stocks (activities and interactions) at time t+1
Computation of total increments of activities
Yes
Is number of increments equal to the limit ?
No
Prediction of land area at time t+1 and derivation of residential density
Output of the change variables; activities, ratios and interactions
Output of the stocks variables at time t+1
Check on population constraints: if violated, then residential attractor is modified
Have all time periods been simulated ?
Yes
No
Stop
Statistical test and graphic output subroutines: not detailed here
Output of the observed-predicted changes t→t+n, and analysis of residuals
Output of the total system stocks, changes and rates, 1951-66

Figure 5: Flowchart of Batty's (1976, p. 311) Dynamic Urban Model

The algorithm is designed to converge towards an equilibrium solution over time. This conforms to economic theory as it is assumed that supply and demand in a market reach an equilibrium over time. The application of the equilibrium approach for dynamic urban models is known as "Harris' principle" (Batty, 1976, p. 297), who argues that "… for well constructed urban models a set of equilibrium solutions will be available for most inputs of policies and environmental conditions." (Harris, 1970 after Batty, 1976, p. 297) "This principle suggests that although disequilibrium may be the usual condition of a dynamic model, such a model should always be tending to equilibrium and, in the absence of further stimuli, should reach this state." (Batty, 1976, p. 297)

Although this approach is technically dynamic, the assumption that all urban systems converge to equilibrium conditions biases the dynamic behaviour of the model towards an equilibrium state, so that is cannot be tested with the model whether this is really the case. Later approaches built on complex systems theory (see 2.3.15) explicitly renounce this view and use differential equations for the description, which can, but not necessarily will, lead to equilibrium states.

Equilibrium methods have been used in network load models as well. Wardrop (1952) developed the first algorithm. Fisk (1979) describes this method as follows:

> *(The user equilibrium principle) … "governs path choice behaviour, assigning each tripmaker to the least cost path between his origin and destination in such a way that his path cost cannot be reduced by switching to another path. For uncongested networks this problem reduces to an all-or-nothing minimum path assignment; for the congested case, link travel times are functions of the assigned link flows and the equilibrium state is determined as the solution of the following system of nonlinear equations:*

$$\sum_a \delta_{ak,i} S_a(f_a) = u_i \quad \text{if } h_{k,i} > 0 \quad \text{all } i \in I_N;\, k = 1,\ldots,K_i$$

$$\sum_a \delta_{ak,i} S_a(f_a) \geq u_i \quad \text{if } h_{k,i} = 0 \quad \text{all } i \in I_N;\, k = 1,\ldots,K_i$$

$$\sum_k h_{k,i} = g_i \qquad \text{all } i \in I_N$$

$$h_{k,i} \geq 0 \qquad \text{all } i \in I_N;\, k = 1,\ldots,K_i$$

$$f_a = \sum_{i,k} \delta_{ak,i} h_{k,i} \qquad \text{all } a \in A$$

where

A : *is the set of directed network links*

u_i : *is the* $o-d$ *travel cost between* $o-d$ *pair i*

$h_{k,i}$: *is the number of vehicles on path k between* $o-d$ *pair i*

g_i : *is the input flow between* $o-d$ *pair i*

$S_a(f_a)$: *is the travel cost on a link a assumed to be a continuous increasing function of the link flow* f_a

N : *is the number of* $o-d$ *pairs*

K_i : *is the number of paths between* $o-d$ *pair i*

$$\delta_{ak,i} = \begin{bmatrix} 1 & \text{if a link lies on path } k \text{ between } o-d \text{ pair } i \\ 0 & \text{otherwise} \end{bmatrix}$$

I_N : *is the set of integers from* 1 *to N.*

Equation 4: Network Equilibrium Condition Equations (after Fisk, 1979, p. 305)

These equations are known as the network equilibrium conditions." (Fisk, 1979, p. 305)

This approach is, although in many variations and improvements (see for instance Smith, 1979; Fisk, 1980), still the most dominant network assignment model in transportation planning. The basic assumption of equilibrium means that all drivers / participants know exactly how long every path between two points A and B will take, so that everybody will choose the shortest path in terms of time or cost given the total network load, which will result in overall equilibrium. In reality in a congested network all links would be utilised up to capacity (or above that, leading to congestion), but it is questionable whether all traffic participants would choose their user optimal path, as they might not be aware of better alternatives to the route used. Even if better routes are made known to drivers, it is not sure that these routes will be adopted. Especially in the case of faster routes requiring longer detours compared to the shortest link, it has been observed that these have been rejected by drivers despite their advantage over shorter routes. In aggregation, the result might come near the solution iterated by the equilibrium approach, but this does not reflect the cognitive processes of the traffic participants.

2.3.6 Discrete Choice Theory and Utility Theory

Discrete choice theory and utility theory have been closely intertwined in their use. This technique is used to determine aggregate distributions of choice between given alternatives. The theory is built on the concept of utility. Utility denotes the usefulness of an action in terms of a possible gain from choosing this action or alternative. As it is not straightforward what people actually like about doing things, there are several ways of formulating utilities. These can be derived from the actual process of doing something (like enjoying to go out with friends) or from regarding activities as useful for their outcome (like going to work in order to earn money, which has some use for buying things) (Gärling, Axhausen and Brydsten, 1996).

In the simplest version all individuals are assumed to try to obtain the maximum utility from the activities chosen. This choice process is therefore optimising and obeys the principle of rational choice. It requires in this case that all individuals know about all aspects of all available alternatives, and that they have the same preferences, so that a general utility function can be formulated. In order to take into account irrational behaviour and incomplete knowledge as it is very frequently observed, the utility functions are transformed into probabilistic formulations, which allow for certain observed distributions of choice. The utility function can in this case be defined using regression or factor analysis or other statistical tools applied to empirical data.

The best-known probabilistic formulation of a discrete choice model is the so-called multinomial logit model. Here the ratio of utilities between alternatives determines the probability of choice. It requires, however, that the systematic utilities are linear in their parameters.

$$p_n(i) = \frac{\exp(bx_{in})}{\sum_{j \in C_n} \exp(bx_{jn})}$$

$$\textit{with}$$

$$b : \textit{a vector of unknown coefficients}$$

$$x_{in}; x_{jn} : \textit{vectors of known, independent varables describing alternatives i and j}$$

Equation 5: Multinomial Logit Model (after Lerman, 1983, p. 201)

Utility theory has a very central place in classic models of individual behaviour. In an urban context, these models are found in the area of activity scheduling models (see for instance Hirsh, Prashkea and Ben-Akiva, 1986; Kitamura, 1984; 1988; Mahmassani, 1988; Pas, 1988; Recker and McNally, 1986a; b) and in mode choice models (for instance Sinha, Khanna and Arora, 1983). Utility-based discrete choice models can easily be calibrated to describe observed behaviour, but this kind of model cannot explain why the alternatives are chosen in the way described by the model, as it is not clear whether the modelled people have the same definition of utility as the modeller. Formulating a utility function means as well that it contains the average utilities of the entire population. Another danger of calibration lies in matching the model for exactly one case, the sample data. In that case the model cannot be used for any other application, because there exist spatial differences, like the layout of a different city, where there exist different lifestyles, as well as in time, because the preferences of the population might change. All this makes the formulation of an elaborate probabilistic model for just one case questionable. The problem of overcalibration is matched with the opposite effect of undercalibration, which means that if the model is calibrated to too big a sample it might just not say anything at all, because the regional variations are wiped out by averaging. The average behaviour of a large sample leads to low precision in the model's results and makes its use questionable again. Furthermore, it also questionable whether the formulation of a single, usually additive, utility function can actually account for the richness of motivations leading to the observed behaviour of people.

2.3.7 Time Budgets and other Methods based on Time Use

Time Budgets and their use by people have been mentioned in the literature as early as 1939 (Sorokin and Berger, 1939, after Kutter, 1973.). The central ideas of Time Geography have been developed from this concept to the point that time is a major constraining factor in human behaviour. Studies by Chapin (1968a, b) linked the use of time by people to the satisfaction of personal needs (for the concept of human needs see Chapter 4.3). Time was then linked to the use of space in an urban context (Chapin and Logan, 1970). The conclusions of this work are quite far-reaching. In the context of transportation, the relationship between available time and available modes of transport determines the maximum spatial range (and thereby the possible activities) of an individual.

In the course of other studies on time use (Szalai, 1971) it has become clear that time use is focused around primary (mandatory) activities such as work. Secondary activities (such as socialising with other people) on the other hand are dependant on available transport which enables people to reach more places where these non-mandatory activities are offered. Of course, the availability of transport is very much connected with the standard of living and the traditional roles of family members. Obviously, individuals who have access to efficient transport do not spend more time on mandatory activities. People rather tend to either use their extended range for the given set of activities by, say, working in a more distant place, or use the time now available to carry out more non-mandatory (leisure) activities. The idea to *save* time through providing better transport, which traditionally underlied transportation planning, has been put in question by these studies on time use.

Some figures from the National Travel Survey (1975/76 and 1985/86) illustrate this point. The National Travel Survey for 1985/86 (p. 20) states: "Speeds for short car journeys changed little since 1972/73. For long journeys, there were substantial increases in speed." On the other hand, the distance travelled on average as well as the number of journeys increased considerably over the last 30 years (see Table 3). The higher speed obtained during longer journeys gives those a relative advantage over shorter ones. This has led to a situation where people travel more often and further than before. In addition to this the average time required for each journey has increased as well, so that people now spend more time in traffic than ever. These facts have to be put into a broader perspective, because people do not have more time available per day. Here changes in society like shorter weekly working hours, higher disposable income and increased car ownership have enabled people to be more mobile, and the faster transportation system has even reinforced this tendency, instead of the intended effect of saving time in traffic!

Year	Journeys per person per week	Average length of journey (miles)	Average speed (mph)	Median speed (mph)	Average duration of journey (min)
1965	11.2	6.3	n / a	11.4	n / a
1975/76	12.4	6.9	13.5	n / a	21
1985/86	13.2	7.5	16	12.2	25

Table 3: Key Figures of Mobility (after National Travel Surveys 1975/76 and 1985/86)

Hägerstrand (1970) put the results of time-geographic research into a systematic form. He created a "socio-economic web model". In this model (see Figure 6) the paths of individuals are mapped in time-space. As it is necessary to communicate with other people, the time-space paths have to be bundled forming a "tube", whereby the distance between the bundled paths denominates the range of speech and vision as means of communication. This can be extended to include other means of communications, such as the telephone etc. It is very clear to see that fast transport increases the range of people enormously, and that the availability of such transport is socio-economically determined.

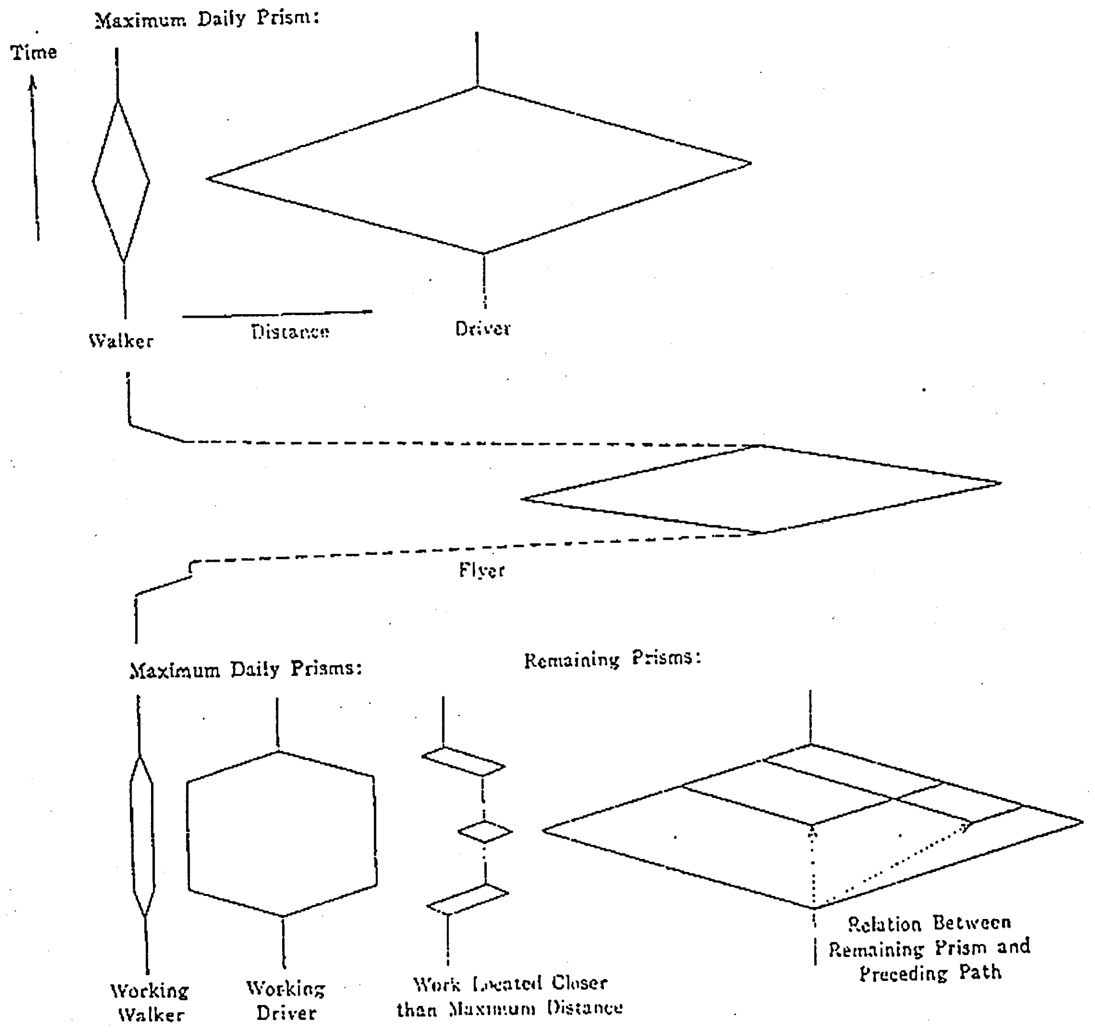

Figure 6: Hägerstrand's Concept of "Daily Prisms" (after Hägerstrand, 1970, p. 13)

The maximum range of individuals is given by the size of the "daily prism" which is created when a person uses all the time available just to travel at maximum speed starting from home and returning at half-time. In practice, the maximum prism is rarely used. In this case it is still possible to determine what the individuals can do during the remainder of the day. The "coupling constraints" given by the location of homes and businesses in time-space restrict what activities can be carried out and where, depending on what means of transport are available to the individual.

The conclusion of these studies should be clear: People's behaviour is much more governed by available time than anything else. In this sense, the 24 hours of the day provide the ultimate constraint for human activity. Therefore a model of human activity patterns should be based on the limit which is imposed by the fact that there are only 24 hours per day. This leads to another implication: People are always doing *something* at all times, and it has to be recognised that sleeping (or recreation) is an activity in its own right as are working or socialising out of home. The problem how people choose their activity patterns is dealt with in the next section.

2.3.8 Behavioural Approaches

The classic methods of transport models, such as the gravity model, do not account for individual behaviour, as traffic is regarded as a flow the size of which depends on the distance between two points and the specific attractivity of these. Behavioural approaches to transport modelling are built to incorporate a more diverse view of a population's behaviour into a model's database. More or less all behavioural models use disaggregated statistical data from surveys, from which determinants of behaviour are extracted. Kitamura (1988) lists the following areas of interest in behavioural modelling:

"• Activity participation and scheduling in time and space.
- Spacio-temporal, interpersonal, and other constraints.
  - Interaction in travel decisions.
  - Trip chaining behaviour.
- Multi-day travel behaviour.
- Interaction between individuals.
- Household structure and roles.
- Adaptation, other dynamic aspects.
- Policy applications.
- Activity models." (Kitamura, 1988, p. 12)

The main theoretical credo of behavioural research has been that a household's lifecycle determines its member's activity profiles (Clarke, Dix and Goodwin, 1982). The household is regarded as a metaphor for constraining factors on individual behaviour. On the other hand the lifecycle stage of a household induces specific needs for activities (Jones et al., 1983). In a dynamic perspective, change in behaviour is mainly attributed to the age cohort effect as members of a household progress from one lifecycle stage to another. It is proposed to conduct longitudinal studies for data acquisition on the dynamics of change in travel behaviour. In a later paper Goodwin, Kitamura and Meurs (1990) criticise the traditional four-stage approach to traffic forecasting (trip generation, trip distribution, mode choice and network assignment) as inadequate as it is lacking important dynamic aspects, especially for its lack of incorporation of change in land use patterns. This is regarded as a crucial element in the generation of transportation demand. For realistic simulations the incorporation of individual response rates to change is regarded as a method to improve the dynamic behaviour of such models. The dynamics themselves, however, are still regarded to tend to equilibrium.

These models are offering the possibility to take into account a set of different behaviours within a population, but it is still the observed, aggregate behaviour of the past (even if it is based on a variety of different behaviours with some dynamics), which is governing these models. The main achievement, however, of this kind of model is the incorporation of a relationship between the socio-economic status of people and their spatial behaviour (for instance in Kutter, 1973; 1984). The combination of social characteristics with the constraints developed by Time Geography (see 2.3.7) determines the probabilistic activity profile of a socio-economic group over the average weekday.

Another innovation is the incorporation of the so-called "journey-concept" into the model. The journey-concept's main innovation was to take into account that trips have usually more than one purpose, a trip to work can be used as well as an opportunity to go shopping on the way back. As opposed to earlier models, which regarded trips as only serving one single purpose, it was now possible to incorporate multiple activities into a single trip. This is for obvious reasons leading to a more accurate projection of the demand for transportation.

In the context of the proposed classification scheme for urban models, all these approaches have still to be classed as static, even if there have been references to dynamic methods. They are descriptive, too, as they rely almost exclusively on statistical methods, which can only capture the processes at the time of data acquisition. The aggregation level is low, but the use of behavioural groups does not allow for the representation of individual decision making.

2.3.9 Cognitive Approaches

The process of activity scheduling is of particular interest for a model of urban development based on individual acting. Activity scheduling models were first introduced to model the chaining of trips motivated by everyday activities. Because the timeframe for activities is given as the day, decisions on what activities to carry out have to be made. Whilst earlier approaches almost inevitably used utility theory (see Chapter 2.3.6), a more cognitive model is presented by Gärling et al. (1993, 1998).

For the first time the satisficing principle introduced by Simon (1981, see 4.1) is explicitly used in an activity scheduling model. This means that all decisions about what activity to take up are subject to available knowledge about opportunities. This knowledge is stored in a cognitive map. If opportunities are encountered on a regular basis they can be stored in the cognitive map, whilst rarely used opportunities are more and more difficult to retrieve over time.

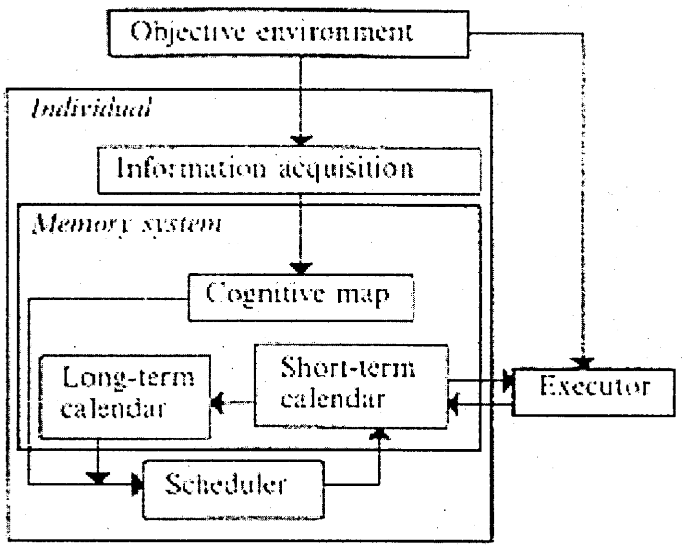

Figure 7: Flow-chart of Gärling et al.'s (1998, p. 667) Activity Scheduling Model

There are two different schedules in the model, a long-term calendar and a short-term calendar hold information about what activities have to or can be carried out at which times of the day. The information available from the cognitive map and the long-term calendar are determining the contents of the short-term calendar, which covers one day. Routine (frequently repeated) actions from the short-term calendar on the other hand will influence the contents of the long-term calendar.

As this model can be regarded as a sub-model of a larger scale transportation model, which are usually not dynamic models, but only project a forecast to one specific point in time, a fixed rule set is used in the scheduler arranging the daily timetable. This is sufficient, provided that the rule set is accurate enough to describe the processes of scheduling. On the other hand, there is only one rule set defined in the presented model of one individual. If this were applied to more than one household, the effects of diversity would not be accounted for. Consistent with the limited dynamic scope of this model is that it relies on an objective outside world, which cannot be influenced by the actions of the individual. These fixed boundary conditions do not allow for individual perception, as the impact of experience will be the same for all individuals.

Experience is one central point of this model, however it leads to a learning process which can be regarded more as an optimising rather than an exploratory process, because knowledge can only be acquired according to the fixed rules which govern the scheduling of activities in the first place. The model can account for the formation of routines, which are one important part of daily life leading to a more or less fixed timetable for standard processes without excluding deliberate action.

2.3.10 System Dynamics

The best-known system dynamics model for urban system is probably the one by Forrester (1969). The advantage of system dynamics over the iterating static model is that a dynamic solution is formulated, e.g. the model calculates the entire system state for a series of discrete time steps. The use of extensive feedback loops resulted in what should be considered to be one of the first complex systems models. However, all events were calculated on the basis of average behaviour for a population of average, uniform type.

Forrester's model has been criticised by a number of authors. Cordey-Hayes (1974) summarises his critique as follows:

> "*1. Forrester's results do not follow from counter-intuitive systems behaviour, but follow directly from the particular structure of his model, and from his evaluative measures. In essence, the model is considered to be a novel way of expressing his subjective views.*
>
> *2. Minor modifications to the migration equations give rise to markedly different values for his evaluative measures, and at the present there is scarcely sufficient understanding of urban processes to specify these equations adequately.*
>
> *3. Despite these major criticisms, the methodology is considered a potentially useful approach to simulation modelling, particularly as a contextual framework for more specific subsystem studies.*" (Cordey-Hayes, 1974, p. 174/5)

In fact the model is extensively using exponential growth functions, which is an assumption by Forrester. In this non-linear framework minor modifications to the functions used can result in completely different outcomes. In this sense, Forrester's approach is a tool, which can be used to test assumptions and theories on the nature of systems, but it will not deliver a forecast of what will happen in the future. On the other hand the assumed mechanisms can deliver results similar to the observed reality, so that a "wrong" model of the system might deliver "correct" results. This case makes it very difficult to validate a model, which does not use a descriptive approach, but tests the assumptions of the modeller against reality.

2.3.11 Cellular Automata

Cellular automata are dividing a spatial area into discrete units, the so-called cells. For each cell there are transition rules, which change the state of the cell into another state if other, neighbouring, cells around it have taken a given configuration of states at each time step. The state of a spatial cell in an urban model would be its land use. The number of cell states taken into account for a transition of state is limited as the number of neighbouring cells increases by the square of the distance to the cell in question. It is therefore a model limited to local interaction. A current model using cellular automata is the one by White and Engelen (1993; Engelen, White and Uljee, 1995). In this model the actual dynamics of cellular automata for local interaction are combined with a Lowry-type model (see 2.3.5) for longer-range interaction, because of the above limitations of cellular automata.

It has been pointed by Couclelis (1985) that this type of model has strong resemblance in its results to complex systems models built on differential equations. However, there is some criticism on how far the methodology is suited to represent urban systems. Firstly, cellular automata are discrete time models. This results in perception and action at the same instant in time as defined by the transition rule for each cell. If the cell is meant to represent human action in at a point in physical space (for instance a developer deciding whether to build additional housing), this does clearly not conform to reality and will distort the time frame of the model. The second point of critique is related to this and concerns the issue in how far a cell can represent spatial units, because the interactions between cells are arbitrary and can be between any cells.

Batty and Xie (1994) propose a probabilistic framework for the simulation of urban structures based on cellular automata. The deterministic transition functions for the cells are reformulated in a probabilistic way, so that development in a cell's neighbourhood results in a probability for the cell to change state. In Batty and Xie's example this state indicates whether the cell is built up area or not. With this methodology it is possible to generate structures analogue to a growing city, which is discriminating only between built-up area and non-built-up area. A similar objective can be found in work using fractal geometry, see 2.3.14.

An extension to the classic cellular automata methodology called "cell space" has been used by Portugali and Benenson (1995) to simulate the effects of migration into an existing city. Like cellular automata, the cell space consists of a number of cells, which can take up a number of states governed by transition rules. Each cell can have a number of occupants, which determine its state. This model works on a very low aggregation level as all occupants of cells are explicitly accounted for by their status and "tendency". Each cell is considered to represent a single house, and these can either be privately owned or rented. As the potential to buy a house is a diverse function of the status of the potential occupants, this is considered to influence the overall status of the neighbourhood (through the inter-cell transition rules).

This dynamic approach builds explicitly on diversity of the system elements on a very low aggregation level. The set-up leads to self-organisation (see 2.3.15) in the system which makes the effects of individual action difficult to foresee, but this can be explored by using planning games such as this model. Again, the interactions between the system elements are an assumption of the modellers and not calibrated to a description of past observations. However, in this example - the mass immigration of Jews from the former Soviet Union into Israel - the situation had never before been observed in reality, so that there was no data describing such events in existence. Planning games based on heuristic assumptions can be used in these circumstances to explore possible futures, and might at the same time provide answers of a higher quality than an application of descriptive methods.

In Portugali and Benenson's model a spatial segregation of different inhabitant status groups was observed in nearly all cases depending on the migration rates of the different groups. This coincides on a qualitative level very much with the observed reality, although this model will not be able to make quantitative statements about the tendency of the system, as the definition of the system interactions is only coarse.

2.3.12 Master Equations

Master Equations have been developed in statistical physics in order to describe change in stochastic systems. The formalism of the master equation incorporates, in simplification, the probability distributions of all possible rates of change into the equation. The result of a Master Equation is therefore a probability distribution of events and processes (Allen, 1988). The shape of this distribution is of great importance for system behaviour. If there is only one peak, it might be sufficient to reduce the description to the point of highest probability or the average value, which results in a differential equation (see 2.3.13). If there exists a multi-modal distribution of solutions, however, the average will completely misleading, as its value will almost certainly be located in an area of low probability and the solution of highest probability ignores other possible, but less likely outcomes. These multi-modal solutions correspond to different possible pathways in reality, out of which only one is followed at a time.

On the other hand, if there is noise present in the system, e.g. there are fluctuations in the value of parameters and variables, it is possible that the system changes from one mode of behaviour into a different one given by the peaks in the probability distribution of the Master Equation's solutions. The fluctuations "test" the system for instabilities and if the fluctuation is large enough the system can be pushed from one mode of behaviour into a different one. Noise in the system is extremely important to explore these possible pathways by simulation in self-organising models (2.3.15). This issue becomes very important, because Master Equations are usually difficult to solve analytically, but easier to simulate in a computer.

The Master Equation approach accounts for varying rates of events and their probabilities to occur, but the variables treated by this formalism are identical (or at least normally distributed around an average in their attributes) by default (Allen, 1998). If the subjects treated by a Master Equation are individuals, there is no possibility to incorporate diversity in a population into the model, although the distribution of the behaviours of average individuals can be described.

This approach has been applied to urban systems in the field of population dynamics. The description of population dynamics with a dynamic, but aggregate model is an obvious choice for this methodology if a compact model is sought. The formulation of a generic migration model by Weidlich and Haag (1988) uses the system of coupled differential equations - derived from the average of the underlying Master Equation - describing the average values of migration flows. All settlements have defined attractivities depending on their size (representing economic activity and therefore employment opportunities) and on the past migration flows between them, which in a feedback loop represents the cultural links established by migration. As the model takes an aggregate view, it does not incorporate individual motivations to migrate.

Haag et al. (1992) apply the generic form of this model to the French system of cities over 50000 inhabitants. The main result of this application is the discovery of a strict hierarchy in the rank-size distribution of these cities, which follows a Pareto distribution. The degree of this hierarchy increases steadily over time, meaning that few large cities will continue to grow stronger than many smaller cities which might in fact suffer a decline in population. This approach is probably the most adequate of treating stochastic dynamic systems in the mathematical sense, but the implementation proves to be difficult. This is the reason why many master equation approaches use only the solution for the average values, instead of the distribution of solutions (if existant) resulting from the probability distribution of the noise term in the equations. However, this formalism has been originally developed for the description of statistical phenomena in physics, and it therefore appears to be difficult to incorporate cognitive factors occurring in social systems into the aggregate master equation approach.

2.3.13 Differential Equations

Differential equations describe the average rate of change in a system mathematically and can therefore be regarded as a special case of the Master Equation mentioned above. As opposed to difference equations, which return the change in a system over a discrete time interval, they give a continuous description for all points in time. Therefore this method is predestined for setting up dynamic models. Varaprasad and Cordey-Hayes (1982) present a simple exploration model based on differential equations. The aim of this model was to model possible implications of migration on the transport system in the London region. The description distinguishes between three spatial zones: the inner city, the outer metropolitan area and the rest of the south-east of England.

Ten differential equations are defined to project the traffic volume by mode, the population levels in each of the three zones and the commuting flows between these. The equations are derived from logistic growth theory, thereby limiting the maximum capacity of transportation system and residential areas and defining their basic tendencies. In order to test different scenarios, a number of variables were treated as exogenous to the model, such as transport costs, job supply, wage levels, fertility and mortality rates. This enabled the model on the one hand side to be calibrated to the observed values of the past and on the other hand to test a number of possible future developments. However, the extrapolation of such important parameters is not without risk for the quality of results obtained. From a systemic point of view, these variables should be regarded as endogenous to the system, because they depend at least partly on the development of the other elements of the system. Therefore great care has to be exercised when treating this part of the system as external to the model, as there might occur radical changes in these parameters other than those projected in the extrapolation.

This approach provides a dynamic alternative to earlier static models like the Lowry model. However, it might not always be possible to find descriptions of a system's behaviour, which can be expressed by differential equations. This applies especially for discrete events, threshold problems etc., where the system behaviour is discontinuous. Here a rule-based description might be superior to a set of differential equations. A system of differential equation can usually be solved analytically if the starting conditions and a number of constraints are known, leading to a continuous description of the system at each point in time. In a computer model this is usually not the case. As it is easier to determine the differential equations themselves and to start the model from some known conditions, instead of solving the set of equations, the issue of time steps in the actual algorithm become extremely important. If the time step is chosen too large the model can jump over crucial time periods in which the system undergoes critical change. This will make the model unable to detect such change, but this is one of the reasons why the model was built in the first place.

2.3.14 Fractals

Fractals are geometric objects that have infinite perimeter length, but finite area. These forms have been discovered in research into complex systems and non-linear dynamics (Gleick, 1987). While Euclidean geometric objects have an integer dimension – 1 for lines, 2 for areas, 3 for cubic objects, fractals feature a dimensional value of between 1 and 2. These objects have to be calculated with a recursive formula, which starts from an Euclidean form such as a triangle and in each step modifies the perimeter shape until its length approaches infinity. Fractals can be grown in this way, and this is the point where fractal geometry was applied as an analogue to the shape of settlements.

Longley, Batty and Fotheringham (1992) describe the use of fractal geometry as a tool for the analysis of geographical boundaries. It has been observed that the outer boundary of settlements gets extremely fuzzy as the settlement grows. This has been regarded as an indication that there might be similar processes occurring as in the growth of fractal forms. The fractal dimension can give an indication of how compact a settlement is and can allow for densities to be analysed.

As in the case of models describing the development of the built-up area with cellular automata, there exist a number of models aiming at reproducing the growth of settlements using fractal geometry (Batty and Longley, 1986; 1994). This approach does however rely exclusively on a method developed in physics for physical phenomena. As the observation units are plots of land, it is intended to reflect the results of human action indirectly. These plots are irrespective of their actual floorspace to area ratio considered either built-up or not, which puts plots of all uses and densities into one category. The dynamics of these models are based on a mechanism, which does not rely on any kind of representation of human decision making at all. The question remains whether the observed phenomena of fractal growth and urban structures are pure coincidence or whether there exist mechanisms in decision making which lead to boundaries being shaped like fractals.

2.3.15 Self-Organisation

In order to enable computer models to account for qualitative change of systems, the principle of self-organisation, which has been discovered in physics and chemistry, can be used. Models built on the principle of self-organisation avoid the limitations imposed by the use of average, statistical behaviour by explicitly relying on diversity and fluctuations in a system. Urban models based on this principle have been around for some time, and a representative selection of these will be discussed below. Before reviewing these models, it is necessary to find a definition for this important phenomenon. A definition for this phenomenon is given by Heuser-Keßler et al. (1994). It is derived from a heuristic, not a formal basis, but it probably is the most comprehensive definition of self-organisation currently available.

> *"(In this pre-theory stage)...self-organisation is defined as the spontaneous emergence, higher order development and differentiation of complex ordered structures which take place by means of feedback between the system elements in non-linear dynamic systems, when the systems are in a state of over-critical distance from static equilibrium through import of unspecific energy, matter or information"* (Heuser-Keßler et al., 1994, p. 40)

The central point of this definition is the emergence of ordered structures within a system, which change the way the system is working from the behavioural mode observed before the self-organisation event. The precondition to self-organisation is the import of energy, matter or information into the system, which keeps it far away from the static equilibrium.

The self-organisation event means that the system "locks" itself into certain modes of behaviour. The system now follows a new pathway, which is called an "attractor". Attractors can have three forms: point (steady state), cyclic (oscillation), or strange (deterministic chaos). A point attractor describes a state of static equilibrium. The system parameters do not change over time once the attractor is reached. Cyclic attractors on the other hand make the system go through periodic oscillations in the parameter values. Behaviour following a cyclic attractor repeats itself exactly on a fixed period in time. The last category, strange attractors, leads to the so-called deterministic chaos. Although the system can be described by deterministic equations, it is not possible to forecast exactly the state of the system parameters. System behaviour can be similar to previous times, and the system state at some point might be the same as before, but it cannot be determined whether the next state will be the same as at the last time this state was reached.

The variable space of a system usually contains more than one attractor. It is therefore possible that the system can change from one attractor (or mode of behaviour) to under the presence of noise. Different parts of the same system might as well obey different attractors at the same time. Self-organisation can take place already in systems governed by deterministic equations, if these are non-linear and – above all – there is noise present in the system. On the other hand the presence of noise makes the deterministic description a probabilistic one.

Although the models described below take an aggregate view by not accounting for individuals, useful experiences regarding the modelling of the built or natural environment have been made through their use. In particular the models of Allen (1982; 1997a; b; Allen and Sanglier, 1981; Allen, Engelen and Sanglier, 1983) have to be mentioned in this context. These models were the first to allow for qualitative change within the spatial configuration of a region, and therefore overcome the limitations to the single static configuration of Christaller's (1933) conceptual model mentioned in Section 2.3.1.

Starting from the fact that economic activity at one point can only support a given number of inhabitants, the idea of a carrying capacity is introduced. The carrying capacity limits the maximum number of inhabitants, which can be supported by the economy to a maximum value. This leads to a logistic growth curve for possible populations if this type of economic activity is present. The idea is enhanced by introducing several levels of economic activity, which are not present in the beginning, but can come into existence in a place if there is sufficient demand.

The price for a given type of good decreases with growing demand. This mechanism aims at incorporating the effects of economies of scale. The distances between places of demand and supply impose transportation costs on the consumer. The economy can now react to demand and increase or decrease production, which leads to changes in employment and population resulting in changes in demand for goods in these places.

In early versions of these models, new economic functions are introduced at random at certain locations, which is intended to reflect the appearance of innovations leading to an increased carrying capacity. The number of functions present in one location, the price for goods and their distance from the consumers determine how attractive a given location is for the consumer. It has to be pointed out that the parameters determining the reaction of the economy in adapting supply to demand and the elasticity of demand are diverse, so that different solutions can be reached as the models are run more often.

Further diversity is introduced in later models focusing on intra-urban evolution. Here several types of industry (exporting, services, and two types of production for the local market) and therefore different job requirements are given. The industries can locate themselves according to the broad principle above, but in this case there are also effects of crowding present, which limit the amount of economic activity in all locations.

More important in our context is that the model allows for two types of inhabitants: blue-collar and white collar workers respectively. This reflects the disaggregation of the economy in the model as the different types of production have different requirements on their workforce. The population has to make choices about where to live and where to spend money. This is realised using relative attractivities of locations, which account for the distance from residence to workplace, crowding and the presence of members of the same group in a given location. The groups have differing preferences captured by individual parameters for all aspects of the location problem.

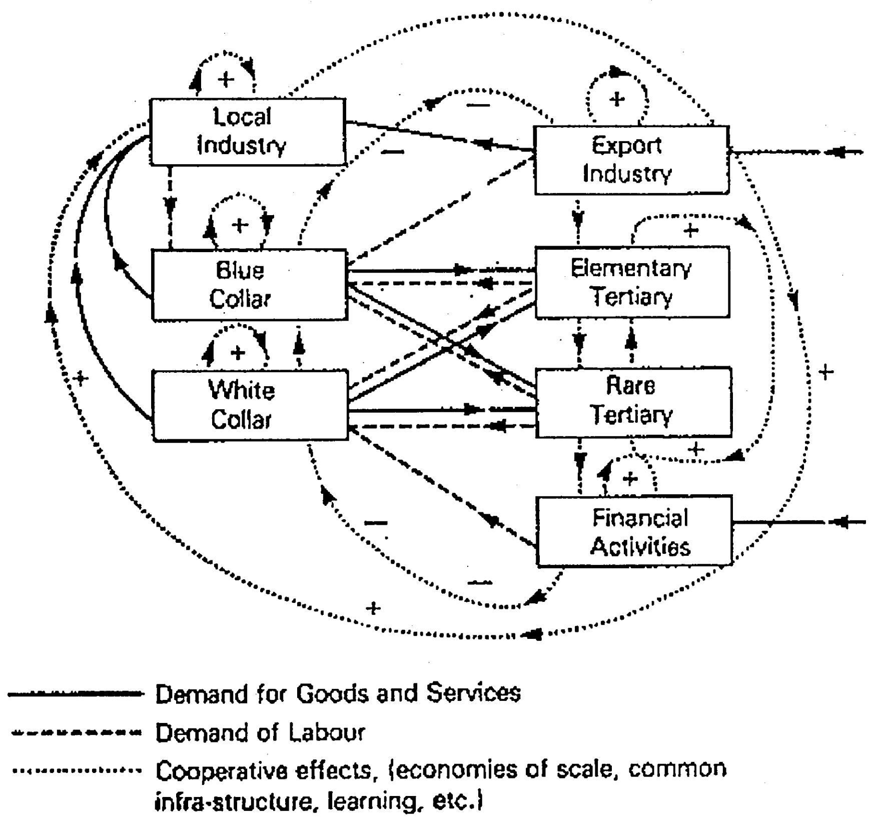

Figure 8: Structure of a Self-Organising Urban Model (after Allen, 1997b, p. 193)

This kind of model can be used to generate scenarios. For instance, one later version was used to try to explain the city structure of Brussels. The results obtained are qualitatively consistent with the observed reality with regard to the distribution of population and economy. On the other hand it is possible to intervene in the system during run-time in order to test different possible realities. This leads to a multitude of configurations of the city structure, none of which has been observed in reality. On the other hand, the capability of the model to generate the qualitative structure of reality from a set of basic assumptions on the nature of the processes taking place shows that the model is valid. The interventions in the system only change parameters without changing the structure of the model, so that it must be assumed that the alternative solutions are realistic outcomes, which might happen in reality as well, if the preconditions for this development are met.

This approach to modelling urban and regional systems shows that a diverse population in the model can lead to many more qualitative insights into the nature and possible development of an urban system than a conventional aggregate or descriptive (statistical) approach can deliver. Although the decision processes of the population are represented only indirectly through specific attractivities, a much larger variety of processes can be accounted for than traditional models are able to display. An even further disaggregated model will be able not only to account for the implications of behaviour on the city structure, but also for the effects of the city structure on the individual's lifestyle.

Other approaches using self-organisation as their main paradigm can be found for instance in Beaumont, Clarke and Wilson (1981) who describe a model of a similar structure to the one shown in Figure 8. Dendrinos and Mullally (1981) propose an aggregate model of city size built on the principles of non-linear dynamics. The focus of this model is shifted towards the exploration of different time scales of change in an urban system. They conclude that fast change occurs almost exclusive in conjunction with radical structural change (bifurcation of pathways) which shifts the system from one attractor to a different one.

A model of urban evolution based on the Lotka-Volterra predator-prey dynamics is the core of an ecological model (Dendrinos and Mullally, 1985). The competition for population and space are regarded as the driving forces of development in an urban system. A two-variable model is outlined for both the inter-urban and intra-urban case. The inter-urban model has the form

$$\frac{dx}{dt} = \alpha(y - \bar{y})x - \beta x^2$$

$$\frac{dy}{dt} = \gamma(\bar{x} - x)y$$

$$\begin{aligned}&\textit{with}\\ &x : \textit{relative population size}\\ &y : \textit{real (deflated) per capita income}\\ &\bar{y} : \textit{average income level}\\ &\bar{x} : \textit{the city's relative carrying capacity}\end{aligned}$$

Equation 6: Dendrinos and Mullally's (1985, p. 50) Urban Evolution Model

This appears to be an extremely simple model, but Dendrinos and Mullally show that the required parameters for can be found for a sample of American cities over long time periods. This model gives a description of city growth, but it cannot generate the crucial parameters of average income and carrying capacity, which would help to explain why the development of the cities took place. In this sense this approach has the same explanatory power as other models built on analogues, for instance the static gravity model.

Kahn, Deneubourg and de Palma (1981) present a transportation mode choice model based on similar principles. As opposed to traditional statistical or discrete choice models, this model is of explanatory nature and makes simple assumptions on people's behaviour and the economic necessities of public transport operations, which are then to be tested using the model. Two modes (car and bus) are regarded as competing with each other. Each of these is characterised by a set of attributes, in the case of the car its speed and for public transport the level of service, the fare level and the information on the service. The precise equations read as follows:

$$\frac{dx}{dt} = \frac{DA_1}{A_1 + A_2} - x$$

$$\frac{dy}{dt} = \frac{DA_2}{A_1 + A_2} - y$$

$$\begin{aligned}&\textit{where}\\ &x: \textit{car users}\\ &y: \textit{bus ridership}\\ &D: \textit{demand for transport (assumed constant)}\\ &A_1: \textit{attractivity for car}\\ &A_2: \textit{attractivity for bus}\end{aligned}$$

Equation 7: Ecological Modal-Split Model (Kahn, Deneubourg and de Palma, 1981, p. 1164)

A third equation describes the changes to the bus system:

$$\frac{dL}{dt} = vy - KL$$

$$\begin{aligned}&\textit{where}\\ &v: \textit{fare charged}\\ &K: \textit{maintenace cost}\end{aligned}$$

Equation 8: Change in Bus Service Level (Kahn, Deneubourg and de Palma, 1981, p. 1164)

The specific attractivities are calculated as follows:

$$A_1 = v_1 \alpha_1 x$$
where
v_1 : *automobile speed*
α_1 : *measures the strength of imitation term* $\alpha_1 x$

$$A_2 = \frac{L}{v^2}(\theta + \alpha_2 y)$$
where
θ : *publicity*
α_2 : *measures the strength of imitation term* $\alpha_2 y$

Equation 9: Attractivities of Modes (Kahn, Deneubourg and de Palma, 1981, p. 1164/5)

As this approach is a system of non-linear differential equations, noise was introduced into the system to make the system move from one attractor to another. In this way critical values for the system can be found, and configurations be explored which are not obvious from the outset. In a later paper (Kahn, de Palma and Deneubourg, 1985) use a similar approach to outline the use of such an exploration model in order to simulate the effects of fluctuations of demand on public transport operations under market conditions.

The basic difference between the approaches by Kahn, Deneubourg and de Palma and those of Dendrinos and Mullally is that the former is used to describe the evolution of a system of cities, whilst the latter does not make an attempt to reproduce reality, although they use practically the same formulation. This model examines the level to which the underlying assumptions of the model are qualitatively consistent with processes observed in reality.

2.3.16 Agent-based Approaches

A recent technique for the representation of individual entities in a computer program is the use of so-called agents. Agents are discrete entities, which have a set of given properties and associated processes. This technique has been applied to urban models, two of which will be mentioned here. Sanders et al. (1997) present an agent-based approach for the simulation of the dynamics of a system of cities. Each settlement is considered an agent and has a set of rules, which determine the transitions from one state to another according to the global conditions. Each settlement can potentially acquire a set of functions, for which a certain level of wealth or population must be present. Information on demand and supply of goods can be passed between agents, so that trade and other interactions between them can evolve.

As in their other simulations on urban systems (see 2.3.12), a hierarchy of places comes into existence, although initially conditions are approximately symmetric. Fluctuations are found to have great influence on local configurations, but the global hierarchical patterns are qualitatively the same for all simulations. It is claimed that this approach is much more flexible than the alternative use of differential equations, which have to be defined to encompass the entire system, while local rules for agents can be defined as required.

An extension to the cellular automata model of Portugali and Benenson (1995) is introducing free agents into a cell space (Portugali and Benenson, 1998). The agents explicitly represent inhabitants as opposed to the previous model, which accounted for the population indirectly as a property of houses, which could hold a number of occupants. The agents have a "cultural identity", and an attitude towards other identities. The modelling approach is based on the theory of cognitive dissonance (Portugali, Benenson and Omer, 1997 after Festinger, 1957) between the individual's intentions and actions. If there exists a difference between the individual's idea of the world and its actual perceived state a "cognitive dissonance" is generated. This cognitive dissonance leads to either change in the individual's behaviour or to a revision of the individual's belief system in order to eliminate the cognitive dissonance. Here the cognitive dissonance of agents, which cannot change their behaviour by moving to another place leads to the adoption a new cultural identity. The cognitive dissonance can therefore be regarded as the driving force of the model. This is interesting insofar as the driving force of the model does only indirectly refer to any explicitly stated needs or motivations of the individual.

It is clear that the properties of the agent-based approach makes this methodology very useful for micro-scale-approaches, although we have seen that also aggregate descriptions can be made with them. The use of locally active rules can produce behaviour that otherwise in an aggregate global approach might have been overlooked. Furthermore, it is possible to account for individual decision making in a model without assuming global patterns beforehand. Agents have also quite extensively been used for the simulation of social behaviour, and these approaches will be presented in Chapter 4.1.

2.3.17 Economic Theory in Urban Models

Most urban models use economic parameters in order to account for aggregate patterns of decision making. The gravity model, as an example, uses cost parameters for determining how many people would travel how far. This "cost" can also be interpreted as travel time, but in a number of models also a mixture of both concepts can be found. The model of Dendrinos and Mullally (see 2.3.15) assumes the relative income level as the single choice parameter for migration. A predominance of economic considerations can be found in many discrete choice models as well. All these approaches suppose that people are acting in a rational way in order to maximise their material well-being. This might be true for companies, whose overall aim it is to maximise profit, but it appears to be questionable in how far this is true for individuals. However, economic parameters are very easy to measure and to compute, which makes these predestined to be used in computer models.

A model outlined by Echenique et al. (1974) attributes the micro scale of behaviour exclusively to economic issues. This model combines the Lowry-type mechanism of employment generation described in Section 2.3.5 with a purely economic mechanism of residential choice based on the income to cost ratio for a household only. The model's transport module determines the mode choice on the basis of the mode's costs, and trip frequencies are calculated using the disposable income of a household. While it is clear that the economy works with these parameters, which in return have many consequences for the individual's decision space, it is far from certain whether it is possible to apply these principles on their own to individual behaviour. It is, however, possible to calibrate a model to economic parameters (as it has been dome by Echenique et al.), but in that case it is questionable whether the change of economic parameters in reality would produce the same results as in the model. It is almost certain that the economic parameters have to be regarded as an analogue to the decision structure in reality. The model loses the explanatory value, which had been put into the approach and can only give a description of the system at some point in time.

Cognitive approaches on the other hand (see 2.3.9) account explicitly for the decision processes of the individual, which will almost certainly have economic aspects. The difference is that the explanation for observed processes is derived from the cognition of the individual and not from a theory which has been developed for a different range of phenomena.

2.4 Critique

The brief assessment of the methods and concepts used in urban modelling confirms the gap identified in the classification of existing models. The reasons for the existence of the gap in explanatory dynamic models which are able to represent individual decision making are manifold. At one end the first models assumed that equilibrium conditions would prevail in a mature system, so that any dynamic approach would in the end deliver the same result as a static one. Furthermore, the interest in macroscopic phenomena led to the construction of models on the macro-scale, which would not be able to incorporate individual decision making. As there was little computer power available at the time, there was also a necessity for computer models to be as compact as possible, which lead from this side to macro-scale models. The extrapolation of statistical behaviour – which gives a description of past behavioural patterns – cannot explain why these patterns exist.

From this point it is deemed desirable to build a model which might be able to explain phenomena instead of just describing them. This would be the place of a model placed in the gap in the taxonomy. By reversing the traditional approach of extrapolating existing patterns of behaviour, a theory-based model of individual behaviour could potentially deliver the results of the top-down methodology without assuming the existence of these phenomena beforehand.

The first step towards such an approach is the view that the development of urban structure is determined by the behaviour of the people who inhabit that particular city. It is people who live in residential areas, who go shopping and use recreational and cultural facilities like parks, theatres and cinemas. At the same time those very same people work in industry and for the facilities mentioned before. The inhabitants are at some times part of larger aggregates called companies (in the case of work) or are members of societies and clubs (in the case of leisure activities). In this view we cannot really distinguish between institutions and inhabitants, because it is those inhabitants who form the institutions.

It is these people who have to get from one place to another in order to get to work, to go shopping or to socialise. People's needs and desires are the reason for traffic between places in the city. The idea to integrate all these areas in one single model leads to a multi-purpose model of people's everyday life. Such a model can account for urban development, economic activity as well as traffic patterns at the same time, because all those areas are results of human activity.

Moreover, it might be possible to model the implications of environmental change on individual behaviour and the population's lifestyles, which have not yet been treated with a model. The dynamics of behaviour will lead to a model of changing (i.e. learning) individuals, which have feedback effects on macroscopic (i.e. group) behaviour as well as on the environment.

Of course it is not an easy task to formalise human behaviour in a computer model; it is a task on which psychology has worked since it has come into existence. On the other hand it appears to be possible to model people's motivations and behaviour with a more cognitive approach which can replace the principles of utility theory and social physics, which we have already criticised in this chapter. The model would naturally have to make abstractions and would only be able to deal with a limited number of phenomena, but this is a restriction we find in the traditional models as well.

Such a model of everyday life would have to be a dynamic model relying on a set of intrinsic driving forces. The existence of these driving forces would have to be the central assumption of the model, but the results of that model will clarify the validity of the approach. The dynamic model will rely on statistics only to define the initial conditions as the driving force will replace the regression formulas, which are used in models based on statistics. The dynamic approach eliminates the restriction to distributions of behaviours observed in the past and can generate new behaviours as possible scenarios of the future. This approach leads automatically to a qualitative treatment of the system. The calibration of such a model is not considered, because of the loss of explanatory power observed in the approaches using social physics, which reduced the theoretical basis of the approach to a mere description of observed patterns.

Before examining more evidence on the feasibility of a bottom-up approach to urban modelling, consideration is given to the inherent limitations of computer modelling.

2.5 Chances and Limitations of Computer Models

The value of computer modelling is undisputed, if this technique is used in a way, which accounts for its inherent limitations. Much has been said about the belief in the results of the infallible computer, which means that the programs, which have produced these results, have to be looked at in a very careful way. A computer program will always reproduce the assumptions underlying the conceptual model used, and therefore never come up with something conceptually new, although the results of these computations might shed light on the way the real system might be working.

A computer will probably never deliver a forecast of the future, as has been shown by Wolpert (1997) in his incompleteness theorem for forecasting the future. No matter how powerful the available computer, it will not give a forecast the future in a quantitative way. The value of simulation as a method of qualitative scenario building on the other hand is undisputed, it has even led to an attempt to formalise the method of simulation in a formal theory (Rasmussen and Barret, 1995). Simulation is generating more and more new scientific knowledge on problems, which cannot be investigated, because real-life experiments are not possible, too expensive or too dangerous to be carried out in the real world.

At this point it is necessary to clarify some basic properties of computer models. Firstly, a computer model is always working in a closed universe. The program, which produces the experimental data, is defined from the outset, and it cannot be changed during the duration of a simulation. The assumptions used to write the program will be reflected by the results. This means that the often-claimed emergence of new features in models of complex systems is not emergence in the sense that something "new" has been added to the model's universe. The observed phenomenon is a result of the rules, which govern the model (Hiett, 1997). The system however can find on its own modes of behaviour, which have not been observed before, therefore adding to the experimenter's information on the system.

In the case of complex systems we encounter one more limitation which this time is imposed by our own ability to analyse and to conceptualise real-world systems. Can we correctly map reality into a computer model, which - if we feed it with incorrect data - will rather confuse than inform us? The reverse is true as well. A "correct" model might challenge our ability to analyse the model's output. This "complexity barrier" has to be considered at all times when modelling complex systems. The trade-off between correct data output and useful information on the investigated system leads inevitably to abstractions of which we do not know whether they wipe out detail crucial to the aim of the modelling exercise.

Why, the reader might ask, is it then still worth exploring the future (or the present) with inherently incorrect models? The answer is simple. The technique of scenario building can provide invaluable information on how the system might behave if constraints are changed or certain fluctuations coincide at some point in time. It is possible to find new attractors for the system which otherwise might not be discovered before they happen and take us by surprise. One example for this will be described in Chapter 4.5.1. It is Allen and McGlade's (1987) model of the fisheries off the coast of Nova Scotia in Canada, where the model could provide the researchers with answers which were not obvious from the outset of the investigation.

3 Investigation Plan

The classification of models and the critique given in the previous chapter gives an overview of how a new urban model should be set up. For an implementation it was necessary to review more methodological evidence on which such a model was to be based. The gap identified in the classification scheme calls for an explanatory, dynamic model incorporating individual decision making. The bottom-up approach necessary for the incorporation of decision making would allow for a multitude of possible subject areas, as individual decision making plays a role in all subject areas reviewed. For this, information on four aspects of decision making had to be gathered:

- Driving forces of behaviour
- Individual cognition
- Dynamic change of cognition and decision making
- Implementation methods

The major approaches and methods to these areas are presented in Chapter 4. In a next step, critique and evidence had to be combined in a conceptual framework to form the theoretical background for a computer model. Because the resulting framework was designed to encompass the entire urban system, not all of the framework's elements could be implemented within the size of the project. Therefore the scope of the model and the extent of the simulation exercise had to be limited. This meant for the model that:

- The environment was to be implemented in the simplest way possible. The project focused on the aspects of individual behaviour in the system.
- Some assumptions had to be made on the nature of the dynamics in the system, so that not all interactions were subject to change over time.
- The agents representing individuals were interacting only indirectly via the environment with each other. The formation of networks was not supported.
- Time scales were considered only on one level, so that the agents were using all rules at all times.

The emphasis on the methodological aspects and the restriction to individual behaviour meant that the main purpose of the model was to explore the basic properties of adaptive agents. This gave a first impression of the usability of the framework and would serve as a qualitative validation of the modelling framework. The model was designed to handle a number of autonomous agents. The agents were to rely on one or more Fuzzy Logic rule bases to incorporate a cognitive model. The agents were to be motivated by a set of intrinsic, invariant needs measured by a set of budgets. Furthermore, the rule bases had to be adaptive, e.g. the initial rules (as combinations of input parameters, operators, and output parameters) had to be changed during run time of the program. Changing rule parameters during run time is not a problem, but covering an entire parameter space with all thinkable rule combinations would usually be a problem, if defined in terms of mathematical equations. The model would have to be able to represent a spatial structure as well as a first step towards a full representation of an urban system. The spatial elements are supposed to obey their own dynamics as well as to react to patterns of usage by their inhabitants.

The model was testing whether evolutionary properties such as a diverse pattern of rule sets for agents in different environments would emerge from within the model. Assuming that stable patterns would be due to self-organising processes, one might expect radical change in the way the agents' rule sets are defined from time to time. But would the system develop rule sets, which - in a changing environment - would lead to long-term successful behaviour of the agents?

As a side line, a definition of successful behaviour had to be found as well, because it was not clear from the beginning what attributes of an agent were to be included in the "measurement" of success in an evolutionary context. The model was neither optimising nor of purely economic nature, which would have made it easy to find measures of success, but it was simply designed to display patterns of everyday behaviour driven by intrinsic goals of the actors. The direction of the system's development was not clear from the outset, and nor was the method how to define a comparative measure between the agents.

The next point was to involve the comparison of different strategies for rule adaptation and their performance. Would the model favour adaptivity (and if so - what kind of adapting strategy?) over static rules starting from a conventional rule based system with random, static rules? Would the agents "learn to learn", and if so, what would they learn?

Finally, the methods used to build the model - and the model itself - had to be compared to existing urban and transportation models and their methodologies. The points of critique of traditional urban and transportation models were compared to the results of the methodology developed and the model's results in order to assess whether it can improve on these.

4 Models of Human Behaviour

The main point of critique to existing urban models is that these do not give any representation of individual decision making. Therefore these models cannot explain any aspects of how individuals influence their environment, and how changes in the environment lead to shifts in behaviour and attitudes in the population. However, other fields of research have developed methods, which can be used to simulate individual behaviour. The four areas of interest have been identified in the previous chapter as

- Driving forces of behaviour
- Individual cognition
- Dynamic change of cognition and decision making
- Implementation methods

Here some key concepts will be reviewed, from which the conceptual framework for a computer model of individual acting will be derived in the following chapter.

4.1 Herbert Simon's Theory of Organisational Behaviour

Herbert Simon's theory of organisational behaviour has already been mentioned in Chapter 2.3.9. In order to elaborate the underlying concepts of the theory, a broader discussion is given here below. Although the theory has been developed in the 1950s and 60s, it has only recently been applied to models of human behaviour.

Simon concluded that organisations, being composed of individuals, have most of their behaviour determined by their members. Even if the members of an organisation want to achieve an optimum efficiency, they are constrained by the fact that humans are incapable of correctly perceiving and analysing the situation they are in. Reality proves to be too complex for an accurate analysis. Simon coined the term "bounded rationality" for this phenomenon.

Having acknowledged the fact that reality cannot be analysed in a straightforward way, what are the conclusions to be drawn for decision making? If our picture of a given situation is inevitably incomplete, how do people decide what to do? Here it was suggested by Simon that humans tend to choose solutions which are not necessarily the best (which might even be acknowledged by the decision maker) on an absolute scale, but "the next to best known" (Simon, 1981) alternative is chosen. This has been called "satisficing" behaviour, for it describes a choice, which is not optimal, but can be trusted "to do the job". The outcome of such a choice might be even better than that of a supposedly superior alternative, which has not been fully understood in all its complexity. Simon's finding coincide with what Forrester (1969) stated about the performance of social systems.

> *"Complex social systems tend toward a condition of poor performance. Their counterintuitive nature causes detrimental design changes. Also, the opposite direction of short-term and long-term responses leads to policies that produce a less satisfactory system. For example, a particular change in policy may improve matters for a year or two while setting the stage for changes that lower performance and desirability further in the future. But the natural interpretation is that good resulted from the change and when*

matters become worse the original effects are redoubled. The intensified action produces another short-term improvement and still deeper long-term difficulty. Again the complex system is cunning in its ability to mislead." (Forrester, 1969, p. 112)

The urge to choose supposedly superior, but badly understood alternatives over well known, but seemingly inferior solutions appears to be very prominent in public sector decision making, whilst economic enterprises seem to function more according to Simon. Urban and transportation planning might serve as one example for this "bounded rationality". The promotion of the motor car during the 1950s and 60s in Europe was regarded as a positive development by the planning authorities, as it would increase the standard of living of the population by saving time over the use of public transport. What was very badly understood were the repercussions of such a mobile population on the urban structure. For instance, the very idea of Greenfield sites for shopping centres or industry did not get any consideration, because it was unimaginable to the decision makers (although the same process had already taken place in the USA before that). Time Budget studies (see 2.3.7) have additionally shown that people in industrialised countries spend even more time in traffic than before, or that at least the time use for transport is near constant. The intention of the improvements in the transportation system has been contradicted by the population's changed behaviour.

Simon's theory proves to be very interesting for a model of human behaviour, as it appears to capture two very important aspects of individual decision making: Satisficing behaviour and bounded rationality in perception. Interestingly enough, these two principles have recently been introduced into cognitive science as well as evolutionary theory (4.4.4) as we will see further on.

4.2 Agent-Based Simulations of Human Behaviour

The methodology of so-called autonomous agents has already been mentioned in Chapter 2.3.16. The main aspects of this methodology will be summarised again, as this technique is considered a key concept for modelling individual decision making and its effects on systems as a whole. Agents are equipped with a set of given properties and characteristics. Agents have their own rule set determining how to react to a given situation. Using agents in a simulation means building a system from the bottom up because agents use local rules to produce the whole system's behaviour. Agent-based models are therefore extremely useful tools to explore phenomena of self-organisation.

Agents share a common environment, which is changed by their actions. The state of the environment determines in return the response of the agents. As opposed to the more conventional micro-simulation models, which use statistically estimated probabilistic equations, agent-based models usually use very simple rules for each agent. The fact that these rules are applied locally can make a big difference for the overall behaviour of the system, because the use of the local environment introduces fluctuations into the system, the main cause for self-organisation in a non-linear system.

Agents are a particularly useful methodology for modelling social systems. From a conceptual point of view agents already bear a certain resemblance to real people, only that the average agent is for obvious reasons much simpler than an average person. There exist a number of recent agent-based approaches to the simulation of social systems, of which only a few can be introduced here. Agents have been used to simulate fisheries systems (Bousquet et al., 1994, LeFur, 1995 and in press), interaction within societies (Doran et al., 1994; Doran and Palmer, 1995), or markets (Kirman, 1994).

One especially interesting model is Epstein and Axtell's (1996) "Sugarscape" model. They define a set of oversimple agents which require a substance (here called "sugar") in order to survive. "Sugar" grows at a certain rate in the space the agents inhabit. If the agents fail to find "sugar" they either die or have the choice to migrate to some other point in the "sugarscape". If they choose to migrate they can rely on a limited field of vision, which means that they have only very limited information on the nature of their surroundings.

As dying agents are removed from the model population and surviving agents produce offspring at a rate depending on how much sugar they can accumulate, a population ecology is generated. Diversity within the population is introduced in the form of different metabolic rates. The growth rate of the "sugar" is varied spatially and temporally to simulate "seasons" which results in the migration of agents. Other simulated phenomena include the formation of "tribes" and social networks, emergence of trade when a second commodity apart from "sugar" is introduced, or the highly realistic distribution of wealth within the population of agents.

This makes the "sugarscape" one of the most advanced simulations of an artificial society to date. The model does not only cover daily life, but features also ageing agents, reproduction, trading and other social behaviour. However, the rules governing the behaviour of the agents are the same for all agents and do not change over time. Therefore the agents are therefore not learning, and they might not adapt to very challenging new situations. The agents have as well only one "need": They have to feed on the "sugar" they find, otherwise they die. This makes the representation very simple, but Epstein and Axtell intentionally kept the model at this abstract level in order to focus on the behaviour, which can be generated already from this very simple model.

A similar agent-based model can be found in Doran et al. (1994). This model focuses on how power structures might be generated in a hypothetical society. The set-up of the agents is more sophisticated according to the aim of the model. The agents are composed of several sub-models to represent the agent's perception of the social and physical environment it is in. Most interestingly the agent's mental models do not represent the actual environment, but the individual's view of the world. Local rules make the agent then act according to its perceived environment. This is one of the first models, which explicitly incorporates what has been described by Simon (see 4.1 above) as a "bounded rationality". The cognitive and behavioural rules governing the agent are however the same for all agents, and these rules do not change over the time of a simulation.

Nevertheless, the above model illustrates that social relations and social behaviour such as the emergence of social groups, and altruistic as well as egotistic behaviour have to be regarded as emerging from within a society. We can conclude that, at least in theory, there is no need to predefine social hierarchies, once a sensible cognitive model for the agents is used.

Models of adaptive agents have also been built. One example for this is an approach to the simulation of kinship structures within a hypothetical society (Parisi, Cecconi and Cerini, 1995) using artificial Neural Nets (see 4.7.1) as agents. These Neural Nets are "trained" during the model run and therefore change their input / output relations. In doing so the connection weights between input parameters are changed, resulting in a different inference process (and output) for the same input configuration. This can be regarded as equivalent to changing the agent's rule set.

However, many agent-based approaches have been criticised for being immanently tautological, e.g. the model is reproducing only the assumptions on which it is based and does not show "real" emergent properties and behaviours. This point is very closely related to the critique that social simulations very often use pre-established social architectures as well as "hypercognitive" agents (Conte and Castelfranchi, 1994). A "hypercognitive" is an agent omniscient and fully aware of his environment. Social action is conceived of only in terms of communication with other agents. Social scientists have also been criticising models of social systems for lack of concepts central to social theory, such as "meaning", "action" or "structure". In this sense, the explanatory power of such models is reduced to the display of behavioural patterns, which could be generated by just about any mechanism. Still, such models can be used in a metaphorical sense for the explanation of real-life phenomena.

Viewing social relations as emergent properties finally leads to a radically bottom-up approach using "intelligent" (e.g. adaptive, learning) agents with endogenous goals placed in a common world in which social relations would evolve rather than be predefined. This framework reflects recent advances in cognitive (such as Varela, Thompson and Rosch, 1991; see 4.6) and social science (such as Giddens' (1986) theory of structuation or Luhmann, 1987). Regarding the agent as well as his environment as co-evolving and mutually dependent parts of the same system serves this purpose much better than the classical "cognitivist" top-down approach treating cognition as representation of an absolute and independent environment.

There appears to be a wide scope for improvement of the existing agent-based models, especially in the field of social simulation. Most approaches use a single driving force for the agents, e.g. the agents have only one motivation to act in a certain way. As we will see in Section 4.3.3, the review of the concept of human needs reveals that instead, life (even if it is simulated in a computer) should be regarded as a careful balance of several needs, demands and objectives. We are therefore left with a multicriteria problem resulting in trade-offs and compromises between several, often conflicting, goals. A second area where agent-based models could be improved concerns the usually time invariant rule sets governing the behaviour of the agents. If agents are designed to represent people, it appears plausible that the agents should change their rules over time - that is to learn. The great problem with an implementation of learning in a computer model is that it remains unclear how learning works in reality, let alone how to model learning. However, much might be learned about the process of learning and adaptation by experimentation with such agents in a computer model.

4.3 The Concept of Human Needs

As utility theory and the concept of optimisation as well as statistical observation do not deliver any valid concepts on long term human behaviour, we have to turn elsewhere to find a point on which to anchor a model of human behaviour. The idea that all humans possess a set of finite intrinsic needs can be very helpful for this. There exist approaches to this concept which are partly rooted in humanistic psychology, partly in economic development theory. We will have a look on both of them.

However, any need system has to deliver a set of properties which are invariant over time, otherwise it is not possible to build a dynamic computer model on it. If it is possible to find invariant parameters, which govern human behaviour, the problem of modelling the latter is very much simplified. As outlined in Chapter 2.5, invariant elements are the cornerstone of any computational model. They are needed to provide the basic structure of the algorithm, which has to remain as it, is over time. We have to point out again that the universe of a model is always a closed one. Changes to the model universe can only be expressed in terms of the variables, which have been defined before the model is run.

4.3.1 Maslow's Position

Maslow (1954) proposed a classic approach to the concept of human needs. He postulated that all humans have a given set of intrinsic needs. These needs form a hierarchy so that in order to fulfil needs of higher order it would be necessary to satisfy lower order needs before. The most basic needs are obviously the ones concerned with the physical requirements of staying alive, like eating, drinking and sleeping. Once these requirements imposed by nature are met, a need for safety and security would arise. This would lead to communities, who care for their members. A next step would involve a need for love and belongingness, followed by a need to be recognised and esteemed by others. After this, humans would aim for self-actualisation, meaning that they would try to fulfil their potentials and interests in creative, humanistic or other pursuits. On the top of the hierarchy he put aesthetic needs, leading to the development of arts or the creation of a pleasing environment. The hierarchy shows then as follows:

a) Physiological needs,

b) A need for safety,

c) A need for love and belongingness,

d) An esteem need,

e) A need for self-actualisation, and finally

f) Aesthetic needs.

The hierarchy was thought to be a conceptual model and represented a novel approach to the problem of what motivates people at the time. It is, however, in the first place a philosophical concept, a concept that does not appear to be very practical to implement. Maslow's idea of a hierarchy of needs has been criticised on a number of occasions. His approach is regarded as utilitarian, and it is based on a materialist ideal. The way the needs are formulated matches probably the traditional western ideal of a "fulfilled life", but we have seen that in other cultures, or in circumstances of crisis people find ways of fulfilment without necessarily having all of Maslow's needs satisfied. In reverse, it has been argued by Landauer (1972) that the satisfaction of these needs would not necessarily produce a happy person, as countless examples of rich people, with all the potential for the satisfaction of the above needs, show.

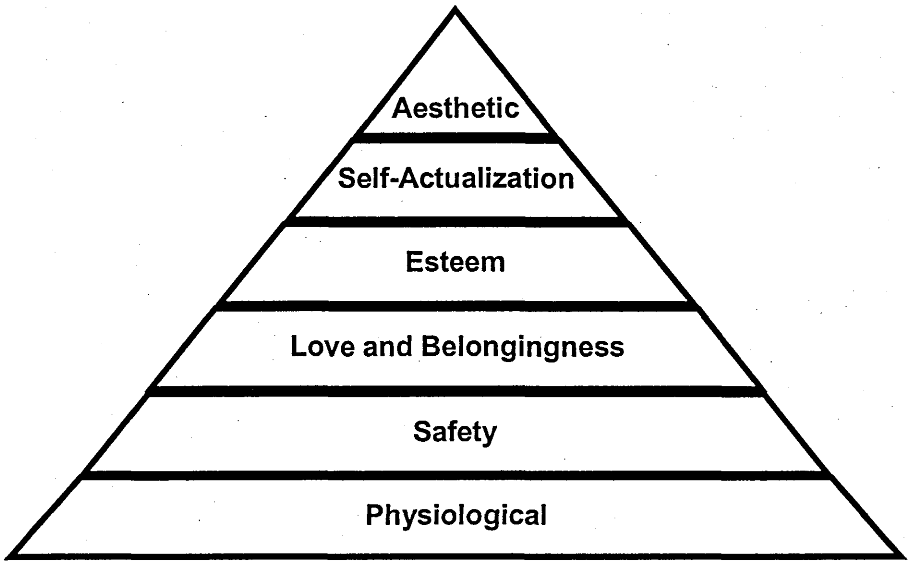

Figure 9: Maslow's Hierarchy of Needs

Although there are not any known examples of actually using Maslow's hierarchy of needs in a computer model, the theory is used, for instance, in nursing. In the urban context, an attempt to translate the hierarchy of needs into urban functions has been made by Walmsley (1988; after Faulkner, 1978). This approach attributes the satisfaction of Maslow's needs to certain social institutions (see Table 4). However, this does not relate the hierarchy of needs to people's observed behaviour or their motivations to behave in a certain way.

Need category	**Description**	**Attributes of the urban environment associated with the satisfaction of needs (examples)**
1. Physiological	Provision of food, shelter and health care	Retailing/wholesaling system distributing food, clothing and health supplies Health care clinics and hospitals Essential services (water, sewerage, power) Dwellings
2. Safety-security	Protection from physical harm and intruders. Privacy and absence of overcrowding Protection of property	Fire and police services Road safety Absence of noxious environmental elements (pollutants) Residential areas that ensure privacy
3. Affection - belonging	Harmonious relationships with other members of the community Identification with and acceptance of groups within the community	Facilities for community organisations (meeting places) Physical layout of neighbourhood such that cooperative and harmonious inter-family relationships are fostered Physical identity of the neighbourhood
4. Esteem	Status and recognition by others in the community	Opportunities of home ownership Prestige of neighbourhood
5. Self actualisation	Role relationship vis á vis others Realisation of one's potential Creativity/self expression	Built environment that facilitates creativity and self-expression Employment opportunities and community organisations that enable the use and development of skills
6. Cognitive/ Aesthetic	Provision of educational experience, intellectual stimulation and experiences Aesthetically appealing events and phenomena	Educational and cultural facilities Recreational facilities Aesthetically appealing built and natural environment

Table 4: A Typology of Urban Needs (Walmsley, 1988, p. 60/1, after Faulkner, 1978)

Taking all the criticisms into account, the idea of a set of intrinsic human needs as a motivation remains a very intriguing one. In the next section the feasibility of using Maslow's hierarchy in a computer model of human behaviour is explored.

4.3.2 A Spreadsheet Model

A first attempt to base a model on the concept of intrinsic human needs has been made using a simple spreadsheet model. The main purpose of this spreadsheet model was to explore the dynamics of a set of system driving forces, represented in this case by Maslow's (1954) hierarchy of human needs (see 4.3.1 above).

Although Maslow never claimed that the needs identified by him form a fixed hierarchy, this set of needs cannot be seen as a fixed structure at all, but rather has to be regarded as a dynamical system. A simple interdependency analysis (Table 5) shows that the needs actually interact with each other. Therefore a strict hierarchy of needs cannot be regarded as the driving force for modelling the behaviour of individuals.

⇓ depends on satisfaction of:	physio-logical	safety	love and belonging-ness	esteem	self-actuali-sation	aesthetic
physiological						
safety	X					
love and belongingness		X				
esteem	X	X			X	X
self-actualisation						
aesthetic	X					

Table 5: Interdependencies within Maslow's Hierarchy of Human Needs

The above table tries to capture obvious interdependencies between Maslow's needs. For instance, a feeling of love and belongingness, high up in Maslow's hierarchy, can be found without having any of the other needs satisfied. The same could be said for esteem. The satisfaction of these needs is therefore independent of the satisfaction of other needs, which contradicts the idea of a hierarchy.

On the other hand, we can see that satisfying one need can have positive repercussions on the way other needs are perceived. As an example, the feeling of safety can be greatly enhanced if one is an esteemed member of a community, or has found a sense of belongingness with a partner. The idea of a hierarchical structure appears to be more valid in case of self-actualisation, which can be assumed to depend greatly on a sense of esteem. But in reversing the argumentation, it can be said, that if a feeling of esteem satisfies (at least partly) the need for self-actualisation. Therefore it cannot be regarded as an independent item in a hierarchy of needs.

The model below attempts to take into account the limitations of the idea of a hierarchy of needs, and instead proposes a more systemic view of the question of intrinsic needs. The second question we have to ask is whether it is possible to attribute specific activities to the satisfaction of needs.

4.3.2.1 Model Set-up

This spreadsheet model has been set up in a very simple way using deterministic system dynamics. It captures the interdependencies of a system of intrinsic needs and proposes a way to link need satisfaction with the choice of activities. A flow-chart of the model is shown in Figure 10.

The model aims to represent one "robot" individual in a static environment. Each time step in the model is representing one day. It is assumed that the satisfaction of needs can be measured using a set of budgets, one for every need. These budgets are filled by the payoffs of the activities chosen. There applies a constant decay rate to each budget. The results of the activities carried out during this day form the basis for next day's evaluation of budgets which in return leads to the allocation of time to activities. As the time is restricted to 24 hours per day, the evaluation of budgets leads to "importance points" for each activity. These are then transformed into "relative importances" which are equal to the fraction of the daytime allocated for the activity.

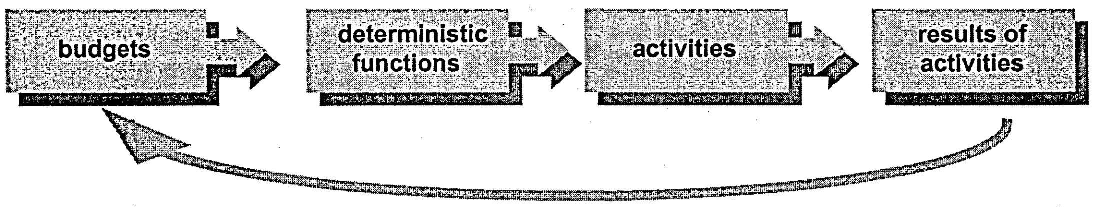

Figure 10: Flow-chart of the Spreadsheet Model

In order to account for the interdependencies between the needs as shown above, the hierarchy of needs was reformulated in a systemic way as shown in Figure 11. The transformation of the hierarchy into an interdependent system of needs proved to be a key issue for the formulation of the conceptual framework in the next chapter. The qualitative arrows in Figure 11 indicate general influences of budget states on each other. For instance, an improvement in the physiological budget will have a negative effect on the perception of the satisfaction of the needs for safety, love and belongingness, self-actualisation and the aesthetic needs. This interaction attempts to incorporate that once the material necessities of existence are satisfied, people will focus on other, non-essential aspects of their lives. On the other hand the feeling of love and belongingness will positively influence the perception of the satisfaction of the needs for esteem, self-actualisation and safety. These influences are incorporated into the model by deduction or addition of a percentage of the last payoff for an activity to the other budgets. It has to be noted that this procedure attempts to account for the perception of a single individual, which would be different for each individual in an application with more than one "robot".

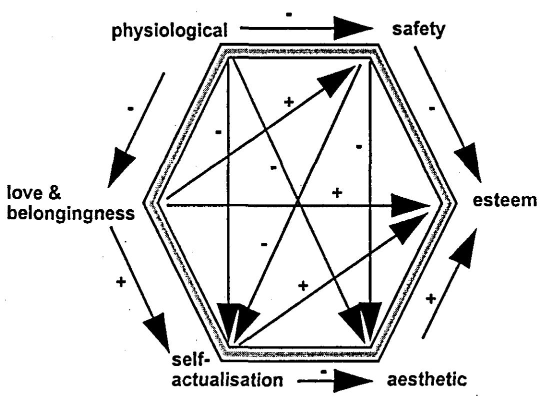

Figure 11: Driving System of Needs in the Spreadsheet Model

A set of given activities (see Table 6) provides means to satisfy the needs. The model is operationalised by connecting the duration of activities with the gross payoffs of the activities. The web of interdependencies then modifies the gross payoffs into net payoffs, which are in return credited to the respective budgets.

need	directly satisfied by	activity	credited to budget
physiological	services and goods, recreation	acquiring services and goods, recreation	physiological
safety	-	-	safety
love and belongingness	interpersonal contacts	socialising	love and belongingness
esteem	goods, money	acquiring services and goods, work	esteem, money (auxiliary budget)
self-actualisation	education, recreation	education, recreation	self-actualisation
aesthetic	education	education	aesthetic

Table 6: Relations between Budgets and Activities

There are some points to be mentioned about Table 6. First of all, some activities serve to satisfy more than one budget. In this case the result of the activity is split and then credited to the respective budgets. The other point is that there is no activity allocated to directly satisfy the safety need. This budget can only be influenced indirectly through the interdependencies within the system of needs. The cross-influences between activity results as shown in Figure 11 apply of course to the results of all other activities as well.

There are only two constraints applied to the model. One is that it is impossible to acquire services and goods if there is no money (the only result of work) available, another that there is a minimum time required to be spent on recreation, reserving time required for sleeping and eating. There is also a decay rate applying to the physiological budget. All activities have to add up to 24 hours per iteration (day). The complete set of equations is defined as follows:

Gross results of activities of previous time period are credited to the budgets:

$$\begin{aligned}
grossres(phys,t-1) &= specpayoff(serv)\cdot time(serv,t-1) + 0.5\cdot specpayoff(rec)\cdot time(rec,t-1)\\
grossres(safe,t-1) &= 0\\
grossres(esteem,t-1) &= money(t-1) + 0.5\cdot specpayoff(serv)\cdot time(serv,t-1)\\
grossres(aest,t-1) &= 0.5\cdot specpayoff(edu)\cdot time(edu,t-1)\\
grossres(s/a,t-1) &= 0.5\cdot specpayoff(edu)\cdot time(edu,t-1) + 0.5\cdot specpayoff(rec)\cdot time(rec,t-1)\\
grossres(lb,t-1) &= specpayoff(soc)\cdot time(soc,t-1)
\end{aligned}$$

with

$$\begin{aligned}
grossres &:= gross\ result\ of\ previous\ activities\ for\ budget\\
phys &:= satisfaction\ of\ physiological\ need\\
safe &:= satisfaction\ of\ safety\ need\\
esteem &:= satisfaction\ of\ esteem\ need\\
aest &:= satisfaction\ of\ aesthetic\ need\\
s/a &:= satisfaction\ of\ self-actualisation\ need\\
lb &:= satisfaction\ of\ need\ for\ love\ and\ belongingness\\
specpayoff &:= specific\ payoff\ of\ actvity\\
serv &:= acquiring\ services\ and\ goods\\
rec &:= recreation\\
edu &:= education\\
soc &:= socialiasing\\
time &:= time\ spent\ on\ activity\\
money &:= state\ money\ budget
\end{aligned}$$

Equation 10: Calculation of Gross Activity Payoffs

Net results for budgets (after interdependencies of needs):

$$\begin{aligned}
netres(phys,t-1) &= grossres(phys,t-1)\\
netres(safe,t-1) &= grossres(safe,t-1) - fact(phys)\cdot grossres(phys,t-1) + fact(lb)\cdot grossres(lb,t-1)\\
netres(esteem,t-1) &= grossres(esteem,t-1) + fact(lb)\cdot grossres(lb,t-1) + fact(aest)\cdot grossres(aest,t-1)\\
&\quad + fact(s/a)\cdot grossres(s/a,t-1) - fact(safe)\cdot grossres(safe,t-1)\\
netres(aest,t-1) &= grossres(aest,t-1) - fact(phys)\cdot grossres(phys,t-1)\\
&\quad - fact(s/a)\cdot grossres(s/a,t-1) - fact(safe)\cdot grossres(safe,t-1)\\
netres(s/a,t-1) &= grossres(s/a,t-1) - fact(phys)\cdot grossres(phys,t-1) + fact(lb)\cdot grossres(lb,t-1)\\
&\quad - fact(safe)\cdot grossres(safe,t-1)\\
netres(lb,t-1) &= grossres(lb,t-1) - fact(phys)\cdot grossres(phys,t-1)
\end{aligned}$$

with

$$\begin{aligned}
netres &:= net\ result\ of\ previous\ activities\ for\ budgets\\
fact &:= budget-specific\ factor
\end{aligned}$$

Equation 11: Calculation of Net Activity Payoffs

The state of the budgets is calculated as follows:

$$\begin{aligned}
budget(phys,t) &= budget(phys,t-1) + netres(phys,t-1) - decay(phys) - 0.5 \cdot decay(s/a) \\
budget(safe,t) &= budget(safe,t-1) + netres(safe,t-1) \\
budget(esteem,t) &= budget(esteem,t-1) + netres(phys,t-1) \\
budget(aest,t) &= budget(aest,t-1) + netres(aest,t-1) \\
budget(s/a,t) &= budget(s/a,t-1) + netres(s/a,t-1) - 0.5 \cdot decay(s/a) \\
budget(lb,t) &= budget(lb,t-1) + netres(lb,t-1)
\end{aligned}$$

with

$$budget := budget\ state$$
$$decay := decay\ rate\ of\ budget$$

Equation 12: Calculation of Budget States

The state of the budgets determines an importance of the respective activities:

$$\begin{aligned}
import(serv,t) &= 1.0001^{-(budget(phys,t))} \cdot trigger(serv,t) \\
import(work,t) &= 1.0001^{-(0.5 \cdot (budget(phys,t)+budget(esteem,t)))} \cdot trigger(work,t) \\
import(rec,t) &= 1.0001^{-(budget(s/a,t)} \cdot trigger(rec,t) \\
import(edu,t) &= 1.0001^{-(budget(s/a,t)} \cdot trigger(edu,t) \\
import(soc,t) &= 1.0001^{-(budget(lb,t)} \cdot trigger(soc,t)
\end{aligned}$$

whereby

$$\begin{aligned}
trigger(serv,t) &= 900 / \sum_{i=t-9}^{t-1} grossres(phys,i) \\
trigger(work,t) &= 900 / \sum_{i=t-9}^{t-1} grossres(phys,i) \\
trigger(rec,t) &= 900 / \sum_{i=t-9}^{t-1} grossres(s/a,i) \\
trigger(edu,t) &= 900 / \sum_{i=t-9}^{t-1} grossres(s/a,i) \\
trigger(soc,t) &= 900 / \sum_{i=t-9}^{t-1} grossres(lb,i)
\end{aligned}$$

Equation 13: Calculation of Activity Importances

The time for the activities is then allocated as follows:

$$
\begin{aligned}
time(serv,t) &= \frac{import(serv,t)}{\sum_j import(j,t)} \cdot 24; \quad if\, money(t-1) \geq 0 \\
&= 0; \quad if\, money(t-1) \geq 0 \\
time(work,t) &= \frac{import(work,t)}{\sum_j import(j,t)} \cdot 24 \\
time(rec,t) &= \frac{import(rec,t)}{\sum_j import(j,t)} \cdot 24; \quad if\, money(t-1) \geq 0 \\
&= \frac{import(rec,t) + import(serv,t)}{\sum_j import(j,t)} \cdot 24; \quad if\, money(t-1) < 0 \\
time(soc,t) &= \frac{import(soc,t)}{\sum_j import(j,t)} \cdot 24 \\
&and \\
money(t) &= money(t-1) - time(serv,t) \cdot specpayoff(serv) + time(work,t) \cdot specpayoff(work)
\end{aligned}
$$

Equation 14: Time Allocation for Activities

4.3.2.2 Findings

It appears to be possible to reproduce typical activity patterns of a working person with this model. However, this model is limited in the extent to which it can reproduce "real" behaviour. For instance, it does only work if the specific payoffs for each activity are set in a rather narrow range, which is still realistic, but virtually just an average. Individuals exposed to more extreme situations fail to "survive" by producing very strange behaviour, although it might be easy for people to cope with this type of situation. Here it is important to note that the rule set for the model individuals is in every case different to one we would find in reality. The failure to cope with "easy" situations is to be blamed on the model and its admittedly mechanistic principles. But, as Forrester (1969, p. 113) noted, "the first step in modelling is to generate a model that creates the problem". This is done here in a first iteration towards reality by creating rather realistic activity patterns not by imposing constraints, but from using intrinsic motivation.

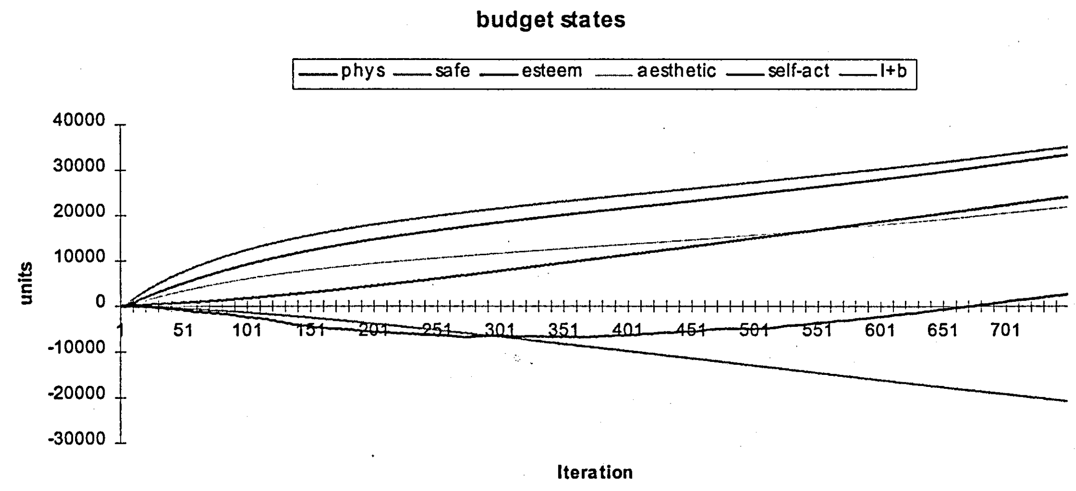

Figure 12: Example Budget Graph for the Spreadsheet Model

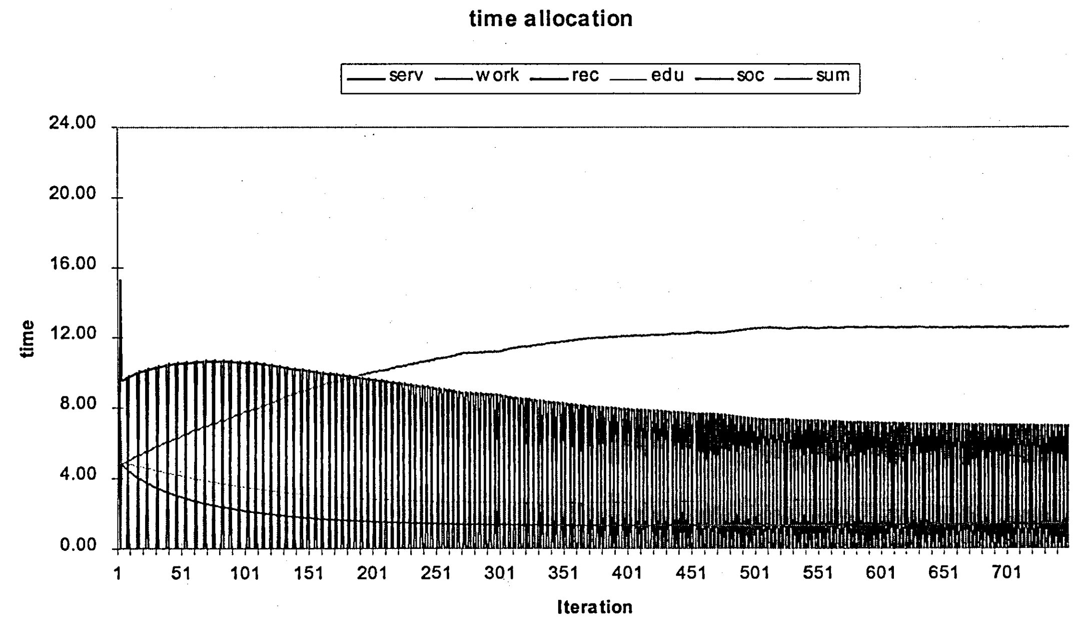

Figure 13: Corresponding Time Allocation for the same Configuration

In a large number of tested parameter configurations the time allocation for some activities proved to be periodically oscillating as shown in Figure 13. The apparent similarity to weekly patterns is probably of no significance, but due to the set-up of the system. However, the "behavioural patterns" are repeated on a period of six days! The other aspect of the results is that it was not possible to satisfy all budgets at any point with the very simple rules used. At least one budget (as shown in Figure 12) will be neglected in the set-up used.

Furthermore, the model is a non-spatial representation of how the satisfaction of needs could lead to patterns of daily activities as we know them. Location, spatial and temporal availability of activities through a differentiated land uses and travel time do not have any influence on the decision process. This model is also only simulating behaviour for one individual which does not interact with any other group or person, and it does not have any ability to adapt its behaviour to a changed environment. The individual acts only upon the immediate present status, and is not able to rely neither on past information (as it has no memory) nor can it anticipate future events.

4.3.2.3 Discussion

The findings from the spreadsheet model reinforce the idea that if there is a set of given intrinsic human needs it will be of systemic rather than purely hierarchical in nature. The principle of a feedback control system applying to the way the model individuals choose to satisfy their (predefined) needs appears to work in principle for a simple model.

However, it seems to be necessary to refine the basic assumptions in order to include more individuals as well as a spatial structure in such a model. Issues of competition between individuals - possibly leading to adaptability in their behaviour - cannot be dealt with in such a simple model.

The results are encouraging in the sense that the idea of needs as a driving force leads to results which resemble the time use of an average working person. It appears to be possible to approach the question of individual behaviour from a deductive point, therefore creating an explanatory model. This stands in sharp contrast to the traditional approach of extrapolating statistics used in conventional models of individual behaviour.

4.3.3 Max-Neef's Standpoint

Max-Neef (1991) has developed an approach to the problem of human needs in the context of (economic) development. This has a more systemic nature than Maslow's ideas. Max-Neef identifies nine fundamental needs (see Table 7) out of which only one, the need for subsistence or basic survival, can be regarded as having a higher value than the rest. This results in a situation, which is dominated by trade-offs and conflicts between the needs. In this sense, the results of the above reflections on Maslow's hierarchy of needs are very much reinforced.

The systemic nature of the needs becomes even clearer when the attributes, which satisfy the needs, appear for more than one need. "Solidarity", for instance appears in the category "Being" in Table 7 for not less than three needs: Protection, Affection and Participation. Read the table like this: If you are (category "Being") physically healthy, then your need for subsistence is likely to be (partly) satisfied. If you have (category "Having") food and shelter, then this is likely to help you as well in respect to subsistence.

Needs according to existential categories ⇒ / Needs according to axiological categories ⇓	Being	Having	Doing	Interacting
Subsistence	Physical health, mental health, equilibrium, sense of humour, adaptability	Food, shelter, work	Feed, procreate, rest, work	Living environment, social setting
Protection	Care, adaptability, autonomy, equilibrium, solidarity	Insurance system, savings, social security, health systems, rights, family, work	Cooperate, prevent, plan, take care of, cure, help	Living space, social environment, dwelling
Affection	Self-esteem, solidarity, respect, tolerance, generosity, receptiveness, passion, determination, sensuality, sense of humour	Friendships, family, partnerships, relationships with nature	Make love, caress, express emotions, share, take care of, cultivate, appreciate	Privacy, intimacy, home, space of togetherness
Understanding	Critical conscience, receptiveness, curiosity, astonishment, discipline, intuition, rationality	Literature, teachers, method, educational policies, communication policies	Investigate, study, experiment, educate, analyse, meditate	Settings of formative interaction, schools, universities, academies, groups, communities, family
Participation	Adaptability, receptiveness, solidarity, willingness, determination, dedication, respect, passion, sense of humour	Rights, responsibilities, duties, privileges, work	Become affiliated, cooperate, propose, share, dissent, obey, interact, agree on, express, opinions	Settings of participative interaction, parties, associations, churches, communities, hoods, family
Idleness	Curiosity, receptiveness, imagination, recklessness, sense of humour, tranquillity, sensuality	Games, spectacles, clubs, parties, peace of mind	Daydream, brood, dream, recall old times, give way to fantasies, remember, relax, have fun, play	Privacy, intimacy, spaces of closeness, free time surroundings, landscapes
Creation	Passion, determination, intuition, imagination, boldness, rationality, autonomy, inventiveness	Abilities, skills, method, work	Work, invent, build, design, compose, interpret	Productive and feedback settings, workshops, cultural groups, audiences, spaces for expression, temporal freedom
Identity	Sense of belonging, consistency, differentiation, self-esteem, assertiveness	Symbols, language, religion, habits customs, reference groups, sexuality, values, norms, historical memory, work	Commit oneself, integrate oneself, confront, decide on, get to know oneself, recognise oneself, actualise oneself, grow	Social rhythms, everyday settings, settings which one belongs to, maturation stages
Freedom	Autonomy, self-esteem, determination, passion, assertiveness, open-mindedness, boldness, rebelliousness, tolerance	Equal rights	Dissent, choose, be different from, run risks, develop awareness, commit oneself, disobey	Temporal/spatial plasticity

Table 7: Need system (Max-Neef, 1991, p. 32/3)

The needs he identifies are considered to be universal and invariant. In this sense, all humans regardless of culture or economic situation are seen to posses all nine needs. Not all needs have always been in existence, some have only recently (on the time scale of biological evolution) emerged, and Max-Neef claims that as the historic development continues, more needs are likely to come into existence. The invariance of the needs can therefore apply only on shorter time scales, which are more likely in the range of several hundreds of years than those time scales on which biological evolution takes place. Still, a time scale of hundreds of years is much larger than those on which even the most optimistic simulations, forecasts or development schemes are based.

The next element of this need system is called a "satisfier".

> *(Satisfiers) " ... are related ... to everything which, by virtue of representing forms of Being, Having, Doing and Interacting, contributes to the actualisation of human needs."* (Max-Neef, 1991, p.24)

According to the varying nature of these satisfiers they might affect some needs positively whilst at the same time having adverse effects on other needs. Max-Neef refers to these two classes as "synergic" and "inhibiting" satisfiers (see Table 8 and Table 9). This means that there is a constant trade-off between the satisfaction of all the needs, because Max-Neef argues that non-satisfaction of *any one* of the needs leads to pathologies in those individuals as well as in their society. Other satisfiers might lead to synergies by having a positive effect on more than one need.

Satisfier	**Need**	**Needs, the Satisfaction of Which It Stimulates**
Breast feeding	Subsistence	Protection, Affection, Identity
Self-managed production	Subsistence	Understanding, Participation, Creation, Identity, Freedom
Popular education	Understanding	Protection, Participation, Creation, Identity, Freedom
Democratic Community Organisations	Participation	Protection, Affection, Leisure, Creation, Identity, Freedom
Barefoot medicine	Protection	Subsistence, Understanding, Participation
Barefoot banking	Protection	Subsistence, Participation, Creation, Freedom
Democratic trade unions	Protection	Understanding, Participation, Identity
Direct democracy	Participation	Protection, Understanding, Identity, Freedom
Educational games	Leisure	Understanding, Creation
Self-managed house-building programs	Subsistence	Understanding, Participation
Preventive medicine	Protection	Understanding, Participation, Subsistence
Meditation	Understanding	Leisure, Creation, Identity
Cultural television	Leisure	Understanding

Table 8: Synergetic Satisfiers (Examples after Max-Neef, 1991, p. 36)

Satisfier	**Need**	**Needs, the Satisfaction of Which Is Inhibited**
Paternalism	Protection	Understanding, Participation, Freedom, Identity
Overprotective family	Protection	Affection, Understanding, Participation, Idleness, Identity, Freedom
Taylorist-type of production	Subsistence	Understanding, Participation, Creation, Identity, Freedom
Authoritarian classroom	Understanding	Participation, Creation, Identity, Freedom
Messianism (Millenialism)	Identity	Protection, Understanding, Participation, Freedom
Unlimited permissiveness	Freedom	Protection, Affection, Identity, Participation
Obsessive economic competitiveness	Freedom	Subsistence, Protection, Affection, Participation, Idleness
Commercial television	Leisure	Understanding, Creation, Identity

Table 9: Inhibiting Satisfiers (Examples after Max-Neef, 1991, p. 35)

Economic goods have a special place in this argumentation. Max-Neef criticises the common belief that economic goods are a human need.

> *"Satisfiers are not the available economic goods"* (Max-Neef, 1991, p. 24)

and further

> *"While a satisfier is in an* ultimate sense *the way in which a need is expressed, goods are in a* strict sense *the means by which individuals will empower the satisfiers to meet their needs. When, however, the form of production and consumption of goods makes goods an end in themselves, then the alleged satisfaction of a need impairs its capacity to create potential. This in turn, leads to an alienated society engaged in a senseless productivity race. Life, then, is placed at the service of artefacts, rather than artefacts in the service of life. The question of the quality of life is overshadowed by our obsession to increase productivity."* (Max-Neef, 1991, p.25, his emphasis)

Although the needs defined by Max-Neef overlap considerably with the ones of Maslow, he is much more specific on how needs can be satisfied. Considering the above remarks on economic goods, the typology of satisfiers suggests that needs are much more likely to be met through the social setting than through material things. This makes a "measurement" of need satisfaction with the help of budgets not as straightforward as in the previous spreadsheet model, although there is a coupling of need satisfaction with activities through the "Doing" category of Table 7.

Furthermore, the specific formulation of needs in search of a sensible theory of economic development has to be contrasted with our problem of a model of everyday life for an urban environment. Still, this very systemic approach to the problem of human needs can provide us with a framework on which a simulation can be built. Needs are invariant over considerable time, and they are universal, which is the cornerstone for a computational model relying on a closed universe (see 2.5). The needs can be satisfied though activities, but the systemic nature of the theory leaves us with a multicriteria problem, as the needs co-exist with each other. This will require an individual to make trade-offs between needs, because it will not always be possible to satisfy all needs to the same extent.

4.4 Implications of Evolutionary Theory

The search for adequate philosophical concepts on which dynamic models can be based has recently focused on evolutionary theory. Evolutionary theory deals with the long-term development of biological entities, and it appears reasonable to borrow concepts and analogies from a very generic theory of how these extremely complex systems develop over time. The application of evolutionary theory to processes, which are not primarily biological (in the sense of biological evolution), is from a strictly scientific point of view debatable (for a discussion on how far evolutionary analogies might be taken see Jeffrey, 1996). However, as a philosophical concept, evolutionary theory provides a good basis on which a model of cultural change might work, as long as the concept is not applied in the same way as the "social physics" approaches examined in Chapter 2.3.

Recent modelling has drawn on evolutionary analogies for the analysis of complex systems. On the other hand, the latest developments in biological evolutionary theory have been applying complex systems theory to the realm of biology. Still, the real mechanisms behind biological evolution are still to a large extent unaccounted for. Evidence suggests that the processes of evolution might actually be a combination of processes traditionally covered by different theories instead of one grand unified theory. Yet the theories about evolution can serve as new paradigm in the analysis of complex systems, like - in this case - human culture.

Evolutionary theory has developed four basic approaches to biological evolution to date. First of all, the most common theory of variation and selection based on Darwin, the even older theory of incorporation of useful features into the genome introduced by Lamarck, the assumption of an undirected "drift" of random mutations, and finally complex systems theory. In this section, all these theories will be assessed in terms of usefulness for a model of individual behaviour. This means that they cannot be discussed in all their facets as this would divert too far from the subject of this thesis. The emphasis is therefore kept on the principle mechanisms underlying the theories and their explanatory capacity for long-term system change.

4.4.1 Natural Selection and Gradual Change

The classic theory of evolution as proposed by Darwin (1872) relies on the selection of attributes, which make species more fit to survive in the competition for resources between them. Changes are regarded to occur at random in the genetic make-up of a given animal or plant. Thereby changes can occur in any direction, thereby either increasing or decreasing the viability of the organism. If this change happens to increase the "fitness" of the species, it will be propagated, because it gives the modified individuals an advantage over the original ones. If certain examples of one species have an advantage over others, this will ultimately lead to lower mortality and more offspring, thereby changing the population ratio between the new and the old species.

Darwin regarded the process of change as gradual and quasi-continuous. This assumption was reinforced by the addition of Mendel's genetic theory to Darwin's original theory. The idea that genes were responsible for specific traits of an organism, and therefore random change to certain genes would lead to random, but gradual, changes in the "fitness" of an individual. As this "genetic beanbag" (Wesson, 1991, p. 9) covers all properties of an organism with a great number of traits, the gradualist aspect of random change and selection seemed to be verified.

This basic essence of Darwinist theory involved a connotation of optimisation, as it claims that the species in existence must obviously have the optimum fitness, which is currently available by means of evolution. The whole process of selection was assumed to have been optimising species over the last couple of billion years. One of the main points of critique to this view is that if evolution were optimising then how do we account for all those species which are obviously maladapted, but which are in fact extremely successful in terms of number of population numbers, just taking the human race as one example. For their survival in a natural environment, humans feature a number of physical handicaps, such as rotting teeth, missing body hair for insulation from the cold etc. (For a more in depth discussion of this issue see Wesson, 1991, pp. 95 ff.) Still, the human race is one of the most successful species ever to exist.

Moreover, the theory does not say anything about how changes come into existence, which is an aspect not covered by other theories as well. There is as well no influence of experience or learning on the makeup of a species (as Lamarck suggested see 4.4.2). One more problem with Darwin's theory is that it cannot explain why there are huge gaps in the fossil record, because - even taking into account that fossilisation is a very rare event - a gradualist evolution would generate many more transitional species in between those which are actually recorded.

4.4.2 Lamarckian Theory

Darwinian theory limits itself to claiming that changes in the genome of a species are due to random mutation, which are subject to the selective pressure of competition. Lamarck, on the other hand, had claimed before Darwin that environmental influences change the individuals of a species as well. This theory has some particular merit in a cultural context (in the sense that learned features can be inherited) as well as that there is some evidence from biology that suggests that Lamarck's theory cannot be entirely dismissed.

It has been suggested that especially in mammals and birds there is a high capacity for the incorporation of learned or imitated behaviour into hereditary traits (Wesson, 1991, p. 225). Here it is thought that a high brain capacity increases the probability that such genetic change occurs. In any case the change caused through positive adaptation has to be supplemented by a complimentary selection process as well, so that this theory would have to be used in conjunction with Darwin's theory of mutation and selection.

Lamarck's theory has particular appeal to the loss of unused organs in species. This phenomenon contradicts Darwinism, because the loss of an (even unused) organ reduces the fitness of that particular species, as it then has fewer capabilities than before. The species is thus more vulnerable to influences, which could have been dealt with the formerly unused organ.

However, the main point of critique of Lamarck's theory remains that it is very much unclear how exactly an individual's behaviour as a response to environmental influences can change the hereditary features of an individual of a species.

Recently, a number of biologists proposed theories that regard culture as an integral part of evolution (New Scientist, 1997). Here culture and behaviour are seen as co-evolving and mutually depending on each other. The idea is very close to the essence of Lamarck's theory, but the great difference is that behaviour and physical make-up of a species (or part of a species) are separate as opposed to the original idea that behaviour influences directly the physical make-up. These theories are much more related to the theory of "natural drift", which is reviewed in Chapter 4.4.4.

4.4.3 Self-Organisation as a Driver for Evolution

One main limitation of the above theories of evolution is that they cannot explain the gaps in the fossil record, which is the main evidence we have about evolution. Even given the fact that fossilisation is a very rare event, the gaps cannot be attributed to solely lack of fossilisation of the right individuals of some species. Furthermore, it appears that only certain features of living creatures are to be found in combination (Wesson, 1991). These features do not change over very long time spans. Species do not appear to change as much as one would expect from either of the above theories, except for the incorporation of behavioural features into what appears to be the genetic set-up of species.

All this leads to the conclusion that evolution might in fact obey at least some of the principles established by complex systems theory. The preconditions for self-organisation are met: Biological systems are open systems, which take in energy (sunlight or food), from outside to export entropy, therefore they are dissipative structures in the thermodynamical sense. These structures have the property to self-organise themselves into patterns of temporal stability (living being). In order to self-organisation to occur, these structures have to import energy from outside. Once these patterns fail to sustain themselves, because fluctuations of the system as a whole have disturbed their trajectory enough, they may organise into some other, but probably most different pattern, without (or only with a very short) transition period.

This matches exactly the kind of patterns, which are delivered by the fossil record, which shows that dominating species (like the dinosaurs) became extinct very rapidly, and were replaced by others (mammals and birds), which have a different set-up. Leaving apart the issue of mass extinction by external factors, the transition from large reptiles to mammals and birds is not documented in the fossil record. According to Darwinian theory this should have been the case, because there should have existed many transitional species between the earlier dinosaurs and later mammals and birds. Self-organising evolution on the other hand can account for this gap, as animals might either exist as reptiles or as mammals and birds without allowing for transitional species. A cross between these phyla (in a transitional species) may not be technically possible.

4.4.4 Evolutionary Drift

The term "drift" has been applied to two very much different approaches to evolution. There is firstly the concept of neutral drift, which basically removes the premise of optimisation from Neo-Darwinist theory, thereby assuming that random mutation is undirected and does not necessarily improve the "fitness" of a species (Kimura, 1985). This view removes adaptation as a key concept from the theory, so that new species are supposed not to be necessarily better adapted to their environment than old ones.

The other concept is called "natural drift" and has been coined by Varela, Thompson and Rosch (1991). They summarise the theory as follows:

> *"1. The unit of evolution (at any level) is a network capable of a rich repertoire of self-organising configurations.*
>
> *2. Under structural coupling with a medium, these configurations generate selection, an ongoing process of satisficing that triggers (but does not specify) change in the form of viable trajectories.*
>
> *3. The specific (nonunique) trajectory or mode of change of the unit of selection is the interwoven (nonoptimal) result of multiple levels of subnetworks of selected self-organised repertoires.*
>
> *4. The opposition between inner and outer causal factors is replaced by a coimplicative relation, since organism and medium mutually specify each other.*
>
> *We intend this set of articulated mechanisms to replace the adaptionist outline ... and to give content to our announced alternative view. The view of evolution depends on the conjoint applicability of three conditions:*
>
> *1a. The richness of the self-organising capacities in biological networks*
>
> *2a. A mode of structural coupling permitting the satisficing of viable trajectories*
>
> *3a. The modularity of subnetworks of independent processes that interact with each other by tinkering "* (Varela, Thompson and Rosch, 1991, pp. 196/7)

This theory includes some elements, which have already been presented in different contexts before. The first is the term "satisficing" which we have encountered in the context of Simon's theory of organisational behaviour. It is used here to clarify that there is no drive to optimality in an evolutionary system. On the other hand it is possible to reach optimality in an evolutionary system, if the interdependent selective pressures are strong enough. This is a rather improbable case, because all elements of the system have to progress towards optimality at the same pace, given the diverse nature of a natural system.

The concept of self-organisation is taken a step further than before. Had it only been applied to evolving structures in a given (static) environment before, it is now used to determine the relationships between the organism and the environment as well. The interactions between organism and environment change over time due to the effects of past interaction and adaptation, which influences properties and behaviour of both. Organism and environment are mutually dependent on each other and they specify by their behaviour and requirements how the relations between them are defined. The main difference to the concept of self-organising evolution (4.4.3) has to be seen in the fact that the former applies the mechanism of evolution to the organism only, whilst this theory explicitly includes both organism and environment into the dynamics of the system. This means that a given ecosystem will only evolve in the very specific conditions encountered in space and time, whilst the same organisms would probably take a very different path in a different environment.

4.4.5 Summary

In conclusion it appears to be difficult to come up with a conclusive theory of evolution. The working principles of evolution needed for this thesis are very much unaccounted for, as there is still no commonly accepted comprehensive theory of evolution in existence. However, all existing theories on evolution make a useful contribution, because they all attempt to explain at least some observed phenomena.

Darwinian theory covers the important aspects of mutation and selection, but it cannot account for other phenomena such as hereditary positive adaptation, for which Lamarck's theory provides an explanation, although the precise mechanisms remain very much obscure. Here the Darwinian theory has the great advantage on relying almost exclusively on population ecology, which is a simple and understandable principle. On the other hand there are still many gaps in the Darwinian framework, like the incompleteness of the fossil record, for which a self-organising genome would provide an explanation.

The most advanced theory of evolution to date is Varela, Thompson and Rosch's approach. Their attempt to integrate the existing theories as much as possible in a non-optimising framework should be regarded as the state-of-the-art for the moment. Although the theory is unspecific in terms of how precisely evolution might work, it still can account for more observed phenomena than any other theory. The introduction of a systems view expressing the mutual interdependencies of all elements of an ecosystem leads to a holistic theory focusing on the generation of viable conditions from within an ecosystem. Viability is regarded as an emergent feature of the ecosystem leading the theory away from distinction of organism and environment. As viability is generated by the ecosystem the theory also removes much of the reference to absolute and universal viability conditions stressed particularly in the Neo-Darwinist theory. "Natural drift" relies on only the loosest definition of reference points (such as a given and mostly static environment) and attributes the local conditions and local reference points (such as the observed interactions in a food chain at a given point in time) to the self-organisation within an ecosystem.

Biological evolution is reviewed here only for the purpose of gaining insights into the working principles of an adaptive system. It is necessary to look further in order to build a framework for a model of everyday life. Although the theory of natural drift is very compelling, we will encounter a reformulation of this theory in the context of human cognition. Because the cognitive context is much nearer to the research issue in this thesis than biological evolution, the cognitive formulation of this theory will be used.

However, when evolutionary principles are mentioned in a more generic context of modelling, the theory of "natural drift" is referred to.

4.5 Applications of Evolutionary Theory

Although some methods in computer science and engineering borrowing from evolutionary theory have been in existence since the early 1970s (for instance Rechenberg, 1973), they have only very recently been applied to computer models. Probabilistic and deterministic methods like system dynamics were preferred for the description of the behaviour of systems. These have recently been superseded by self-organising models, which introduced non-linear non-equilibrium dynamics. Evolutionary models, however, require at least one further step away from the traditional mathematically deterministic methods on which computer science is built. This class of models calls for structural change of all system elements and the relations between these during run-time.

4.5.1 Evolutionary Models

Evolutionary models serve basically two purposes: On the hand side there are models which aim to explain mechanisms of evolution itself, and on the other hand there are models which use concepts of evolutionary theory as analogies to explain phenomena in domains other than biological evolution. Our primary concern in the given context should be with the latter ones, but the former category can be used in clarifying some basic properties of evolutionary systems. This can facilitate the design of a framework for a model of everyday life.

An early application of evolutionary principles to a model of human acting is the fisheries model of Allen and McGlade (1987). The model was built to explain the dynamics of the fisheries off the coast of Nova Scotia, Canada. Because the traditional equilibrium approach to the viability of fishing operations had not been successful and relied on a static system, Allen and McGlade resorted to a complex systems approach. The introduction of fluctuations on the reproduction of the fish stocks proved to come near the observed dynamics of the system. Still, this was not an evolutionary model, but a more traditional, Lotka-Volterra (predator - prey) model. The reformulation of this basic model into a spatial one introduced not only the principal economic mechanisms of supply and demand, varying elasticity of demand and advances in technology, but also a distribution of different behaviours on the side of the fishermen. The extreme positions of fishing behaviour have been called "stochasts" who explore the area more or less at random, and "cartesians" who do not take any risks and fish only where they know that there is fish to be found.

Unsuccessful fishermen go "bankrupt" and are removed from the system, whilst successful ones make enough money to buy more boats and expand their strategy. This is the application of the Darwinist idea of population ecology insofar as fishermen with superior "fitness" will finally dominate the system. However, it is not as simple as this. The basic configuration of this extension of the original model was modified to allow for change in the initial behaviour of the fishermen. Was it the "stochasts" who dominated the initial system, because they had the possibility to explore new, possibly more profitable strategies, the "cartesians" took over from the "stochasts" once they were allowed to copy information on good catches from the "stochasts". On the other hand the total removal of either "stochasts" or "cartesians" resulted in an overall lower performance of the system in terms of total catches, and the highest performance was recorded using a mix between "cartesian" and "stochast" strategy.

This behaviour suggests that any complex evolutionary system actually relies on the presence of both exploration and exploitation. Exploration is needed to find new viable modes of behaviour, whilst exploitation makes the best use of these newly found behaviours. A correspondence with the Gaussian distribution curve is obvious. In many natural systems the bulk of behaviours is found close to the average, but a small minority of behaviours is very far from the average exploring new aspects of the decision space.

An earlier model by Allen and Ebeling (1983) dealt on a more theoretical level with these kinds of phenomena. The model was set up to explore the mechanisms of evolution in a hypothetical ecosystem. Allen and Ebeling used a stochastic master equation description for the processes involved. The ecosystem was designed to resemble a predator-prey system.

The aim of the model was to show that a mutation - no matter whether increasing or decreasing fitness for a given species - could invade the system, and therefore influence the further evolution of the system. The focus was therefore shifted towards the stochastic behaviour of new mutants, which were generated by randomly changing the existing members of the species.

As opposed to the deterministic case in which a new mutant would have to have at least the same fitness as the existing individuals, it turned out that even a much lower fitness was enough to guarantee the survival of mutants for at least some generations. In the long term, however, it was found that the fitness of mutants would have to be at least 10% higher than that of the existing individuals to ensure that the new strain of characteristics is preserved in the long term.

> *"Quasi-neutral, and even quite negative mutations can survive for a long time in a locality, long enough anyway for new evolutionary paths to be explored, while frequently even quite advantageous mutations may be lost in the "noise" surrounding their birth."* (Allen and Ebeling, 1983, p.125*)*

The above conclusions make it clear that from the point of view of a stochastic analysis the ideas put forward by Varela et al. (1991; see 4.4.4 above) are very much reinforced. It showed as well that inhomogeneous spatial distributions are of the utmost importance in preserving and introducing new mutant species. In the case of a homogeneous distribution of characteristics and for very long time spans the solutions should resemble more and more the Neo-Darwinist ideas about evolution. This case is rather theoretical and reminds us of the assumptions made in early systems models. The analogy is obvious: Very important determinants of a system are wiped out by a much too aggregated description, and therefore simplistic conclusions arise from the use of the conceptually inaccurate model. It is not possible to claim that the above models are more accurate in a quantitative sense, but they have a much greater explanatory power than the aggregate description, because the system is described more accurately on the qualitative level.

Especially the use of random noise probably leads to results, which will never be quantitatively accurate, but the working mechanisms of the system are highlighted, helping to understand the system qualitatively. In the very complex setting of evolutionary mechanisms it appears to be more important to analyse a system by modelling it qualitative than to try to make quantitative predictions on the system's future behaviour.

4.5.2 Genetic Algorithms

The most widespread method based on evolutionary theory in computer science is the so-called genetic algorithm. A genetic algorithm is basically an optimisation method for very large search spaces, where conventional search mechanisms would probably not find the better solutions because of confinement to a narrow parameter and variable range. The concept builds on Darwin's theory of random mutation and selection.

A set of alternative solutions is defined and tested against whatever constitutes the "fitness" of the solution. The worst performing solution is then removed from the set and replaced with one, which is derived from other solutions by randomly changing some parameters or variables. Alternatively, combining parameters of existing solutions might also generate a new solution.

The first method refers to the accidental mutation of the genome in biology whilst the second one aims to mimic recombination of genetic material, both processes observed in nature. Then the process of testing and change is repeated until a satisfactory solution is generated. Usually, after some time (which can be quite long, because the genetic algorithm is basically a random search) the quality of the solutions approaches some optimal value asymptotically. In this case it remains unclear whether this really is the optimum solution, because the algorithm might not find even better configurations.

However, Genetic Algorithms have proved to be efficient tools for solving complex optimisation tasks, for instance in engineering. We have to note here that the Neo-Darwinist notion of optimality is used in this technique, which does not account for many phenomena in biological as well as cultural evolution. If the definition of evolution as mutually interdependent change between environment and organism (Maturana and Varela, 1984; Varela, Thompson and Rosch, 1991) is applied, the genetic algorithm is not even an evolutionary technique, because evolution in this definition co-evolves system elements and does not optimise them. The genetic algorithm tests the system elements against a static environment (the "fitness" requirement) which has no influence on how the solution is generated.

4.5.3 Classifier Systems

Holland et al. (1986) put forward a concept for learning systems in Artificial Intelligence based on induction. This allows for any rule-based system to generate new rules or to modify old ones, which have been found inadequate for the task they have been designed for. The approach is using experience as the main force in rule modification, and thereby differs clearly from deductive methods (used in conventional Expert Systems) which formulate a theory, and then try to verify it empirically. The inductive method does not presuppose any theory, but forms the theory (the rules) from experiencing the environment.

> *"The inductive mechanisms must accomplish three difficult, interrelated tasks. They must (1) evaluate the system's rules as instruments for goal attainment, improving them where possible and favouring the better ones in application; (2) generate plausibly useful new rule sets that are capable of*

> *extracting and exploiting regularities in experience and (3) provide associations between and clusterings among rules in order to create larger knowledge structures leading to efficient modelling of the environment. In carrying out these tasks, inductive mechanisms rely upon feedback concerning predictions about the environment."* (Holland et al., 1986, p. 68)

Holland et al. distinguish between two different kinds of rule sets in their systems: synchronic rules for state classification and diachronic rules for state prediction based on the classifications of the former rules. The inductive mechanism has to compare a prediction made with the existing rules to the actual experience or payoff of that prediction. If these rules fail, rule revision is triggered. The goal-directedness of the inductive process is stressed throughout:

> *"Induction is not simply something the cognitive system does to occupy its idle moments, nor does it have the character of undirected inference making or random combination of ideas. Rather, induction is a problem-directed activity, executed in response to specific system objectives such as seeking plausible explanations for an unexpected outcome."* (Holland et al., 1986, p. 68)

This means that the rule system used is designed to perform a given task from the outset. In our application it is not clear what tasks are to be performed, and the purpose is to model individual behaviour in the context of everyday life. What is not clear is the nature of how changes in individual behaviour come along, it can just be suggested that there is a connection between experience of the individual and changes in the environment. Therefore it is not known whether induction is the "correct" way of learning in individuals. However, as the process of induction is modelled on real-life learning from experience, it remains a very interesting approach to our problem, although a less specific way of creating learning individuals is thought to be more appropriate for a first implementation of such a system.

The so-called "classifier system" was built on this induction-based framework. Classifier systems are basically a rule modification method "(operating) with highly general learning mechanisms applied to a simple representational scheme" (Holland et al., 1986, p. 102). The reason for the name "classifier system" becomes clearer when looking at the properties of these systems:

> *"1)* Parallelism. *Large numbers of rules, called* classifiers, *can be active simultaneously. Problems of "scheduling" are avoided because the only action of an active classifier is to post a message to a message list - more active classifiers simply mean more messages. (The rules are called classifiers because they can be used to classify messages into general sets, but they are much more powerful than this name would indicate, providing both processing and recording.) The system uses the rules as building blocks, activating many rules concurrently to summarise and act upon a situation.*
>
> *2)* Message passing. *Classifiers, as is usual with production systems, consist of a* condition *part and an* action *part, but the conditions are all defined in terms of the set or class of messages that satisfy them. That is, a classifier*

system becomes active *when each of its conditions (there may be more than one in the condition part) is satisfied by the presence of an appropriate kind of message on the message list. The action part of the classifier system then specifies a message to be posted to a message list on the next time-step. A classifier system can be viewed as a kind of office, with various individuals (the classifiers) processing selected memos (the messages) from a pool (the message list). To keep the definition of individual classifiers simple (which facilitates the generation of new classifiers, as we will see), all messages and conditions are standardised as binary strings of fixed length. All communication from and to the outside (output and input) is via messages, so that any given classifier system can be connected easily to an environment or to other classifier systems. Interactions with an environment are handled via input interfaces (often a set of property detectors) that generate messages, and output interfaces (message-controlled effectors) that react to messages.*

3) Lack of interpreters. *Because the interaction of classifiers is solely via messages and does not depend upon the ordering of the classifiers in some store, and because the satisfaction of conditions is determined by a simple matching operation, there is no need for high level interpretation as part of the computational mechanism. Messages incorporating tags, along with conditions requiring the presence of those tags, can be used to couple classifiers force predetermined execution sequences, and so on. In consequence ... classifier systems are highly modular and graceful: it is possible, usually, to add new candidate classifiers to a classifier system without causing global disruptions, and there are local syntactic operators that generate such candidates."* (Holland et al., 1986, p. 103/4)

The parallel working of rules is a very interesting feature of classifier systems and it bears a strong similarity to Fuzzy Logic, which will be reviewed in Section 4.7.2. The rules are set using three-valued logic: 0 (no), 1 (yes) and # (ignore). The definition space for the rules covers all possible input parameters, so that any classifier can in theory refer to any combination of input parameters. The execution sequence of a simple classifier system reads as follows:

"1) Place all messages from the input interface on the current message list.

2) Compare all messages on the current message list to all conditions of all classifiers and record all matches.

3) For each set of matches satisfying the condition part of some classifier, post the message specified by its action part to a new message list.

4) Replace the current message list with the new message list.

5) Process the message list through the output interface to produce the system's current output.

6) Return to step 1." (Holland et al., 1986, p. 105)

The way the input messages are compiled in order to activate a classifier means that all features which are detected by any rule are considered in the next step. The rules can therefore specialise in the detection of certain features, and combination of features can be considered as well without defining a new rule as it would be necessary in a conventional rule based system. This makes the classifier system extremely flexible and resilient against fluctuations in the system's input.

There are two alternative ways of changing a classifier system suggested by Holland et al.. On the one hand side there is the so-called "bucket-brigade algorithm", a method to assign weights to successful classifiers and their couplings. Classifiers are given weights, which initially are the same for all. Then a fraction of the classifier's weight (which can be randomly varied around an average in order to explore new combinations) is sent with the message to the next classifier in sequence, which adds the tag to its own weight after sending its own tag (again a fraction of its weight) to the next classifier. Finally, the payoff of the activated sequence is credited to the first classifier.

This algorithm is similar to Hebb's learning algorithm (Hebb, 1949) which is used in artificial Neural Networks (see 4.7.1), only that the weights are assigned to explicit rules as opposed to neural nodes. In contrast to a Neural Network, there is no need for beforehand training of a classifier system, and the modification process continues throughout the use of the system.

The other suggested way of modifying a classifier system is the use of a genetic algorithm (see 4.5.2). Here the genetic algorithm is used only after a number of time steps, so that the performance of classifiers can be monitored over a longer time span. By using the genetic algorithm the system is transformed into a dynamic rule based system, which can form one part of a truly evolutionary model, if the principle is applied to both the environment and the individual.

4.6 New Developments in Cognitive Science

The discovery of emergent properties in the neural systems which make up the human nervous system, and which are therefore the basis for our cognition, has lead to the paradigm of cognition as an emergent property of the body. The cognitive experience is triggered by environmental (sensory) influences on the nervous system. There are of course basic properties of the nervous system, but these are not immanent in cognition itself, as the latter is caused by interaction of the parts of the nervous system, and cannot be regarded as existing per se.

One example for a theory based on this paradigm is Minsky's (1986) idea that the mind is composed of a set of different "agents" each sensing or actuating different things. These agents are very closely connected with each other, so that certain configurations of states of the agents can give rise to perceptions or actions very much different to the actual designation of the agent itself. In this sense, the human mind cannot be regarded as a single entity, but rather as a *society of mind.* The concept of "agents" is similar to what we find in the so-called agent-based models (see 4.2), but is here used to denote functional entities as opposed to models of physical entities in the agent-based models.

This has important implications for a model of human acting. If the mind can be regarded as a "society" composed of different agents, each responsible for different actions or decisions on a large number of different time scales, it becomes clear that the often contradicting perceptions and aims in certain moments can give rise to a totally different, but in itself consistent type of behaviour (due to self-organisation), which might have a contraproductive effect on some of the goals of the individual.

The theory does also imply that when dealing with individuals in a model, there should exist some agents within one individual to deal with different perceptions and aims. These agents have to be highly interconnected to provide the possibility for self-organisation within these agents.

Varela, Thompson, and Rosch (1991) have carried the self-organising paradigm on step further. As in their theory of evolution (4.4.4) they regard cognition as a self-organising feature which relies on the mutual interaction between individual and environment. The cognitive system is regarded as being composed of agents analogue to Minsky's theory. Instead of regarding the agents as ready-made, static entities which self-organise due to their interaction, the parts of the cognitive system are also able to influence each other as well as being the subject of the interaction with the environment.

The term "enaction" is used to summarise the central concept of the theory. The individual and its nervous system cannot be regarded as separate from the environment, because the organism is very closely interacting with and changing the environment, which in return is changing cognition. The perception of the environment depends on how it is "enacted" by the individual. This means in reverse that the individual is actually creating its own world by perceiving it (Not only for the idea that perception determines action, but also for how the image of the world is created by the individual). As the individual and environment mutually specify each other, the never-ending process of the becoming of cognition for the specific individual becomes extremely important. The current situation and its perception are based on the historic pathway leading up to that specific point in space and time. Therefore *history* is determining an important part of the system. The second conclusion of this theory means in fact that there are no objective reference points in people's perception and actions. Action modifies perception, and perception mediates action.

A computational model of human acting thus has to have a rule base (in whatever form this might be put into practice) which allows for modifications from within the model. In this way the rules at every point *t* in time represent the accumulated knowledge of the individual, no matter how simple the rules are formulated. It does also imply that there has to be a rule base for the individual's environment to represent the changing processes around the individual.

4.7 Application of Cognitive Concepts in Computer Science

Computer science has built on the new theories in cognitive science creating new methods building on analogies to nature. The two concepts described and compared here are Artificial Neural Nets and Fuzzy Logic. Artificial Neural Nets are modelled on the way human brain cells are thought to be functioning. This methodology gave rise to the so-called "low-level" Artificial Intelligence (AI), because these computational structures are self-organising from their parts into working entities, which do not use a pre-defined explicit model of the problem, they are used on.

Fuzzy Logic on the other hand cannot be regarded as a cognitive concept per se. It is rather an extension of conventional logic, which has lead to a methodology, which in can be used to map cognition into computational structures.

4.7.1 Neural Networks

"Low level" AI consists above all of the method of using Artificial Neural Nets. These are a computer programme structure working similarly to actual neurons in mammalian neural systems. The working principle behind artificial Neural Nets is based on the paradigm of self-organisation, which leads to emerging properties in the computer programme.

To make a Neural Net work, it has to be trained with sample data from the field of application containing input and output values of a number of problems. The training procedure aims at minimising the error between the training data and the actual outputs of the Neural Net by varying weights between so-called nodes in the programme (Beale and Jackson, 1990). Once a Neural Net is trained it is usually capable of dealing with similar problems than the ones contained by the training data. It is also capable of abstracting from the training data to deal with problems never encountered before.

Neural Nets are in this way able to represent expert knowledge in a certain field (Nauck, Klawonn and Kruse, 1994), but the method used is totally different to knowledge representation by classical Expert Systems, because the knowledge is conveyed by the training data. The crucial point in using artificial Neural Nets is therefore the choice of the training data.

Another negative point about Neural Nets is that they are working as a black box, and do not have great explanatory power, because they are only manipulating abstract weights of connections in the programme structure. It is also difficult to readjust the behaviour of Neural Nets at runtime as the structure of the problem (like in our case) might change fundamentally during a simulation. Neural Nets are usually descriptive items, although they do not use an explicit model. They are thus model-free estimators (Kosko, 1991).

4.7.2 Fuzzy Systems

Fuzzy Systems are based on the theory of Fuzzy Sets developed by Zadeh (1965). Fuzzy Set theory is extending conventional bivalent logic by allowing elements to be only partly member of a given set. This can be used to describe incompletely defined problems and avoids problems of setting the right threshold in classification problems.

Fuzzy Systems are a combination of Fuzzy Sets connected by an inferencing mechanism. These have been very successfully applied to a wide range of problems, especially in the engineering sector.

Fuzzy Systems consist of a rule base very similar to the ones used in classic Expert Systems. The big difference between the two approaches is that conventional Expert Systems are using bivalent logic, while Fuzzy Systems rely upon multivalent logic. This means in practice that in a conventional Expert System only some rules apply at a time, while in a Fuzzy System all rules are always applied to the degree to which the input fits the set triggering a certain rule (Kosko, 1994). So most rules would be applied to a degree of zero.

The output of all rules in the system is then weighed by the degree to which the rules apply. An analogue to this can be found in a table of experts working out a consensus on an issue, each rule representing an expert.

Fuzzy Systems are very handy to operationalise problems which can be formulated vaguely in spoken language (Like: If the temperature is *rather* high, turn down the heating a *bit*.), but which are difficult to capture in a set of mathematical formulas. By using linguistic rules even highly non-linear or discontinuous systems can be described fairly accurately without excessively using mathematical formulations defined only in a very narrow range, and which might even be too exact for the problem in question.

Although Fuzzy Systems are a very easy-to-use method to flexibly operationalise complex models, a conventional Fuzzy System is not able to change its rules over time. Here we have to refer to another (again time-invariant) Fuzzy System or a different method like Genetic Algorithms described in the Chapter 4.5.2 to modify the initial rule base and inferencing mechanism.

For this reason the main application of Fuzzy Systems are control systems. This is important in our context, as the experience with the control system in the spreadsheet model (4.3.2) has shown that this set-up is at least to some extent applicable to the multicriteria problem of human needs (Section 4.3.3). Fuzzy Systems offer the opportunity to measure the fulfilment of needs in a way, which is very easy to analyse in the model. Other aspects of the idea of a system of needs can be captured as well, like potential self-reinforcement of some needs in a positive feedback loop. Fuzzy Logic enables us to set up a more elaborate control system in a very straightforward way which as well map the cognitive concepts discussed above into the model and still remains easy to interpret.

In a model a suitable rule set has to be found that connects needs (which have to be measured by some indicator) with possible activities of people. As people have very diverse definitions of perceptions arising from their individual experiences, these might be best captured by using Fuzzy Logic using the least common denominator to name parameters, but leaving the actual parameter range to the individuals and their perception.

4.8 Summary

The research reviewed in this chapter is to be used to build a framework for our model of individual behaviour in order to build an urban model taking into account the processes taking part on the micro level. But before we embark on this venture in the next chapter, a summary of the evidence will be presented to give an outline what can be expected from the conceptual framework.

The methodology of using autonomous agents (2.3.16 and 4.2) in computer models provides us with a way of implementation for a model of everyday life. Agents resemble individuals as well as other entities such as companies, government agencies, and even the natural environment can be viewed as a collection of agents. These agents interact with each other on a local scale, which leads to diversity and a heterogeneous system. These features can be regarded as one of the keys to modelling a complex system (see 2.3.15 and 4.5), and it facilitates building a model if the methodology matches the type of system to be modelled from the outset, thereby leading to a structurally less complex model.

However, agent based approaches do usually rely on rule sets defined in sharp logic. As these are conventional rule-based systems, they are very easy to implement in a computer model as most programming languages are built on this concept. On the other hand there are immanent deficiencies like the need to define thresholds for the rules to fire, which are by definition arbitrary and in most cases not applicable in reality. This is where Fuzzy Logic (4.7.2) proves to be an extremely useful tool. As we cannot know the exact "rule set" of any individual we have to make approximations at least in the beginning. Fuzzy Logic provides us with a method of defining qualitative rule sets, without depriving us of a straightforward interpretation like Neural Nets (4.7.1) do it with their "black box" set-up.

As a side effect, the usually non-linear relationships within the agents and with their environment cause self-organisation on the macro (societal) and micro (individual) level of the system. This phenomenon has already been exploited very successfully in urban models as we have seen in Section 2.3.15.

The exercise on human needs in Chapter 4.3 has shown that the assumption of needs can be used as the driving force of a model of individual behaviour. The conclusions from the spreadsheet model of Chapter 4.3.2 reinforce Max-Neef's theoretical approach to the problem of needs on the experimental level. The definition of needs leads to a multicriteria control system as the core of our agent's motivations. This links again very nicely into the potential use of Fuzzy Logic in the model to be built (see Chapter 4.7.2).

On the environment's side of the model, Time Geography (2.3.7) has formalised many of the constraints which apply to the individual's behaviour. Many of the possible modes of behaviour are determined by the time available, and time limits the spatial availability of activities. Therefore it appears sensible to take into account the Time-Budget approach as well.

We have learnt that human behaviour is not as straightforward as it was seen by the builders of earlier models. One key to a better model is the application of Simon's theory of "satisficing" behaviour (see Section 4.1). People are acting within a bounded rationality, which means that agents modelled on people can no be longer omniscient, and they will make mistakes in perceiving what is going on around them. As opposed to the traditional optimising mantra, a model can rely on much less performance in search processes. A "good enough" solution is all we have to look for.

The control system approach based on satisficing behaviour and a qualitative Fuzzy Logic rule base modelled on a system of needs can produce the momentary behaviours of a population of agents. As behaviour and with it the underlying rules change over time it is necessary to consider a theoretical framework for the dynamics of such change. Simon's work links neatly into the theory of "enaction" formulated by Varela, Thompson and Rosch (1991; 4.6). The absence of reference points in cognition (and therefore behaviour) can be modelled with Fuzzy Logic as well. The position of Fuzzy sets on a budget axis has then to be based on the experience of the individual to determine the individual's characterisation of states.

In the more generic context of change in the environment we can turn to evolutionary theory. Because the theory of "enaction" can basically be regarded as the application of the theory of "natural drift" (4.4.4) in the cognitive context, conflicts of theory are not expected. The evolutionary principle will as well provide some insights in how individuals and environment specify each other by mutual interaction.

The key points can be summarised as follows:

- Individuals are to be modelled using agents.
- The agents are to be equipped with a set of intrinsic needs. The needs have a systemic structure which leads to a multicriteria problem
- The perception of the agents is based on a Fuzzy Logic rule base, which is modified according to the experience of the agents.
- The agents and the environment specify each other through interaction. Agents and environment change according to the principle of "natural drift".

In the next chapter these points are extended into a conceptual framework which will provide us with the foundation for a computer model. The evidence presented above will be formulated in a more rigid fashion in order to build a model operating on exactly the principles presented here.

5 Conceptual Framework

The review of evidence has shown clearly that a dynamic model of individual behaviour in the context of daily life is feasible. In this view we can now set out to formalise the evidence in a conceptual framework. The framework has to incorporate the aspects of a model of individual behaviour as well as a methodology on how to implement adaptivity based on evolutionary mechanisms.

The conceptual framework will explain how the elements of the theory add up to a comprehensive method resulting in the modelling approach followed and described in Chapter 6. However, as this framework is intended to provide a generic approach to the simulation of adaptive behaviour, not all the features outlined below will be found in the model as well, which has to be regarded as a first step in the validation process of the conceptual framework.

5.1 A World of Agents – the Bottom-Up Approach

The conclusions from the evidence gathered suggest that an agent-based model appears to be the adequate method for a model of individual behaviour. Especially Varela, Thompson and Rosch's (1991) theory on cognition (4.6) and evolution (4.4.4) as a network of interacting entities suggests that the agent-based approach (see 2.3.16 and 4.2) is the most promising methodology for a model of a social system.

However, as it has been pointed out, not only the inhabitants, but also institutions and the natural environment can be regarded as (computational) agents. The extent to which the approach should be followed depends on the nature of the model and the aims pursued by it. In these areas it might be useful to reduce computational effort by relying on conventional systems dynamics for some parts of the model, which are not in the focus of the question to be investigated. As long as the distinction which approach is followed in which place is made clear at all times, the methodologies can be employed simultaneously. However, different results might be obtained depending on the methodology.

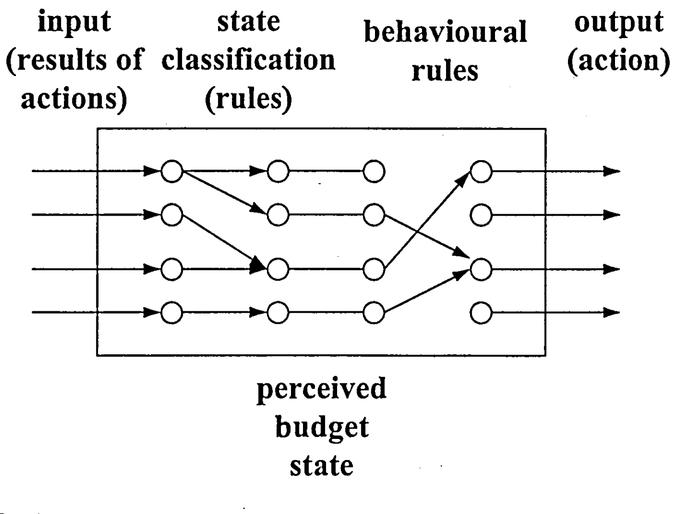

Figure 14: Example Agent

Figure 14 shows a proto-agent. The agent consists basically of a set of input parameters, which are then classified according to a set of rules. The state classification is the basis for a decision on outputs or actions determined by a set of behavioural rules (indicated by the arrows connecting parameter sets). It has to be noted that this agent is based on a cognitive approach, which relies on a subjectively perceived budget state for decision making. An agent representing a natural system would usually not feature this cognitive approach as this can be regarded as exclusive to social systems. The key features of the agent-based approach as used in the model described in Chapter 6 and its implications will be outlined below.

The probably most significant property of an agent based model is the fact that the system is described not by using any aggregate dynamics like the ones we would use in, for instance, a master equation approach. The agent-based approach builds the dynamics of the system by letting the agents interact with each other on a local level. This resembles the real world more closely than any traditional probabilistic dynamics. The probabilistic approach does not distinguish between the precise interactions, which have been taking place, so that the perturbations, which are introduced into the system, are applied with certainty not to other elements in a cause - effect relationship. The bottom-up approach on the other hand makes exactly this difference, so that deviations from average behaviour find their propagation in exactly the place where they have been caused. This might at first sight appear to be a mere philosophical distinction, but in fact the complex systems approach demands that "noise" that is generated by the system should be taken into account at exactly the place it is generated, because these fluctuations might produce major system changes.

In general, however, it has to be assessed whether the bottom-up approach is really needed for a model. For many investigations it is not important to reproduce lower level dynamics (like in the models of Allen et al. in Chapter 2.3.15), if the time scale of the simulation does not exceed the validity of a more aggregate view of the system in question. This might facilitate the implementation of the model. In addition to this, an aggregate model means usually a simpler model as well. The simpler the model is, the more accessible it will be to analysis and this means as well - if implemented on a computer - fewer resources are needed to run it.

5.2 The Evolutionary Principle

The agent-based approach outlined above will on its own only add up to a dynamic model using time-invariant local rules and diverse behaviour. This can lead to extremely interesting behaviour in the model, but as the rules underlying the behaviour of the agents do not change over time, the response to a given state will always be the same. Behaviour on the other hand has to be regarded as a dynamic feature, which changes over time according to experience (which might lead to a shift of preferences) or learning.

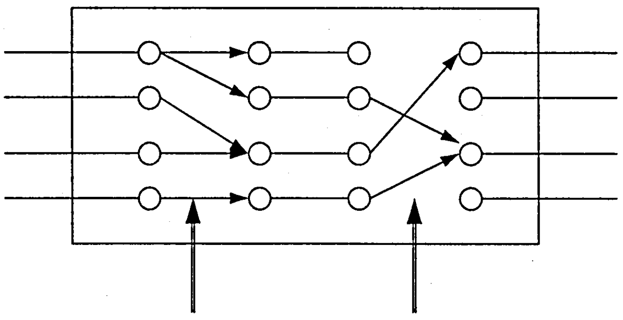

Figure 15: Agent with Rule Modification Mechanism

Therefore it is necessary to introduce a dynamic mechanism into the model, which changes the working principles of the model during run time. This leads us back to the concept of an evolutionary model (see 4.5.1). In Figure 15 such rule modification features in an agent as defined above are indicated by the double-lined arrows. Connecting new input and output parameters can now change the perceptual and behavioural rules. The number of rules (each rule is signified by a single-line arrow) is not necessarily constant, and some input parameters might lose their importance completely if no rules originate from this input.

5.2.1 Assumptions

In following the evolutionary approach, it is necessary to pay attention to the basic assumptions underlying any model. Allen (1997a) points out that models based on system dynamics use two assumptions:

- Events occur at their average rate
- All individuals of a given type are identical and of average type (Allen, 1997a)

Models using the principle of self-organisation have to discard at least the first assumption, because fluctuations are needed to push a system from one attractor to another. This is done by using the methodology of so-called master equations or simply (sufficient in most cases) by adding noise to a set of differential equations describing the system.

An evolutionary model has to go one step further and discard the second assumption as well, as the essence of evolution is exactly the change of individuals of a given type into something else. This might be best done by either including random or other variation in parameter values of individuals, or - going one step further in differentiation - to allow for a different set of properties for each individual. The approach inevitably leads to a "bottom-up" model, in which the macroscopic dynamics of the model are governed by the dynamics of the lower scale levels.

Still, this is not enough for a truly evolutionary model following Varela, Thompson and Rosch's theory. Most existing so-called evolutionary models have basically two different ways of changing the system interactions. Either they use a deterministic algorithm like, for instance, Hebb's (1949) learning rule, or they rely on random change of rules or parts of rules. However, all system elements are subjected to *exactly the same* mechanisms of rule change. The outcome of a technique using random influences might differ, but the technique itself applies to all elements. This amounts to a third assumption in the sense of Allen (1997a):

- Changes to the system elements are identical or at least of average type

This violates the theoretical framework of Varela, Thompson and Rosch (1991). In Section 4.4.4 it has been pointed out that

> *"The opposition between inner and outer causal factors is replaced by a coimplicative relation, since organism and medium mutually specify each other."* (Varela, Thompson and Rosch, 1991, p. 196)

This means that not only the system elements are diverse because of their past interactions with a medium (the environment), but the mutual specification of organism and medium is to be extended to the point where the mechanisms of change are concerned. In this argumentation the organism has to discover by itself whether, and if so, how to change (learn). This differentiated perspective on system change means that we have to regard change (or in our case, learning) as an emergent feature of an evolutionary system. An evolutionary model has therefore to allow for such processes.

One danger with this argumentation in a computer model relying on a closed universe according to Chapter 2.5 is that it leads ultimately to ever recursing levels of modifying mechanisms, because behind the behavioural rules and the learning rules (which we have covered by now) appear the rules to change the learning rules, and so forth. It is not known whether generic rules can be found at any level (maybe the level of physics), or whether the mechanism or changing the modification mechanisms becomes insignificant at some point. Still, these points have to be kept in mind in the process of implementation of an evolutionary model.

5.2.2 Driving Forces

Within the evolutionary principle of our conceptual framework, the aspect of driving forces for the agents has a very special place. It has been outlined that the assumption of driving forces for individual behaviour stands opposing the use of statistical measures of observed behaviours. While the extrapolation of statistical measures can be validly used to compile a set of current behaviours, it appears to be difficult to base a (in a modelling sense) meaningful evolution of behaviours on a collection of extrapolated deterministic functions.

It appears to have a much greater explanatory value for the model builder to test the validity of a set of assumed driving forces against the observed reality. The model can thus be used to learn about the modelled system as the central assumptions of the model are either falsified or confirmed. Therefore this conceptual framework relies explicitly on the definition of driving forces, which are here assumed to consist of a set of intrinsic needs as defined by Max-Neef (see Chapter 4.3.3). The systemic nature of this system of needs, which are interdependent, leads to a multicriteria control system. All needs will have to be satisfied in order to sustain the individual. The degree of satisfaction of these needs is regarded as a determining factor for the agent's behaviour.

The idea that evolving systems (like our urban system) are subject to a process of satisficing (see 4.4.4) fits very nicely with the theory of a system of human needs. A multicriteria system cannot be optimised in a straightforward way, any measure on how to optimise the system as a whole depends very much on how the requirements on the elements are defined. The classic approach to optimisation, the definition of an additive utility function bears in this context the danger of averaging out crucial shortcomings in some budgets with over-performance in others.

If there is optimisation taking place, the performance measure for what to be optimised has to be evolved, probably by the system itself. On the other hand, the minimum requirement to the system - here the satisfaction of all needs - can be easily established. Whilst the assumption of optimisation requires us to make even more assumptions on the nature of the system, the assumption of satisficing can already be supported by the theory presented in the previous chapter.

5.2.3 Self-organisation

As it has been shown in Chapter 2.3.15, self-organisation is one of the most important features of a non-linear dynamic model. Self-organisation can take place already in systems governed by deterministic equations, which makes this phenomenon not special to evolutionary systems, but evolutionary systems rely on self-organisation to take them through their trajectory in parameter space and time.

Self-organisation supplies an evolutionary system with temporal stability in the mutual interactions between the parts of the system. As opposed to a dynamical system described by one fixed set of equations leading to one self-organisation event, evolutionary systems also change the equations by which they can be described. This usually leads to a series of self-organisation events, which determine the temporarily stable configurations, and behaviours of the system. Ebeling (1989) in fact defined evolution as a series of self-organisation events.

The concept of self-organisation is to be taken one step further than before. Has it been applied in the past to structural configurations and modes of behaviour of systems, the concept can be extended to the definition of dynamic relations as well. This means that traditionally, there was one set of equations for a system, which could lead to different modes of behaviour by means of self-organisation. In an adaptive system this very set of equations is changing as well. Self-organisation will lead to stable combinations of variables in new equations or rules. This issue will be elaborated in the next section.

5.2.4 Emergent dynamics

The question of the central assumptions underlying our model leads inevitably to the question of how the dynamics of an evolutionary model are defined. It was mentioned in Section 5.2.1 that the removal of average change in the agent set-up is required for evolution to take place. This means that the dynamics of the model have to be regarded as emergent properties of that model. However, as outlined in Section 2.5, it appears to be very difficult to account for truly emerging features in a computer model. Here the closed universe of a computer program allows only for new combinations of already defined parameters into new rules, not for the introduction of previously unknown parameters. In this sense, the combination of base parameters into a new one is not an entirely new parameter, but an aggregation of two previously existing ones. Therefore nothing really new can emerge from the model except for maybe a new perception.

Still the chance of recombining parameters in new rule sets, adding to existing ones or replacing old ones, is the nearest we can get to emergent dynamics in the model. If now the definition space of the model is large enough and agents rely only on a fraction of the definition space, the agents can still make discoveries and change their environment.

The process of discovery consists mainly in the building of new connections between system parameters representing features of the natural world.

The creation of new networks of agents and the aggregation of agents leads to new higher order entities which might be endowed with new properties and rules sets. These rule sets can still only take into account the basic parameters which are defined for the model, but the dynamics of these new meta-agents are emergent in the sense that they have not been there before. In Figure 16 a network of agents is shown. The aggregate of these three agents can be regarded as a "meta-agent", but this meta-agent would have interactions with the outside, unless the whole of the system consists only of the three agents shown. As an example, some rule modification mechanisms (indicated by the double lines) are shown in the figure. As it has been said before, these influences have to be regarded as patterns of only temporary stability, and the connections between input and output are subject to change over time.

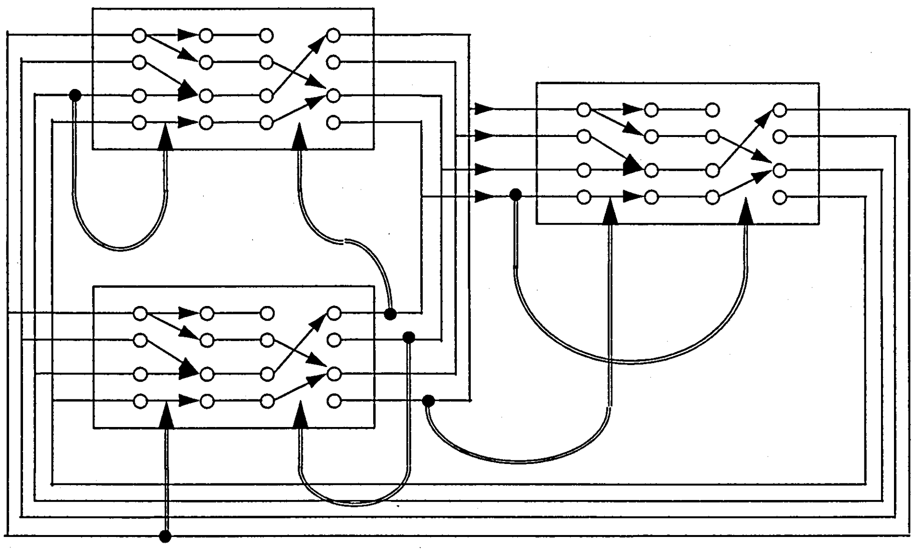
Figure 16: A Network of Agents

The method of how to let the system discover such configurations is very much a problem. On some level this certainly leads to the use of evenly distributed random numbers, as this can be regarded as the least biased method of generating something new. On the other hand it is again an assumption we want to avoid, because this mechanism is the same for all agents in the model, therefore reducing possible causal diversity. The same problem is encountered from the aspect of computational effort: The more assumptions are made on the nature of the system, the less computational effort is necessary, because more things are defined from the outset. The search space of the system is reduced, and the computation is accelerated. When it is desirable in the context of the modelling exercise to explore possible system configurations, the number of assumptions has to be kept as small as possible in order to explore as much as possible of the search space, which in return leads to the use of randomness on high scale levels and subsequent high computational effort.

5.3 Emergent Scales

In conventional dynamic models the right choice of scales is extremely important in respect to what precision of results can be obtained with the model. The scales determine how deep one zooms into the details of the subject matter, and subsequently too little detail might lead to avoidable oversimplification resulting in "simple" system dynamics instead of self-organisation. On the other hand too much detail results in high computational effort, but the kind of model chosen might not bring about more information on the modelled system. Too much information might even distract us from the issues we wanted to investigate.

In an evolutionary agent-based model, however, things are a bit different. The lowest level of resolution is determined by the definition of the agents. The agents are the elementary particles of the model. Agents cannot split, but they can form aggregates and they can be removed from a system as well as new agents can be introduced into a system. With this the highest resolution scale is given by the definition of the agents. Although not all aspects of the agent's behaviour might be required to illustrate and explain the problem, these processes are still taking place in the model.

The constraint to the lowest available scale level is reflected as well in evolutionary dynamics. Here the change in the interactions within the system over time (like in learning individuals) can lead to new aggregations of agents which might find new attractors for the system. In the case of a cyclic attractor a new time scale for the processes obeying this attractor is then emerging. This time scale is then not predefined, but a result of the bottom-up modelling approach. In that sense the bottom-up approach is superior to any other methodology, because it might explain how processes on different scale levels come into existence.

On the other hand when dealing with systems which are known not to change scale levels it might not make much sense to build the whole system from the bottom upwards. In this case the model will take a much simpler form using well-defined dynamics on given scale levels. However, if the system is not well understood and the aim of model-building is to learn more about the system as such (as in this case), it appears to be worthwhile to build the system from the bottom up, unless the effort to do so becomes prohibitive.

In this chapter the question how far scale levels have to be predefined and how far they can be regarded as being intrinsic to the model has to be discussed. The parameters, which are most significantly the subject of scaling, are time, space and agent clusterings (social groups in this example).

5.3.1 Time

Time plays a most important role in a dynamic system. Although time can be regarded as a continuum, in a computer model we have to divide the continuous flow of time into discrete units of time steps. This leads to the use of differential (for quasi-continuous time flows) or difference equations (in the case of discrete time steps), which tell us the rate of change of the system parameters. This is not a problem in principle, if we assume that nothing except for the defined processes happens between point t and t+1. However, we cannot be sure of this, because our differential equation assumes that it is valid for all times between t and t+1 in reality.

The processes taking part on the various time scales can be very different. In human systems we can observe scales which coincide with the natural rhythm of our lives, such as years and the cycle of the seasons, months and weeks which are more culturally defined, and finally days and their parts, measured in hours. An attempt to illustrate all the different time scales and the processes of change affecting an individual's daily life has been made in Figure 17. The general impression is that processes taking place on larger time scales can have a more severe impact than short-term change. The slow processes such as educational degrees, change of living place or retirement occur only very rarely, but might affect an individual's life more severely than a change in the weekly routine.

Short-term decisions such as whether to go out tonight or to stay in usually have only very little impact on the continuity of someone's life (unless you get hit by a car on the way out). The cognitive map containing the individual's image of the environment represents the perceptual part of an agent, and changes in the cognitive map might result in radical alterations of everyday routines. One element constraining the individual's actions and which in reverse is only indirectly influenced are the society's time rules. These represent the conventions a society has set up to organise everyday life. Examples might be public holidays, work and school hours as well as opening times for leisure facilities.

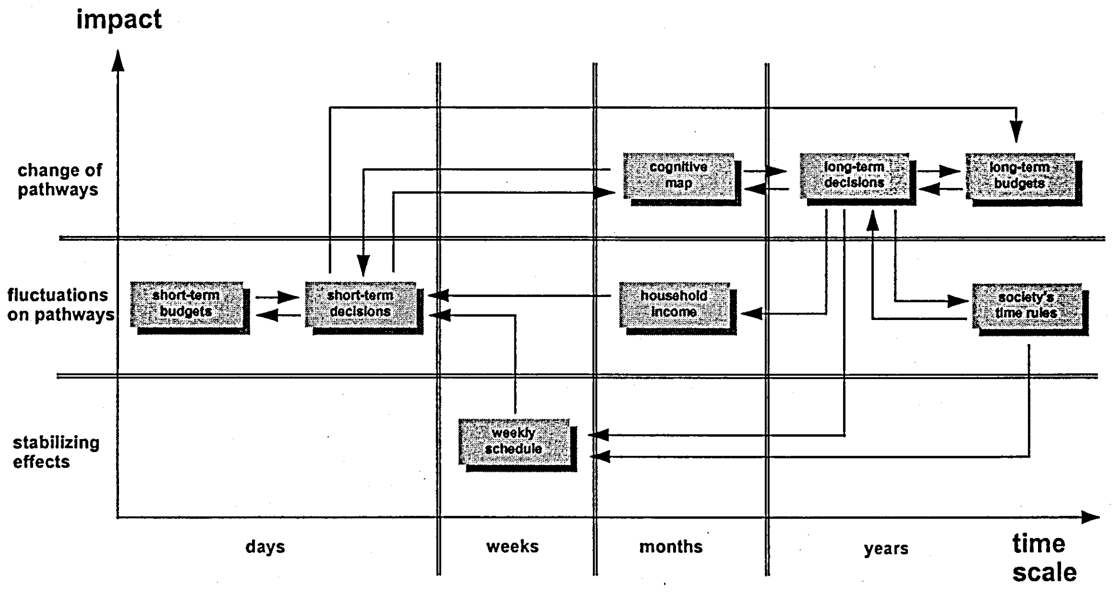

Figure 17: Processes Taking Place on Different Time Scales

All these different time scales are not exclusive to individuals. On an urban level for example, "fast" and "slow" processes have been identified by Weidlich (1997). These short- and long-term processes might as well have contradicting aims or means and weights to the individuals concerned with them. It is therefore quite important to take into account as many time scales as possible. As outlined before, an evolutionary model is regarded to generate these different time scales from within, and therefore the definition of time scales will be intrinsic to the definition of the model itself.

The issue of adaptation is as well linked to the question of time scales. Adaptability will generally take place on a larger time scale than the primary processes as it is a response to the outcomes of many actions which have taken place already; but is it to be assumed to take place on a larger time scale from the outset? The answer to this question should be no, because we do not know whether this is really the case. The evolutionary model should be designed in a way, which allows the model to settle into a specific mode of adaptation depending on its trajectory and state.

5.3.2 Individuals and Groups

As the aim of this project was to build a model of individual acting as a first step towards an integrated urban model, it might seem unimportant to be concerned about the dynamics of groups. Groups obey principles different to those of individuals, but this is effectively just the same scale problem like the one observed on the level of time. Each individual is a member of several groups (a family, at work, in clubs or in a group of friends) and the aggregation of individuals in a group gives again rise to a different type of dynamics depending on the local and temporal conditions encountered. On the other hand people are influenced by their peer groups so that group behaviour is in return influencing individual acting.

These different scales are in fact increasing the complexity of the problem enormously if a conventional model is used. The evolutionary approach however, which explicitly builds on the aggregation of agents into meta-agents, can deal very elegantly with this scaling problem as well. In this sense, the formation of groups comes as a standard feature with the approach.

Like in the case of time scales, it is not obvious that the different dynamics of groups and individuals are not properties, which would come along with a sensible model of individual acting anyway. The "correct" set-up of a model of individual acting would actually give rise to the formation of groups and group behaviour, so that, as in the case of time scales, it might not be necessary to assume a different set of dynamics for groups. This project, however, is focusing on individual behaviour. For the reasons given above, no direct interaction between agents has been included in the actual model, although it is a very interesting extension of this very framework.

5.3.3 Spatial Aggregation Levels

Finally, we observe different dynamics on various spatial scales in the environment of the individuals. These might be in an urban context a neighbourhood, a part of a city, the city itself and a system of cities, all of which are governed by their own set of rules.

This framework will stay with a level of fairly small-scale environments, such as villages, or city neighbourhoods. Given the fact that processes on a higher level of scale might severely influence the behaviour of the system, it is still necessary to take into account all these processes. A simple solution to this problem is the definition of static constraints, which might be changed manually from time to time. However, this does not mean that there will be no emergent properties of the model's population, but it is a restriction to the evolutionary forces within the model.

5.4 Ways of Implementation

The basic framework for an evolutionary agent-based model of a social system is now in place. Before building a model, which can test the framework against reality, it is necessary to clarify some issues related to the practicalities of implementation. A truly evolutionary model is according to our theory all-dynamic and has practically no static parameters. In the course of a simulation anything might have an influence on everything else, or not, depending on the course of the model run.

In a first step towards such a model it is, however, useful to restrict the free interaction of a certain extent to make the analysis of the results easier. The full evolutionary mechanism is traded off with the chance to understand more about the precise interactions taking place in the model. Therefore it has to be determined how much of the environment and the agents should be defined as invariant at the beginning of a simulation, and what assumptions on system constraints are useful for the application in question.

5.4.1 A Question of Life and Death

Whilst trying to find working rules on which to base their behaviour, it is to be expected that a large proportion of the agents will - at least in the beginning and depending on the initial conditions - fail to find good rule sets. The question how to treat these failed agents has to be discussed before setting out to implement any model.

The question to be dealt with in this project is mainly concerned with the creation of adaptive agents, which are to be used in a larger scale model of an urban environment. The agents have to prove their adaptability in a co-evolving environment. The adaptive capabilities of the agents have to be regarded as basically cultural and not biological properties, although much of this conceptual framework is building on analogies to biological evolutionary theory. It can therefore be argued that this kind of model does not have to be concerned with population ecology, and that for this reason population ecology related issues can be neglected.

There are two basic ways of dealing with unsuccessful agents: Agents can be removed from the population, or they can be reset and reintroduced into the existing population. The first alternative represents the biological, population ecology based approach. When opting for this strategy, the question of how new agents with new strategies come into existence has to be addressed as well. This inevitably leads into issues of population dynamics, which, as we have outlined before, are not the main concern of this modelling exercise, although a model of population dynamics will finally make up part of an agent-based urban model.

For the reason mentioned above, the simpler "reset at failure" strategy will be adopted. The number of agents in the system will remain the same over time, all of them searching for rule sets, which would satisfy their intrinsic needs. The analogy is clearly that the society grants unsuccessful agents "dept relief" in the sense of a welfare state. This does of course not correspond to a working economy, but as a first step it should be sufficient to evolve rules sets with a fixed number of agents.

5.4.2 Possibility Spaces

In the context of implementation the question of how the system should be constrained is certainly an important issue. In real life we encounter many constraining factors in our daily lives. For instance, shops are not open at all times and most people are required to be at work between certain hours. Work takes place only on certain days, other days are reserved for leisure or other personal use. All these time rules are however not there by default, and many other arrangements appear to be possible. These time rules are the result of a cultural evolution.

In most models this kind of constraints would be assumed to be static and invariant over the time of the simulation. In an evolutionary model on the other hand we have got the chance not to do so, but to evolve the system constraints together with the system. This gives us the advantage to explore many more possible configurations, which might be as valid as the ones we already know, and to draw conclusions on how the system constraints are induced by the system and its history. The down side to this approach comes with the limited reproduction of the observed system, so that it might be possible that the observed system configuration appears only in a fraction of all model runs.

All agents (inhabitants, organisations, the natural environment) in an evolutionary model create most of their possibility space by mutual interaction. The system constraints are usually the result of such interaction: The system gets locked into a certain mode of behaviour by self-organisation. In most applications we will not implement the full evolutionary approach for practical reasons, because it takes considerable effort to allow the system to evolve from a random "soup". This approach would aim at recreating the cultural evolution of the past four million years in order to arrive at the present state. The time it takes to let the system self-organise itself is not always available, and many processes taking place on large time scales might just be assumed to be invariant over the duration of a model run. The important thing here is to keep in mind that this is not really so. If it is possible to test the assumption of static constraints against the evolving mechanism, it should always be done, as the change of self-induced constraints will have important repercussions on the future behaviour of the system. The change in constraints might result in a major new self-organisation event pushing the system away from the current attractor and into a new trajectory, which can open up whole new, previously unknown, pathways.

5.5 Summary

In this chapter we have seen how the evidence from the previous chapters is transformed into a conceptual framework for the simulation of adaptive individual behaviour. This framework provides a methodology, which can overcome many of the limitation of earlier models.

The basic elements of the framework comprise:

i) The Agent-Based Approach

- The bottom-up construction of a social systems model based on computational entities called "agents" is the prerequisite for

ii) The Evolutionary Principle

- In an evolutionary system the system elements specify each other mutually through repeated events of self-organisation. Self-organisation provides the system with periods of temporal stability. This applies to the state of the system elements as well as to the dynamics of the system.
- The mutual specification the dynamics of the system give rise to new, emergent interactions in the model leading to
- Emergent scale levels.

Because the evidence on which this framework is built has been drawn from research into cognition and evolution, the concepts are regarded to be generic to social systems. The theme of everyday life in an urban setting in this project can serve as just one example to the manifold areas of application. In the next chapters the next step in the process of investigation will describe how this framework is validated using a computer model based on the essential points made here.

6 An Adaptive Agent-Based Simulation Model

The conceptual framework developed in the previous chapters is now to be implemented in the form of a computer model in order to test the conceptual framework as well as to possibly discover new, emergent features of such a system.

The example chosen is an urban environment. The agents in the model are trying to live a "daily life", e.g. they have the choice to carry out a certain set of activities which resemble real-life activities like recreation, work, socialising or shopping. The agents will have to maintain a careful balance between these activities in order to satisfy their intrinsic needs. For this purpose they will have to develop a set of behavioural rules which govern the decision making process of the agents. These rules - as they are also the basis for the evaluation of need satisfaction - can be interpreted as an individual value system, too. The model is therefore mapping the necessary cultural evolution of a set of agents in order to develop a working cooperative artificial society working to the benefits of all agents.

6.1 General Features and Specification

The model is implemented using an agent-based approach as outlined in Chapter 5.1. Agents are autonomous entities equipped with a defined set of properties, which are modelled on certain traits of humans or other individual entities. The agents share a common environment with possibly more, different types of agents, such as economic enterprises or regulatory units which obey their own dynamics. However, in this context, the term "agents" will be used to denominate only those computational agents, which inhabit the environment and try to find a life-style, which sustains their intrinsic needs.

The dominating constraint for all agents is the fact that a day pattern is imposed upon them. This means that there are 24 model hours per model day, and in each of these model hours the agents can pick exactly one activity out of a set of available activities. As the activities cover all possible opportunities what the agents can do, it is not possible for them to do "nothing". Doing "nothing" is regarded as an activity in its own right.

Predefined time scales can have profound effects on the performance of a model. Here, the only time scales imposed are the model day, which consists of 24 model hours as the smallest unit of time. All phenomena which might occur on cycles larger than one day are due to the interactions within the model itself.

One important feature of all computer models is the question how to treat events, which in nature would occur simultaneously. As conventional computers allow only for serial (one-at-a-time) processing it is necessary to create a kind of pseudo-synchronicity in the model to prevent some agents being disadvantaged by the processing order. This is realised by giving all agents a new random order at the beginning of each model day to make sure that at least the disadvantage is evenly spread among the agents.

All agents are treated as individuals and therefore can take on unique configurations of their basic properties, which, of course, have to be predefined to be treated with a conventional computer program (see 2.5). In this sense the agents create diversity not only of behaviours (which can already be achieved using a single, uniform configuration (see 7.4)), but of cultural attributes like value systems and individual behavioural rules. Therefore the diversity within the system is created by the system itself and is not imposed from the outside by the modeller.

As the objective of the model is to generate cultural evolution within a small artificial society, agents are not removed upon being unsuccessful, but being reset when a threshold is reached. This reset procedure keeps the total number of agents in the system constant over time. In practice, this means that all unsuccessful agents get another try in order to find a successful rule set, once they have failed to do so (see 6.3.5).

6.2 Model Components

The model consists of basically two components: agents and their environment. The environment can be regarded as one (or more) agent(s) as well in the sense of the methodology (5.1). Each of the components has specific tasks and properties and acts independently, although the actions of any agent affect the possibility space of all other agents.

Agents are therefore influencing each other only indirectly via induced changes in the common environment. There is no direct communication between the agents in the model, although there exists a possibility to spread successful rule sets between the agents via a blackboard system (see 6.3.5). This feature holds the rules sets of successful agents which write to the blackboard, and can be accessed by unsuccessful agents in order to copy a good rule set.

The environment has a more responsive role in the model. It has two basic properties, its capacity for certain activities and a price for these activities. Both price and capacity can adapt - if desired - according to the average demand for the activities in question (see 6.4). Both environment and agents are described in detail in the following sections below.

6.3 Agents

The heart of the model are the agents which inhabit it. The agents' set-up follows the conceptual framework as closely as possible, although limitations had to be made as the conceptual framework leads to extreme complexity in a model's behaviour (but probably not that much for the structure), which makes the analysis of such a model's results very difficult.

The agents are acting in a capacity constraint environment, which leads to competition for scarce resources, and carrying out an activity does not necessarily lead to a payoff. But cooperation between the agents is also necessary to a certain extent, because a division of labour is assumed in the environment, so agents are required to carry out certain activities in order to enable others to do different things.

The model is working on only a short-term day-to-day basis, e.g. the agents have no ability to plan ahead or to account for rare events, such as moving house etc.. The agents are acting as single individuals and their ability to communicate with other agents is severely restricted, so that ideas like households and/or families are not catered for in the model. As opposed to the approach taken by Gärling et al. (1998), (see also 2.3.9) the agents are short-termists and do not possess anything specifically designed to create routines or schedules, but they do possess a cognitive map of their environment. On the other hand the experience from the spreadsheet model (4.3.2) shows that routines can emerge from a model in the form of cyclic patterns of behaviour without assuming beforehand that such processes exist.

Apart from this the agents feature a set of short-term budgets and activities representing their intrinsic needs. The time rules of the artificial society are incorporated not as a result of the agent's acting but as an external constraint by limiting the payoffs of all possible activities to certain times of the day.

6.3.1 Budgets and Needs

The driving force for the agents is a set of needs (see Chapter 4.3). These needs are expressed through and measured by a set of associated budgets. As these budgets account for "physiological" processes, they are subject to a quasi-linear decay rate, which means that a constant number of units are subtracted from each budget at the beginning of each model day.

The model is programmed in a way that allows the budgets to be named as required with the area of usage. Here, as the agents are acting in an urban setting, the budgets are called

1. Recreation [Unit: time]
2. Money [dimensionless units], no decay rate
3. Goods [units]
4. Socialising [time]

The recreation budget accounts for a need to sleep and relax. Spending time on the activity recreation can fill this budget. The money budget on the other hand cannot be interpreted as measuring a need for having money, as such a need cannot be deducted from the theoretical framework or any of the literature on human needs (Maslow, 1954, Max-Neef, 1991). It is simply an economic convention and needed to acquire goods and services.

Goods are consumed by the agents at a uniform rate, and have to be purchased with money. Social contacts at last feed the fourth budget, socialising. Here again the agents have to spend time on socialising in order to increase this budget state. As opposed to recreation, there is a price to this activity so that the agents have to spend money as well in order to get any payoff from this activity.

6.3.2 The Fuzzy Logic Rule Bases

The agents are using two Fuzzy Logic rule bases for decision making. The first rule base is active on a daily basis and the second one on the hourly evaluation of alternative activities and places. Fuzzy Logic was chosen, because it allows the classification of system states on a very simple basis. Once a set of descriptors for the parameters to be classified is defined, Fuzzy Sets can be allocated to measure to what degree the descriptors apply to the current state.

In this case the descriptors for the state of the agent can be easily established. Each agent has its set of budgets (see above) which is filled by the payoffs of the agent's activities and which is subject to a static decay rate. These budgets are meant to represent the agent's set of intrinsic needs. Therefore the state of these budgets can be taken as a measure of the agent's well-being. The classification has to take place on an individual basis as each agent has a different concept of what a measured budget state means to the individual.

As opposed to Aristotelian logic, Fuzzy Logic (see 4.7.2) allows for more than the two states of "true" and "false". It rather measures the degree to which a Fuzzy Set (describing a concept, or a classification) applies to the encountered state. In the case of the discussed model, this means that each agent applies an individual rating of whether a budget state is "High" or "Low". This classification determines the action to be taken using rules like "If my *money budget* is LOW, then *work* is IMPORTANT to me.". To make this rule operational, it is necessary to define what "LOW" and "IMPORTANT" mean.

The Fuzzy Sets are shaped as logistic curves (Figure 18) defined by the point where their value equals 0.5 and their slope determining how fast the extreme values of 0 and 1 are reached. These two parameters can be set individually for each agent, so that it might be the case that a given budget state can mean 75% "low" and 30% "high" to one agent whilst another one classifies this state as 45% "low" and 65% "high". The degrees to which the classifications apply are proportional to weights, which are given to the associated consequences according to the rules. In practice, all rules fire at all times, but most of them would only apply to the degree of zero.

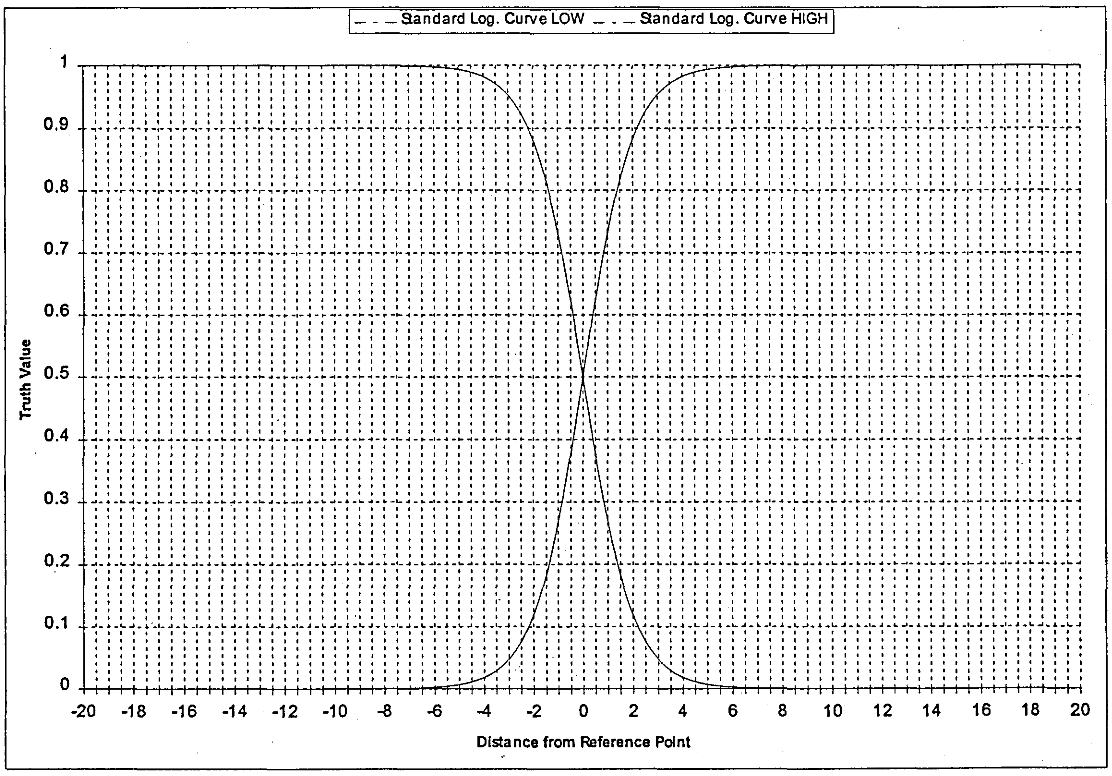

Figure 18: The Fuzzy Sets Characterising the Budgets are Shaped as Logistic Curves

This can be interpreted as the rules finding a "consensus" over what to do, because the result will always be a weighted average of what the rules would effect when being applied fully. To keep the model as simple as possible, only the most basic characterisation with two Fuzzy Sets per parameter (representing the states HIGH and LOW as in Figure 18) was used. These input sets characterise a situation encountered by the agent. In order to draw a conclusion for action from this situation, an inference mechanism has to be defined, too, which leads to the definition of a rule. For this it is necessary to define output sets as well. In the simplest form (which has been adopted here) these are shaped as triangles centred over an output value. This output value is the result of a rule if it applies to a degree of 1. Each rule connects via the inference mechanism one or more input sets with an output set. The truth value of the state classification captured by the input sets is mapped onto the output sets. Because classifications rarely apply to the degree of 1, the output sets are cut off at the respective fraction of their height. For instance if an input set applies to a degree of 65%, the corresponding output set is cut off at 65% of its height and reducing the triangle to a trapezoid of this height. This reduces the area of the output set. In the next step of the algorithm all the resulting centre of gravity of all output sets is calculated, which is the final result of the inference process. All rules applied to the inference influence the final result to the degree they apply.

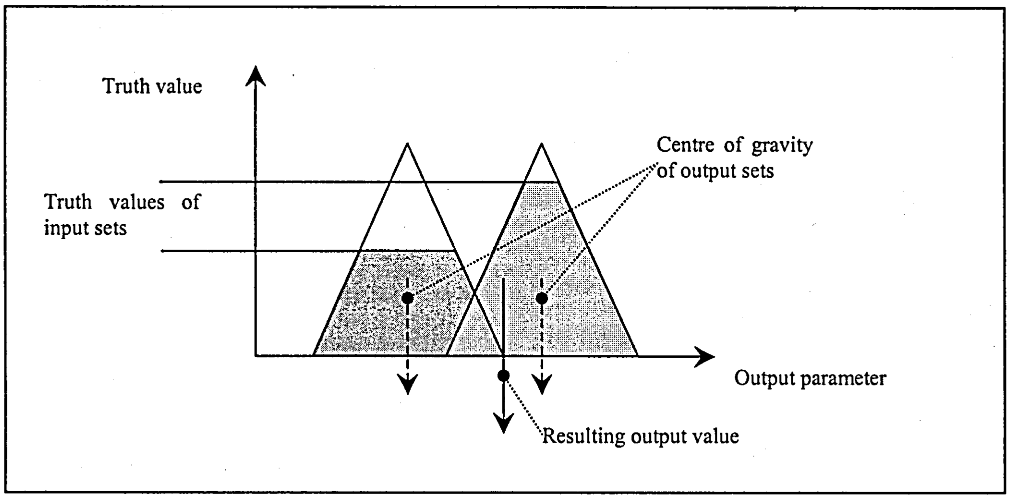

Figure 19: Output Sets of a Fuzzy Logic Rule Base

The most important feature of the rule bases is that it is possible to connect any input parameter with any output parameter using AND and OR operators during run time. Up to four rules per output parameter can be defined to account for one AND combination of input parameters (and its opposite) as well as an alternative (OR) inference to be made.

The input parameters of the first rule base are the budgets of an agent. The output parameters are importance factors for each activity, determining how important each activity is for the agent. As this rule base determines the behaviour of the agents to a good part, and this will be the rule base in which the adaptation features (see 6.3.5) are implemented, it will be referred to as "behavioural rule base". The second rule base is used at every hourly time step, and it will not be changed at any point during a simulation. Its function is described in Section 6.3.3 below.

6.3.3 Decision Process

The agents are going through a two-stage decision process in order to choose what activities to pick. At the beginning of each model day an evaluation of the agents' budgets takes place. According to the rules in the behavioural rule base each agent evaluates its budgets at the beginning of each model day. This results in an importance value on a scale between 0 and 100 for each activity. These values are then converted into relative importances and multiplied with 24 (the number of hours per model day) to give the time the agent wants to spend on each activity on that day.

A second stage in the process takes place at the beginning of each hour of the model day. As the agent has a plan for the day, it now tries to find the most advantageous activity and place. Here a second rule base determines a utility value for each known alternative (up to four for each activity), taking into account

- The possible payoff for the activity at that point in time
- The distance from the current position to the place in question
- The time still to be spent on this activity
- The previous success in taking this alternative

The alternative with the highest utility value is subsequently chosen. The second step favours agents to go to the nearest place with the highest payoff, if they have been successful at this place in the past, and they still want to spend much time on the activity in question. The term "utility" denotes the personal perceived usefulness of a given alternative to the agent. This disaggregation leads to individualised multicriteria utility functions. For this reason it must not be confused with the term utility as used in Chapter 2.3.6.

Although the model allows for non-uniform rules and rule parameters at this stage, it is kept the same for all agents over time in order to focus on changes in the behavioural rule base determining the overall importance of the activities. This is the subject of Section 6.3.5.

6.3.4 Cognitive Map

The theoretical framework demands that the agents have only incomplete knowledge of their environment. This is realised through definition of a cognitive map for the agents. In terms of cognitive science this describes an individual perception of the environment by the agents. This perception determines the features the individual knows about as well as qualitative attributes of the known environment.

The cognitive maps of the agents allow them to have knowledge of up to four places for each activity, e.g. all in all up to 16 places. The cognitive map is combined with a count showing how successful the agent has been in the past trying to carry out the activity in question in that place. The count is limited to the range between -100 and +100 and is increased by one each time the agent is rewarded for the activity. On the other hand the count is decreased by one if the agent has not been able to obtain any payoff during the last period it was there.

The cognitive map is static over the simulation time in as far as the places are concerned, of course the rating of the places changes over time. The cognitive map determines the individual preferences of the agents based on their experience, albeit in a very much simplified way.

6.3.5 Learning Features

The model allows for several different ways of adaptation of the agents' behavioural rule base. One mechanism is based on correlation between budgets and their gradients, the other two are using random numbers to generate new rules. The agents can choose both the algorithmic and the random approach alternatively.

In the standard configuration, the behavioural rule base is not changed at any time, thereby presenting a standard agent and rule based model. All knowledge about the agents and their behaviour has to be assumed a priori as no learning takes place during the run. The agents' budgets are reset to a predefined value (usually 0) when the agent fails to balance all four budgets above the defined reset threshold (-500 units for example). Upon reset, the agent continues with the same rules as before, just the budget states have been adjusted.

The easiest way to search for better rules if the initial settings do not render the agent capable of keeping all budgets above the reset threshold is to generate new rules at random. In this setting, all rules connecting the budgets with the importance of the corresponding activities are replaced by random new rules once one of the budgets falls below a fixed fraction of the reset threshold. If the new rules are successful, e.g. the budgets are all above the mutation threshold after a certain time, the rule set is kept, otherwise the rule set is replaced by a new one.

This method is sampling the entire search space, which means that the probability of finding good rule sets is very small. As there are four input parameters with two fuzzy sets feeding into four rules for four activities, the total number of possible permutations is

$$\begin{aligned} 2^{sets \cdot rules \cdot activities} &= 2^{8 \cdot 4 \cdot 4} \\ &= 3.4028 \cdot 10^{38} \end{aligned}$$

Assuming that viable rule sets constitute only a fraction of this enormous space, simple random mutation appears not to be entirely effective.

A second way of generating new rules is to draw a random number only for those rules, which affect the budgets currently below the mutation threshold. This assumes that changing the way one budget is treated does not affect the other budgets. This does probably not hold, given the fact that the same characterisation of budgets can produce different behaviour in different circumstances. This method does however have the advantage that the search space is much smaller than before (fourth root of before), so that it might be more probable to find a viable rule set.

As opposed to the random generation of rules described before the model features an algorithmic mechanism as well. It is based on the correlation between two budgets and their tendency and works as a Fuzzy Logic rule base. As the budget states are the input parameters for the rule base to be changed, the algorithm includes or excludes parameters from the process of finding the importance of activities.

If a budget is classified as "LOW" by the behavioural rule base, it triggers the following procedure: The algorithm determines whether the budget itself and all other budgets had an upward or downward tendency during the last number of time steps. If the budget in evaluation had a downward tendency then the algorithm looks for other budgets with the opposite tendency assuming that there is a correlation between the budgets. All budgets with that tendency will be included in the rule set for the evaluated budget/activity, whilst all budgets with the same tendency as the evaluated one are to be excluded from the rule set. As the rule base allows for the formation of AND as well as OR combinations of rules, there is a pre-set probability (usually 0.5) to form either combination.

Whilst all discussed methods of acquiring a new rule set to this point relied on innovation, a second way of reaching the same objective relies on exploitation of knowledge in existence (if one likes to call the rule sets a form of knowledge). To allow for this, agents have a pre-set probability of copying a successful rule set from a so-called blackboard. The blackboard is a collection of rule sets, which have been written there by agents, which have reached the age of 500 time steps, or above.

Agents which copy rule sets do so at random, as well as the successful agents write to a random slot on the blackboard. This can result in having much less different rule sets than possible on the blackboard as one agent might write at successive points in time to different slots, which have not been overwritten by other agents.

In a first step all strategies were to be tested on their own, so that this version of the model allowed for only one strategy to be adopted by the agents. The copying feature could be selected in addition to this. As the next objective of the project was to test whether "learning" (or adaptation) was an emergent feature of the system, the second version of the model enabled the agents to chose between the non-adaptive mode and any of the adaptation strategies at reset. Two options on how the switch from one strategy two another takes place are provided for: The experimenter can choose between an even random draw between the strategies or choose to exclude the previous mode from the draw. A probability determines how likely a change of strategy is. As the point of resets can be regarded as the time of "death" respectively "birth" of an agent, the biological analogy of this probability is the idea that reproduction is imperfect, and that errors occur in the transmission of information.

6.3.6 Example Calculation

This section aims at demonstrating how the model algorithm works in practice. As one model day involves a multitude of loops over the different agents, activities, known alternatives, and hours, only the simplest version using one agent over a limited time will be demonstrated. As the principles of the algorithm will become clear in the course of the calculation, only two activities – "sleep" and "work" will be used. More activities will only have an effect on the total time allocated for the activities, as more budgets will have to be considered. In Figure 20 the extent of the example calculation is indicated in the flow-chart of the complete model.

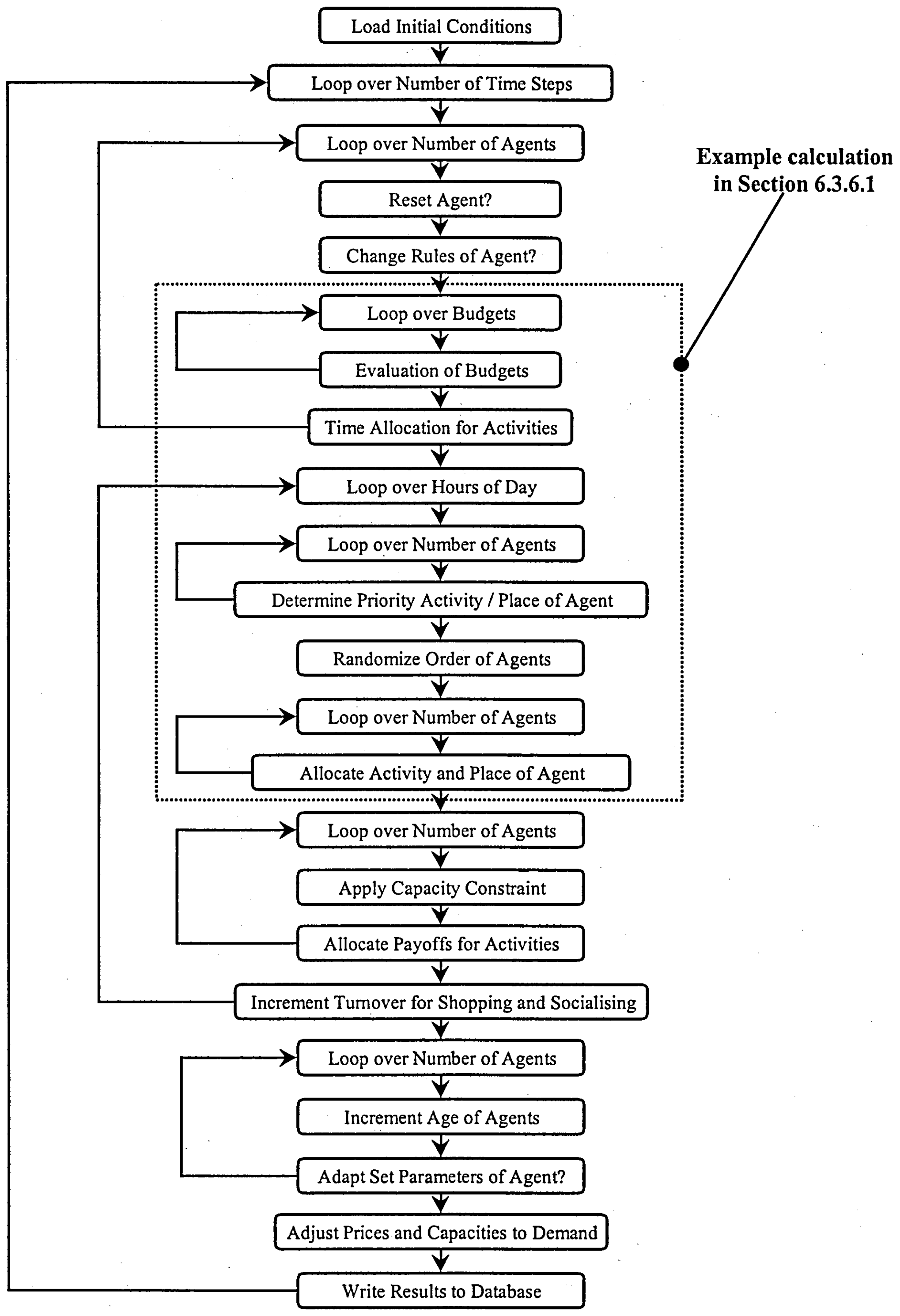
Load Initial Conditions
Loop over Number of Time Steps
Loop over Number of Agents
Reset Agent?
Change Rules of Agent?
Loop over Budgets
Evaluation of Budgets
Time Allocation for Activities
Loop over Hours of Day
Loop over Number of Agents
Determine Priority Activity / Place of Agent
Randomize Order of Agents
Loop over Number of Agents
Allocate Activity and Place of Agent
Loop over Number of Agents
Apply Capacity Constraint
Allocate Payoffs for Activities
Increment Turnover for Shopping and Socialising
Loop over Number of Agents
Increment Age of Agents
Adapt Set Parameters of Agent?
Adjust Prices and Capacities to Demand
Write Results to Database
Example calculation in Section 6.3.6.1

Figure 20: Flow-chart of the Computer Model

6.3.6.1 Definitions

At the beginning of each model day the agents check their budgets and decide how much time of the following day they want to spend on each available activity. In this example, only two budgets and activities were used. The budgets were called "*tired*" and "*money*", which can filled by performing "*sleep*" or "*work*" respectively. There is a decay rate of 8 units per day to each budget, e.g. at the end of each model day 8 units are deducted from the budgets.

Suppose the 9 places of the model world are all equal and allow for a reward of 1 unit/hour for sleep in hours 0 to 6 and 19 to 23. The rewards for work are 1 unit/hour from hour 7 through 18. During hour 0 to 6 and 19 to 23 there is no reward for performing "work" and no reward for "sleep" during hour 7 to 18. This means that ideally the agent can add 12 units to each budget while 8 units are decaying each day.

The distance between all places is 10 units on a square grid, giving the coordinates (0,0); (0,10); (0,20); (10,0); (10,10); (10,20); (20,0); (20,10); (20,20). The home place of the agent is (0,0). The definition of a home place means that after each model day the agent will return to this place.

The agents knows the places (0,0) and (0,10) for "sleep" and (0,20), (10,10), (10,20), and (20,0) for "work".

For the evaluation of the budgets the following rules are defined:

- If "tired" (budget) is *low* then "sleep" has an importance of 100. (This definition is the opposite of reality, it actually refers to the amount of time the agent has spent sleeping during the past.) (RULE 1)
- If "tired" is *high* then "sleep" has an importance of 0. (RULE 2)
- If "money" is *low* then "work" has an importance of 100. (RULE 3)
- If "money" is *high* then "work" has an importance of 0. (RULE 4)

The fuzzy sets low and high have to be defined separately for both budgets:

Low: $t(low) = \dfrac{1}{1 + \exp(sigma \cdot (x - centre)}$

t is the degree to which "low" applies to the current budget's state, sigma is the slope of the fuzzy set, the greater sigma, the more t resembles a step function. The centre position gives is the point where t=0.5.

High: $t(high) = \dfrac{1}{1 + \exp(-(sigma \cdot (x - centre))}$

The only difference between the two definitions is that t(low) has an extreme value of 1 for x = -infinity, reaching 0 for x = +infinity, while t(high) starts with 0 and reaches 1 for the same x-values.

The parameters have been set to

"tired": $\text{sigma}_{tlow} = 1$

$\text{sigma}_{thigh} = 1$

$\text{centre}_{tlow} = 0$

$\text{centre}_{thigh} = 10$

"money": $\text{sigma}_{tlow} = 1$

$\text{sigma}_{thigh} = 1$

$\text{centre}_{tlow} = 0$

$\text{centre}_{thigh} = 60$

It is also necessary to define the rules for the choice of activity and place at each hour. These rules have been set to

- If the "reward" (for performing this activity at this time of day in this place) is *high* AND the "distance" (to that opportunity) is *low* AND the "remaining time" (for that activity) is *high* then this opportunity has a utility of 100. (RULE 5)
- If the "reward" is *low* AND the "distance" is *high* AND the "remaining time" is *low* then this opportunity has a utility of 0. (RULE 6)

The functions used to determine the degrees to which the classifications *low* and *high* apply to the current situation are the same as for the budget evaluation, only the function parameters are different.

"reward": $\text{sigma}_{tlow} = 15$

$\text{sigma}_{thigh} = 15$

$\text{centre}_{tlow} = 0.5$

$\text{centre}_{thigh} = 0.5$

"distance": $\text{sigma}_{tlow} = 1.7$

$\text{sigma}_{thigh} = 1.7$

$\text{centre}_{tlow} = 3$

$\text{centre}_{thigh} = 3$

"remaining time": $\text{sigma}_{tlow} = 0.4$

$\text{sigma}_{thigh} = 0.4$

$\text{centre}_{tlow} = 1$

$\text{centre}_{thigh} = 4$

The defuzzyfication sets are all identical, except for their central position, given by the "then" condition in the rules. The base width is set to 30 units for all sets.

It is also necessary to define the initial conditions for the budgets. It shall be assumed that both budgets are set to 0.

6.3.6.2 Budget Evaluation

Having all necessary parameters set, the model day can begin with the agent evaluating its budgets.

Activity "sleep"

RULE 1: $t(low) = \dfrac{1}{1+\exp(1\cdot(0-0)}$
$= 0.5$

RULE 2: $t(high) = \dfrac{1}{1+\exp(-1\cdot(0-10)}$
$= 4.54\cdot 10^{-5}$

Now t(low) is mapped on the output set centred on 100 , while t(high) is mapped on the output set centred on 0. Then the centre of gravity of both output sets is calculated.

height of set_{100}: 0.5

area of set_{100}: $0.5\cdot \mathit{base\ length}\cdot 0.5 = 0.5\cdot 30\cdot 0.5$
$= 7.5$

height of set_0: $4.54\cdot 10^{-5}$

area of set_0: $4.54\cdot 10^{-5}\cdot 30\cdot 0.5 = \underline{\underline{6.81}}\cdot 10^{-4}$

Centre of gravity of both areas:

$$\frac{centre(set_{100})\cdot area(set_{100}) + centre(set_0)\cdot area(set_0)}{area(set_{100}) + area(set_0)} = \frac{750+0}{7.5+6.81\cdot 10^{-4}}$$
$$= 99.99$$

The centre of gravity equals the importance of "sleep"

Activity "work"

RULE 3: $t(low) = \dfrac{1}{1+\exp(1\cdot(0-0)}$
$= 0.5$

RULE 4: $t(high) = \dfrac{1}{1+\exp(-1\cdot(0-60)}$
$= 8.76\cdot 10^{-27}$

height of set_{100}: 0.5

area of set_{100}: $0.5\cdot \mathit{base\ length}\cdot 0.5 = 0.5\cdot 30\cdot 0.5$
$= 7.5$

height of set_0: $8.76\cdot 10^{-27}$

area of set$_0$: $8.76 \cdot 10^{-27} \cdot 30 \cdot 0.5 = 1.31 \cdot 10^{-25}$

Centre of gravity of both areas:

$$\frac{centre(set_{100}) \cdot area(set_{100}) + centre(set_0) \cdot area(set_0)}{area(set_{100}) + area(set_0)} = \frac{750+0}{7.5+1.31 \cdot 10^{-25}}$$
$$=100$$

Now the relative importances for both activities have to be determined:

$$\begin{aligned} rel.\,importance(sleep) &= \frac{importance(sleep)}{importance(sleep)+importance(work)} \\ &= \frac{99.99}{99.99+100} \\ &= 0.499 \end{aligned}$$

$$\begin{aligned} rel.\,importance(work) &= \frac{importance(work)}{importance(sleep)+importance(work)} \\ &= \frac{100}{99.99+100} \\ &= 0.500 \end{aligned}$$

In the next step the time of the day is allocated to the activities according to their relative importance:

$$\begin{aligned} time(sleep) &= 24hrs. \cdot rel.\,importance(sleep) \\ &= 0.499 \cdot 24hrs. \\ &= 11.976hrs. \end{aligned}$$

$$\begin{aligned} time(work) &= 24hrs. \cdot rel.\,importance(work) \\ &= 0.500 \cdot 24hrs. \\ &= 12.024hrs. \end{aligned}$$

These values are the basis for the next decision step determining which activity and place are to be picked in each hour.

6.3.6.3 Decision on Activity and Place

This procedure evaluates all known alternatives for activities taking into account the possible reward for the activity, the distance to the place, and the remaining time for that activity.

Hour 0

- Activity "sleep" known places: (0,0) (=current position); (0,10)
  reward in all places in hour 0: 1 unit/hour

Rule 5: place (0,0)

$$t(\mathit{reward\ high}) = \frac{1}{1+\exp(-15\cdot(1-0.5)} = 0.99$$

$$t(\mathit{distance\ low}) = \frac{1}{1+\exp(1.7\cdot(0-3)} = 0.99$$

$$t(\mathit{remaining\ time\ high}) = \frac{1}{1+\exp(-0.4\cdot(11.976-4)} = 0.96$$

These parameters are connected with the AND operator, which means that only the smallest t (being 0.96) is considered in the further calculation.

Rule 6: place (0,0)

$$t(\mathit{reward\ low}) = \frac{1}{1+\exp(15\cdot(1-0.5)} = 5.52\cdot10^{-4}$$

$$t(\mathit{distance\ high}) = \frac{1}{1+\exp(-1.7\cdot(0-3)} = 6.05\cdot10^{-3}$$

$$t(\mathit{remaining\ time\ low}) = \frac{1}{1+\exp(0.4\cdot(11.976-1)} = 0.01$$

Again, we only consider the minimum, which is 5.52 E-04.

Rule 5 produces a value of 100, while Rule 6 produces 0.

Centre of gravity (utility of alternative):

$$U = \frac{100\cdot0.96\cdot30\cdot0.5+0\cdot5.52\cdot10^{-4}\cdot30\cdot0.5}{0.96\cdot30\cdot0.5+5.52\cdot10^{-4}\cdot30\cdot0.5} = 99.94$$

Now the next alternative is evaluated.

Rule 5: place (0,10)

$$t(reward\ high) = \frac{1}{1+\exp(-15\cdot(1-0.5)} = 0.99$$

$$t(distance\ low) = \frac{1}{1+\exp(1.7\cdot(10-3)} = 6.79\cdot 10^{-6}$$

$$t(remaining\ time\ high) = \frac{1}{1+\exp(-0.4\cdot(11.976-4)} = 0.96$$

minimum: 6.79 E-06

Rule 6: place (0,10)

$$t(reward\ low) = \frac{1}{1+\exp(15\cdot(1-0.5)} = 5.52\cdot 10^{-4}$$

$$t(distance\ high) = \frac{1}{1+\exp(-1.7\cdot(10-3)} = 0.99$$

$$t(remaining\ time\ low) = \frac{1}{1+\exp(0.4\cdot(11.976-1)} = 0.01$$

minimum: 5.52 E-04

Utility:

$$U = \frac{100\cdot 6.79\cdot 10^{-6}\cdot 30\cdot 0.5 + 0\cdot 5.52\cdot 10^{-4}\cdot 30\cdot 0.5}{6.79\cdot 10^{-6}\cdot 30\cdot 0.5 + 5.52\cdot 10^{-4}\cdot 30\cdot 0.5} = 1.21$$

- Activity "work" known places: (0,20), (10,10), (10,20), (20,0)
  reward for "work" in hour 0: 0 units/hour

Rule 5: place (0,20)

$$t(\textit{reward high}) = \frac{1}{1+\exp(-15\cdot(0-0.5)} = 5.52\cdot 10^{-4}$$

$$t(\textit{distance low}) = \frac{1}{1+\exp(1.7\cdot(20-3)} = 2.81\cdot 10^{-13}$$

$$t(\textit{remaining time high}) = \frac{1}{1+\exp(-0.4\cdot(12.024-4)} = 0.96$$

minimum: 2.81 E-13

Rule 6: place (0,20)

$$t(\textit{reward low}) = \frac{1}{1+\exp(15\cdot(0-0.5)} = 0.99$$

$$t(\textit{distance high}) = \frac{1}{1+\exp(-1.7\cdot(20-3)} = 1.0$$

$$t(\textit{remaining time low}) = \frac{1}{1+\exp(0.4\cdot(12.024-1)} = 0.01$$

minimum: 0.01

Utility:

$$U = \frac{100\cdot 2.81\cdot 10^{-13}\cdot 30\cdot 0.5 + 0\cdot 0.01\cdot 30\cdot 0.5}{2.81\cdot 10^{-13}\cdot 30\cdot 0.5 + 0.01\cdot 30\cdot 0.5} = 2.81\cdot 10^{-9}$$

Rule 5: place (10,10)

$$t(\mathit{reward\ high}) = \frac{1}{1+\exp(-15\cdot(0-0.5)} = 5.52\cdot 10^{-4}$$

$$t(\mathit{distance\ low}) = \frac{1}{1+\exp(1.7\cdot(14.14-3)} = 5.96\cdot 10^{-9}$$

$$t(\mathit{remaining\ time\ high}) = \frac{1}{1+\exp(-0.4\cdot(12.024-4)} = 0.96$$

minimum: 5.96 E-09

Rule 6: place (10,10)

$$t(\mathit{reward\ low}) = \frac{1}{1+\exp(15\cdot(0-0.5)} = 0.99$$

$$t(\mathit{distance\ high}) = \frac{1}{1+\exp(-1.7\cdot(14.14-3)} = 0.99$$

$$t(\mathit{remaining\ time\ low}) = \frac{1}{1+\exp(0.4\cdot(12.024-1)} = 0.01$$

minimum: 0.01

Utility:

$$U = \frac{100\cdot 5.96\cdot 10^{-9}\cdot 30\cdot 0.5 + 0\cdot 0.01\cdot 30\cdot 0.5}{5.96\cdot 10^{-9}\cdot 30\cdot 0.5 + 0.01\cdot 30\cdot 0.5} = 5.95\cdot 10^{-5}$$

Rule 5: place (10,20)

$$t(reward\ high) = \frac{1}{1+\exp(-15\cdot(0-0.5)} = 5.52\cdot 10^{-4}$$

$$t(distance\ low) = \frac{1}{1+\exp(1.7\cdot(22.36-3)} = 5.18\cdot 10^{-15}$$

$$t(remaining\ time\ high) = \frac{1}{1+\exp(-0.4\cdot(12.024-4)} = 0.96$$

minimum: 5.08 E-15

Rule 6: place (10,20)

$$t(reward\ low) = \frac{1}{1+\exp(15\cdot(0-0.5)} = 0.99$$

$$t(distance\ high) = \frac{1}{1+\exp(-1.7\cdot(22.36-3)} = 0.99$$

$$t(remaining\ time\ low) = \frac{1}{1+\exp(0.4\cdot(12.024-1)} = 0.01$$

minimum: 0.01

Utility:

$$U = \frac{100\cdot 5.08\cdot 10^{-15}\cdot 30\cdot 0.5 + 0\cdot 0.01\cdot 30\cdot 0.5}{5.08\cdot 10^{-15}\cdot 30\cdot 0.5 + 0.01\cdot 30\cdot 0.5} = 5.08\cdot 10^{-11}$$

The last alternative ("work" at (20,0)) will produce the same utility as "work at (0,20) as all parameters are the same. This means that the utilities of all alternatives are

99.94, ("sleep" at (0,0))
1.21, ("sleep" at (0,10))
2.81 E-09, ("work" at (10,10))
5.95 E-05, ("work" at (0,20))
5.08 E-11, ("work" at (10,20))

2.81 E-09, ("work" at (20,0))

Out of these utilities the maximum is chosen. The agent will "sleep" in its home place (0,0), and its "tiredness budget" will be credited with one unit of sleep. In the next hour the evaluation procedure is repeated, with the remaining time for "sleep" reduced by one hour, which decreases the utility of that activity. Note that the budgets are only evaluated once every model day, while the decision where to go and what to do takes place on an hourly basis, so that immediate success in performing activities shows only at the beginning of each day, and has no influence on the short term behaviour.

6.4 Environment

The environment consists spatially of nine points representing a grid of three by three cells. This can be varied between one cell (non-spatial problem) to up to ten by ten points on an orthogonal grid. Except for its position on the grid each cell / point has the following attributes:

- A *capacity* for four activities
- A *price* for four activities
- A specific time dependant *payoff* for four activities

The model can be run with variable or fixed prices and/or capacities, whilst the specific payoff is fixed over the time of a simulation run.

6.4.1 Prices

The price of an activity determines how many money units per model hour are subtracted from the money budget of an agent carrying out the activity in question. Prices are effective for the activities shopping and socialising as it would make little sense to "charge" for work. Recreation is free as well, but here it could be justified to introduce a price as this activity is supposed to cover housing, too. However, as a measure of demand for the activities, prices are calculated and logged for all activities.

The price can either be static or adaptable in a simulation run. The adaptable setting determines the previous day's demand for the activity in the place in question averaged over all 24 hours of the model day. A logistic curve (Figure 21) is then used to calculate the new price. The logistic curve has the advantage of limiting the minimum and maximum values of the price function to a defined range. The third parameter of the logistic curve, its slope, determines how quickly the price is changed. A further parameter sets the point where the curve reaches 50% of its maximum value.

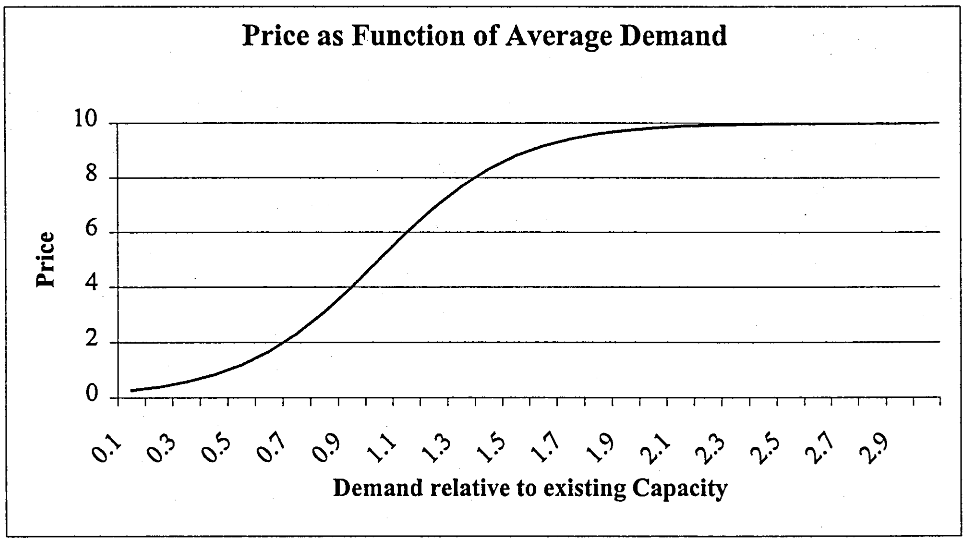

Figure 21: Price for an Activity as a Function of Previous Demand

Although the adaptation function is non-linear, it is deterministic and the same in all places. The agents are able to change the environment by choosing where to carry out activities, but the nature of the environment's response is not likely to lead to effects of self-organisation in the spatial structure.

6.4.2 Capacities

Capacities are introduced to limit the number of agents carrying out an activity in the same place at the same time. This means that agents have to compete for limited resources, and might have to look for different places in order to satisfy their needs.

Like the prices, the capacities can either be static or adaptable during a model run. Capacities adapt according to average demand for the activities in the last 300 model days. Like in the case of the prices a logistic curve is used to calculate the new capacity value. A maximum capacity limits the density in each place.

The exception is the capacity for work, which is calculated as the sum of the square roots of the capacity for shopping and socialising multiplied with a correction factor. This reflects that work is regarded as an activity, which is derived from the demand for shopping and socialising. The use of the square root of the capacities is an attempt to incorporate "economies of scale" into the adaptation process.

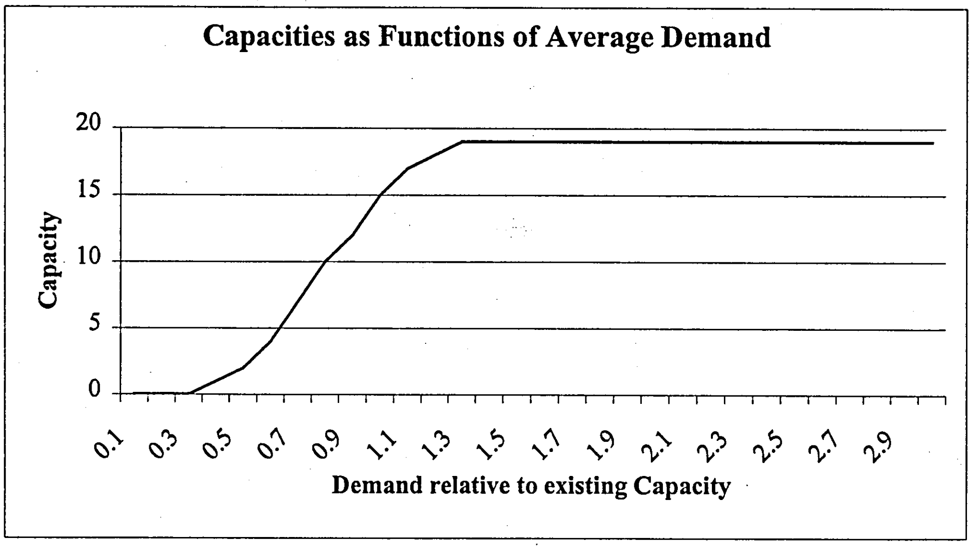

Figure 22: Capacity for Recreation, Shopping and Socialising as a Function of Previous Demand

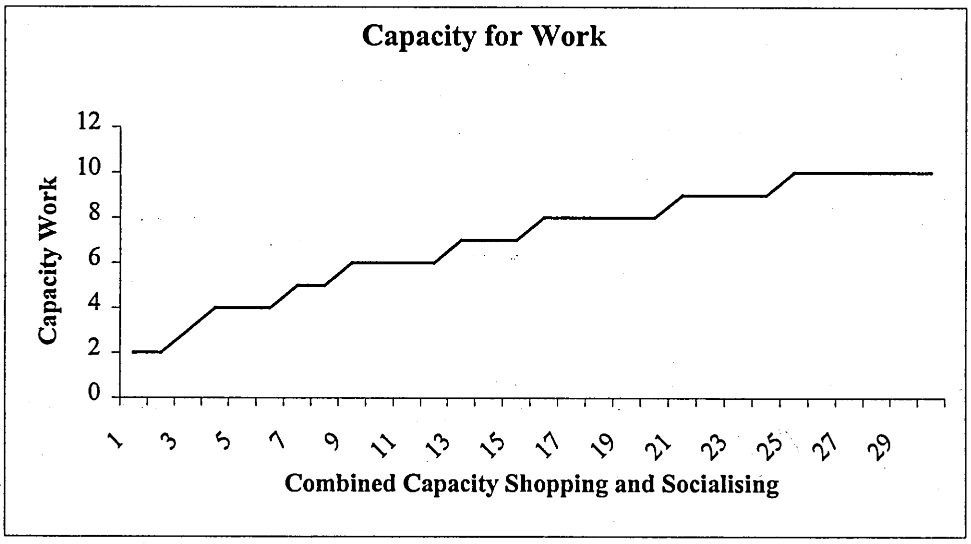

Figure 23: Capacity for Work as a Function of the Capacities for Shopping and Socialising

6.4.3 Payoffs for Activities

Carrying out an activity leads to a payoff for the agents as long as the capacity for the activity is not exceeded. As a second constraint, a payoff for shopping and socialising is awarded only if at least a second agent is present to work. Although in reality one agent would not be able to serve other agents for both shopping and socialising at the same time, it is the minimum constraint assumed.

All payoffs are time-dependent in as far as only certain hours of the model day qualify to carry out activities successfully. These time periods are fixed over the duration of a model run and are aimed to reflect effects like day and night for recreation, shop and entertainment venue opening hours, and as a result of these, work time.

The payoffs themselves are linear in the case of working and shopping if agents choose to perform these activities successively for more than on hour. The values for the payoffs are pre-set for shopping, but in the case of work, a turnover for that place is calculated adding up the total amount of money spent the previous day by the agents. A fraction of this amount is reallocated as wages for the next day.

In the case of the other two activities, a different approach was taken. The payoffs here are not only time dependent, but the time already spent on these activities has an influence on the total payoff, thereby taking into account that physiological processes like recreation or sleep have different effects depending on how long the activity has been carried out without interruption. This non-linear relation for recreation and socialising is captured by a curtailed bell-shaped curve (Figure 24)

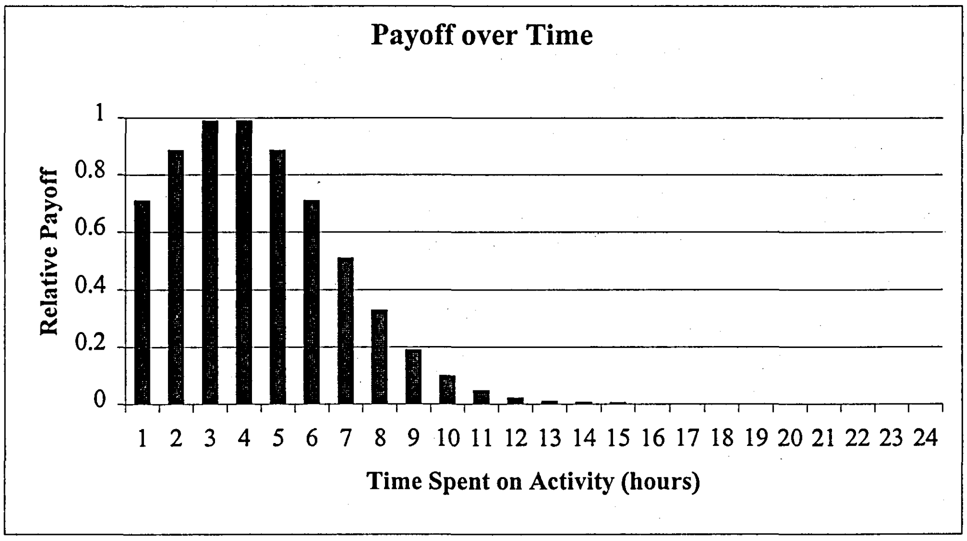

Figure 24: Payoffs for Recreation and Socialising as a Function of Successive Choice of the same Activity

The payoff functions somewhat simplify the conceptual framework insofar as that the time rules of the artificial society should be a result of the interactions of the agents with the environment, as much as they are enabling and limiting factor at the same time for the agents, thereby determining their success. This option was not followed, because the emphasis of the model was to be on the adaptation processes within the agents. A feedback into the environment (as desirable as it is from the methodological side) would make the already quite complex processes within the model even more difficult to track, so that the analysis of the model behaviour would probably hindered by the complexity of the model.

6.5 Implementation

As a PC platform was readily available the model is programmed in Microsoft Visual Basic™ version 3.0. This is an easy to use programming environment, which facilitates especially graphics output and data storage in database format. For this reason all results are written in Microsoft Access™ version 1.0 format. The database format is advantageous for easy processing of the results, but has its downside on processing speed and storage space used.

However, the programming language is not ideal for this kind of model, and for this reason the maximum number of agents in the model is limited to 64. The number of agents can be varied in multiples of four due to the restrictions imposed by the programming environment. This is a rather small number of agents, so scaling effects of large populations cannot be observed in this implementation, but it is possible to run a large variety of different configurations. Using the same initial conditions it is possible to compare different model set-ups, effects of random number generation and subsequent pathways taken by the model.

7 Results

In this section the results obtained from the model described in Chapter 6 will be outlined. The focus of the investigation is on the evolutionary properties of the model. Is it possible to generate agents, which are adaptive according to the conceptual framework (Chapter 5), and at the same time lead to credible behaviour based on their intrinsic needs? This would qualify the modelling approach for the use in a larger model of urban development as outlined in Section 2.4.

The nature of the model leads to a huge parameter space in which the model can position itself. This makes the interpretation of results difficult, as the amount result data is demanding on the ability to analyse it. Furthermore, the nature of the data generated is usually not normally distributed which restricts the applicability of standard quantitative statistical analysis to non-parametric tests. Therefore the data analysis will as far as possible be restricted to a qualitative evaluation in order to test the validity of the modelling approach at least on a qualitative level.

7.1 Run Configuration

As the central question of the investigation is whether it is possible to design adaptive agents which are on their own capable of finding sensible behavioural rules according to which they can conduct an everyday life, all spatial aspects have been eliminated at an early stage of the investigation. This means that a number of features that the model is capable of are not used. The environment is reduced to only one cell / point, which practically disables the spatial features of the built-in cognitive map. Still, the agents make use of the cognitive map feature in order to log their past success in carrying out activities, only that there is only one alternative for each activity.

For reasons of practicality the number of agents has been set to 20 for most runs which still results in runs of 18 hours duration for a run of 5000 model days. In order to test the effects of larger numbers of agents, the maximum number of 64 agents has been tested in runs of 3000 model days, but no significant differences have been detected. For larger numbers of agents (or longer runs) the resulting data files would become too large for the computer platform used and the possibility to obtain multiple runs would have been severely restricted.

Although the model set-up allows for the settings of rule parameters to be changed together with the behavioural rules, it was not deemed practical to mutate both rules and rule parameters (which can be interpreted as sensitivities to input parameters). In the case of changing both parameters and rules it is not obvious which of the two has finally led to an agent becoming successful or staying unsuccessful. Besides this, the search space for the agents, which have to find sixteen good rules in order to become successful, would be enlarged by two parameters per input and output. As one rule consists of eight inputs (four budgets @ two Fuzzy Sets *high* and *low*) and one output set, this amounts to an additional 128 parameters to be defined. This is clearly impracticable in a first step which tests only how sensible rules can be generated from within the model. For this reason, all Fuzzy Sets on the input side have been set to standard logistic curves.

7.2 Definition of Performance Measures

Before we move on to describe the data generated with the model, it is necessary to define performance measures, which reflect the aims of the modelling exercise. The conceptual framework demands that the results are to be interpreted in an evolutionary (in the definition of Varela, Thompson and Rosch, 1991) context. This puts some doubts on the usability of traditional materialistic interpretations of performance measures, such as the accumulation of material wealth. However, it has been said that the budgets measure the satisfaction of the agent's needs, but the definition of cultural evolution as an "ongoing process of satisficing" (Varela, Thompson and Rosch, 1991) implies that once a minimum budget state is reached, a state of success is obtained, which does not need to be improved further. Furthermore, the rise of one budget can in fact come at a cost for issues of longevity and therefore the sustainability of that particular lifestyle.

For the same reasons also aggregate measures such as the sum of all budgets of an agent are not considered. The simple sum of budgets would mean that the performance measure amounts to basically the same additive utility function, which has already been criticised in Chapter 2.3.6. This formulation can cover deficiencies in some budgets with over-performance in others, so that the very idea underlying the multicriteria approach would be contradicted by the data analysis.

In the case of the performance of the system as a whole, additive performance indicators cannot be used in a meaningful way either. Any additive measures such as the sum of the sums of all agent's budgets will - as in the individual case - obscure deficiencies which are thought to be crucial, because the aim of the agents is to satisfy all of their needs, not to accumulate as much wealth as possible. In the perspective of the artificial community, all agents should be successful in the sense of the concept of satisficing.

It can be concluded that as the budget states do not provide us directly with a system performance measure, the budget states can amount to an indirect performance measure. Because unsuccessful agents are not removed from the system, but reset if one of the budgets reaches a defined threshold (-500 units), the time between resets can serve as probably the best performance measure which can be found in this case. In practice this amounts to the agents' age, but as the agents are reset and not removed and reintroduced, the time between resets is not equal to the agents' age, which will be the same as the length of the simulation run. The distribution of the time between resets for the whole of the population as well as the average can give indication how well individual agents and the entire the system are doing.

Longevity gives us a measure of how well the agents are adapted to their environment. In the case of an evolutionary system, this environment would be created to some extent by the agents themselves. In the case of a static environment longevity is a measure how fit the agents are in the Neo-Darwinist sense, and this amounts in our case to testing whether the agents are doing what could be expected from them in the context of a daily life in an urban environment. If the agents fail to succeed in the static environment, which is defined in a way that resembles a real city, it must be suspected that the model underlying their behaviour is not valid, whilst the conceptual framework would be validated if the agents behave successfully. However, it cannot be expected that all agents are successful from the outset, because they have to accumulate the necessary knowledge, which enables them to survive.

7.3 Nature of Data Obtained

Having defined the performance measures against which the generated data will be matched, some basic properties of the data to be analysed have to be clarified. When speaking of successful agents according to the performance measures defined above, we will refer to agents which keep all of their budgets above the reset threshold for the whole duration of a simulation run of 5000 (3000) time steps (days), or at least for a great part of the run. An example for such an agent is given in Figure 25.

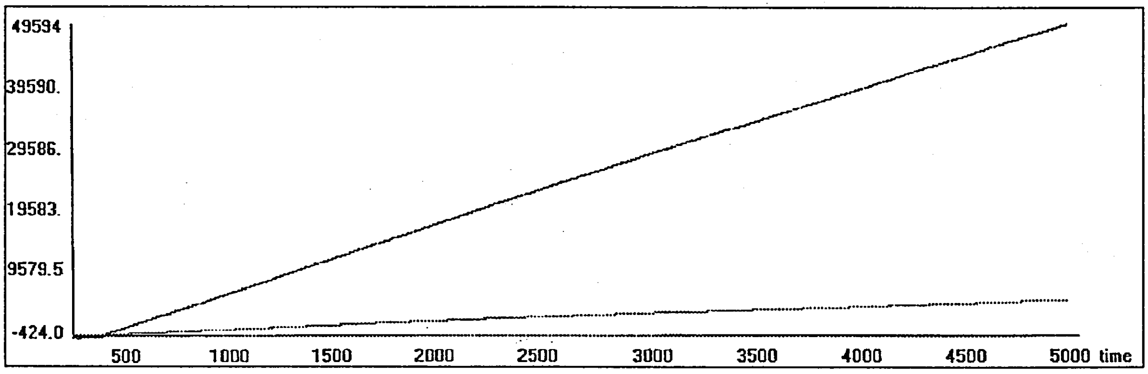

Figure 25: Budget Graph of a Successful Agent

In the case of an unsuccessful agent, we observe frequent resets due to the inadequate behavioural rule set. The budget graph for such an agent is shown in Figure 26. Finally it is possible that an initially unsuccessful agent discovers a rule set leading to successful behaviour due to one of the modification mechanisms described in Section 6.3.5. This is shown in Figure 27.

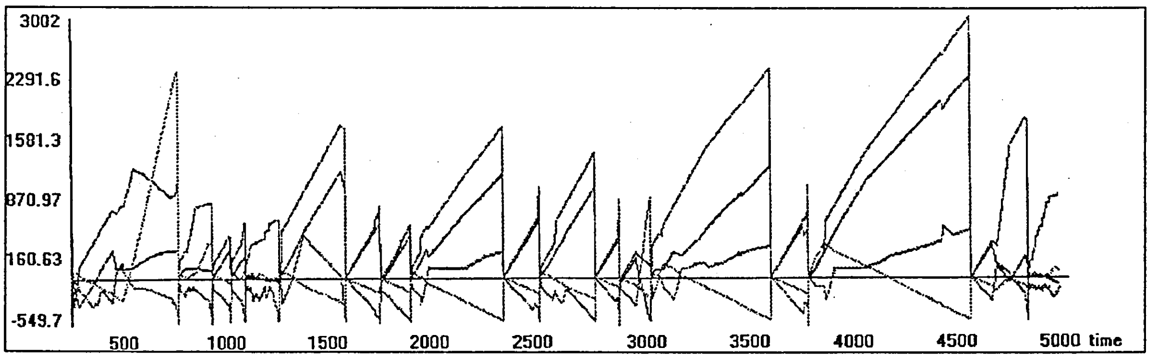

Figure 26: Budget Graph of an Unsuccessful Agent

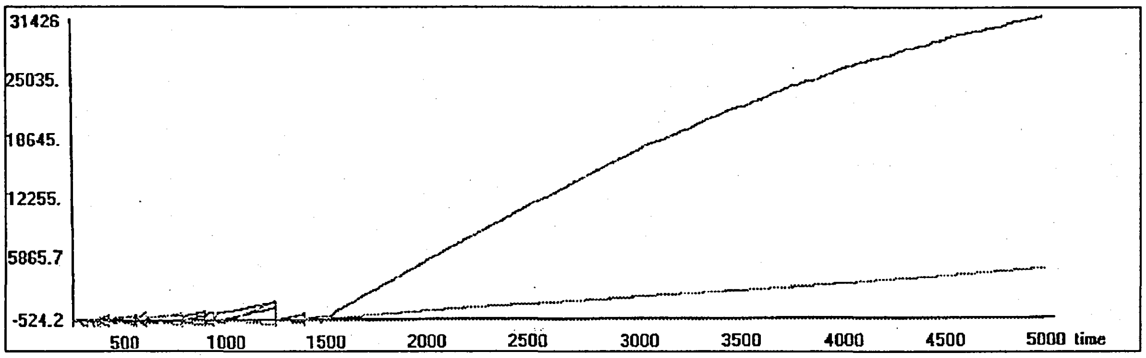

Figure 27: Unsuccessful Agent Finding a Good Rule Set During Run

As the performance of a given run is measured by the ability of agents to adapt to the environment measured by the time between resets, the relationship between the number of resets in a given run with the average time between resets of the population has to be discussed. In the ideal case, no agent is reset, in which case the average age is duration of the run. In the worst case all agents are reset at every time step, and results in a run with 20 agents over 5000 time steps in 100000 resets. The relationship between the number of resets and the average age of the population is given as

$$av.\,time\ between\ resets = \frac{number\ of\ agents \cdot run\ length}{number\ of\ resets}$$

Here we see that in the best case (no resets) the average using the above formula would give us infinity instead of the length of the run. This can be overcome counting the end of the run as a "reset" as well. This is deemed acceptable, because we are dealing with a finite run length, and in doing so we revert from counting the actual number of resets into the number of tries to find a sensible rule set for the agents. The relationship between the average time between resets and the number of tries for 20 agents in a run of 5000 time steps is shown in Figure 28. It has to be noted that the relationship is non-linear, which results in a very rapid decline in the average age once only a small number of agents are unsuccessful. To give an example, in the hypothetical case that 19 out of 20 agents are not reset, but one agent is reset on average every twenty time steps, the resulting average time between resets decreases from 5000 for all 20 agents to 400! The significance of the average time between resets is further reduced when it is considered that an average of 400 time steps could as well mean that all agents are on average reset every 400 time steps. In the strict sense this would mean that none of the agents manages to find a really successful rule set. It could as well mean that 95% of the agents are acting successfully, so that the decisive measure is the number of agents, which are not reset over the duration of a run.

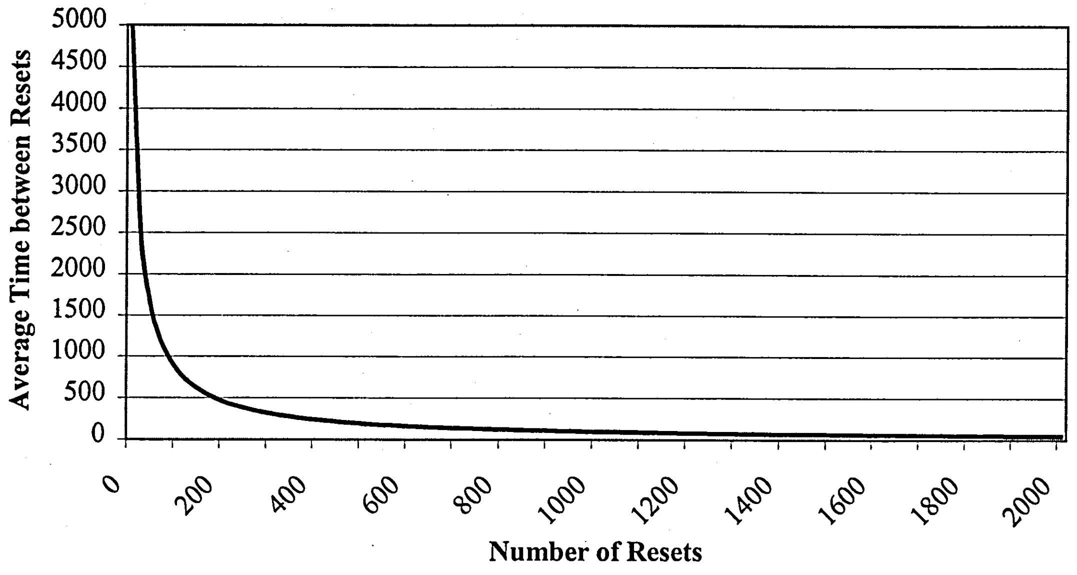

Figure 28: Relation between Number of Resets and Average Time between Resets for 20 Agents and Runs of 5000 Time Steps

In order to gain a first impression of the tendency of the average reset time the time between resets has been logged in intervals of 500 time steps (Figure 29). Even as the average age is not a significant description of the actual processes taking place during the run (see above), this measure can provide information on whether previously successful agents have been reset at some point, thereby lowering the average. Unfortunately, it has been observed that while only a few agents make the jump to good rules, short-lived ones tend to find "worse" rule sets over time, which lead to even shorter times, so that the average stays approximately the same.

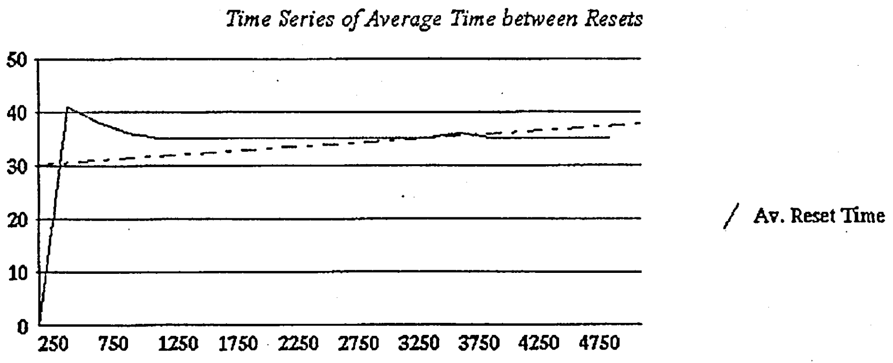

Figure 29: Example Average Time between Resets over 500 Time Steps

Because of the non-linear relation between the number of agent resets and the average time between resets it is of particular interest to get information on the distribution of reset times. This can compliment the figures given by the average reset time. The distributions have been logged in categories of 50 time steps, with only one category covering all events over 450 time steps. The experiments have shown that the resulting distribution is highly skewed towards low values, so that the frequency of unsuccessful behaviours is much more often observed than successful behaviours (Figure 30).

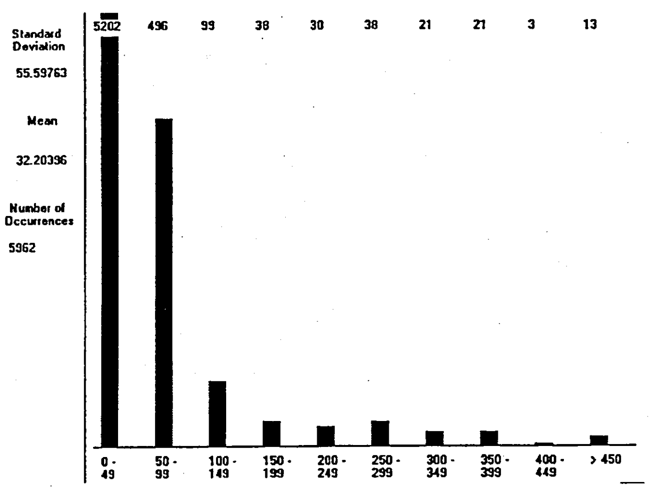

Figure 30: Example Distribution of Reset Times

In order to account for the value (in the sense of the performance measures) of longevity a weighted distribution of reset times has been measured alongside with the simple frequency distribution. Here the frequencies are weighted with their actual values. This reflects much more clearly the effect of successful behaviour. As the total of the weighted values always equals the number of agents multiplied with the duration of the model run (100000 agent-time steps in the case of 20 agents and 5000 time steps, respectively 192000 in the case of 64 agents and 3000 time steps), we would observe in a learning system a shift from the lower categories towards the higher ones, especially into the highest one.

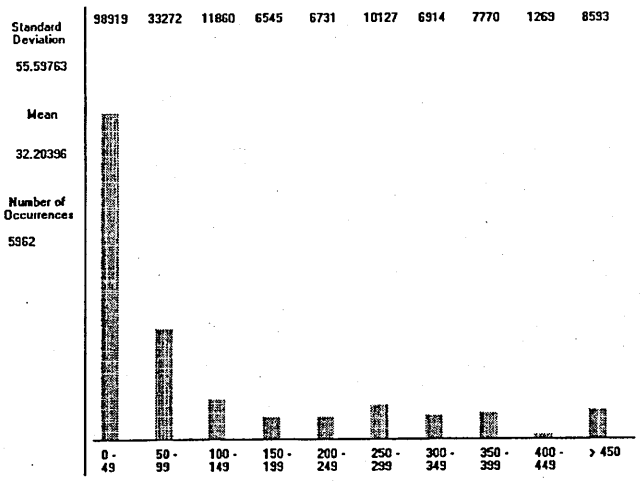

Figure 31: Example Weighted Distribution of Reset Times

Finally it is useful in the case of learning systems to observe what strategy the agents take up and how successful they are using these strategies. The distribution of learning strategies (Figure 31) gives a first impression of the ecology of learning strategies, which will be used in the case of the evolutionary system.

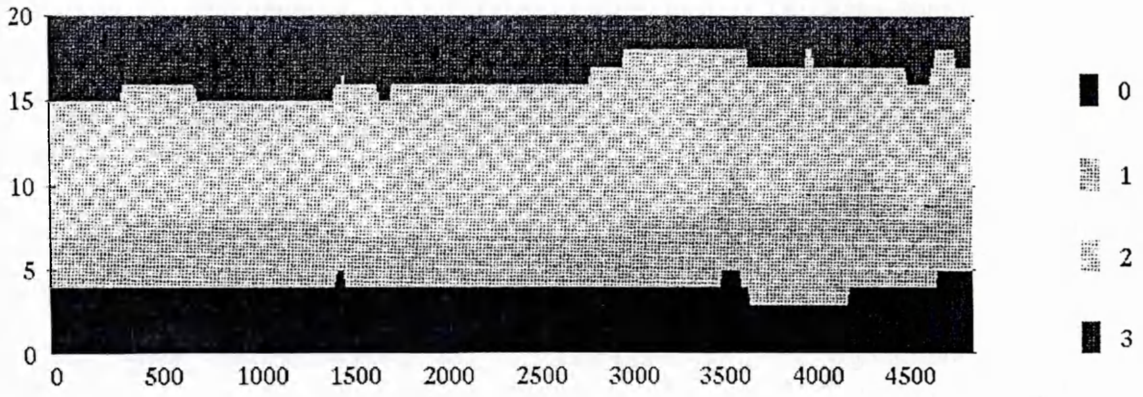

Figure 32: Example Distribution of Learning Strategies (Strategy 0: Non-adaptive; 1: Random Mutation of all Rules; 2: Random Mutation of Rules for Low Budgets; 3: Correlation)

It has been shown above that the data generated does unfortunately not possess any of the qualities needed to conduct a quantitative analysis. For a quantitative analysis the data would has to be normally distributed in order to carry out the standard statistical tests. Furthermore, the sample size (which had been chosen for computational reasons) is not large enough to for any features to be detected with statistical significance. On the other hand we are dealing here with a complex system, which have traditionally not been open to the standard measures of statistical analysis, so that we cannot expect that these methods are applicable in our case. A method, which in theory is still open to compare different runs, is the use of non-parametric tests. These tests are used to compare differences in the shape of distributions, but as it has been outlined the sample size is limited, and more detailed distribution curves would lead to many empty categories in the distribution histogram so that the validity of such a test would be questionable anyway. On top of this, we are interested in the dynamics of the processes taking place during model runs, so that any distribution documenting the end state as a summary of what has happened is of no great interest.

7.4 Reference Case

As a first step into the validation of the model a reference case was defined. The crucial feature of our model is that it is adaptive in the sense that it can generate new rule sets by connecting parameters in a new way. Conventional rule-based systems on the other hand apply the rules, which have been defined from the outset. In addition to this, the model has the possibility of assigning different adaptation strategies to the agents, which takes the idea of adaptation one step further than earlier adaptive models using genetic algorithms or neural nets. The latter techniques modify the rules in the system with the same algorithm, whilst in our case we have the choice between three different ones. The most interesting case, however, is to test whether a non-adaptive population of agents would take up adaptive techniques and use adaptation to their advantage by being longer-lived than the non-adaptive ones.

The reference case is used to establish the characteristics of a conventional rule based system set up like our model. This means that no rules change during run time and unsuccessful agents are reset to the same rule set as before. The agents are not allowed to take up adaptation, therefore having perfect reproduction.

For the reference case two configurations have been investigated: random rules and "sensible" rules. The random rules are the same as for all other runs involving random initial conditions, whilst the "sensible" rules were defined using common sense assumptions on how a set of usable rules could look like. The "sensible" rules are described in Section 7.4.1 below.

7.4.1 Pre-defined "Sensible" Rules

This configuration is emulating a rule-based system using heuristic behavioural rules. As outlined above, the rules remain the same for the entire duration of the run. The rules used have one budget as the input parameter and involve neither AND nor OR operators. They are defined as:

(1) If the recreation budget is low then recreation is 100 points important

(2) If the recreation budget is high then recreation is 0 points important

(3) If the money budget is low then work is 100 points important

(4) If the money budget is high then work is 0 points important

(5) If the goods budget is low then shopping is 100 points important

(6) If the goods budget is high then shopping is 0 points important

(7) If the socialising budget is low then socialising is 100 points important

(8) If the socialising budget is high then socialising is 0 points important

The definition of the rules is equivalent to an independent multicriteria control system. The state of the budgets relative to each other is reflected in the time allocation rule (see 6.3.3) which calculates the relative values of all importances to each other, and then accordingly allocates fractions of the 24 hours of the model day to the respective activities.

If the capacities are set to sufficient sizes it should be expected that all agents are able to act successfully, as the rules are defined in a way that in theory allows for balancing the budgets above the reset threshold. However, the main result of the runs in this configuration is that this is against all expectation not the case. Only a fraction of about 40% of agents (25 out of 64) are not reset during these runs.

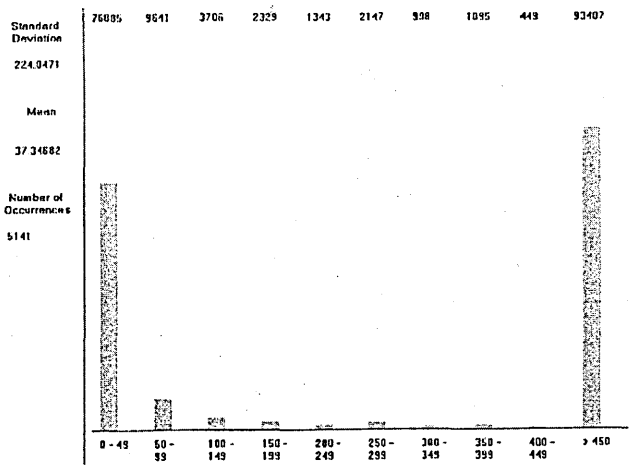

Figure 33: Weighted Distribution of Times between Resets for Reference Case – Predefined Rules

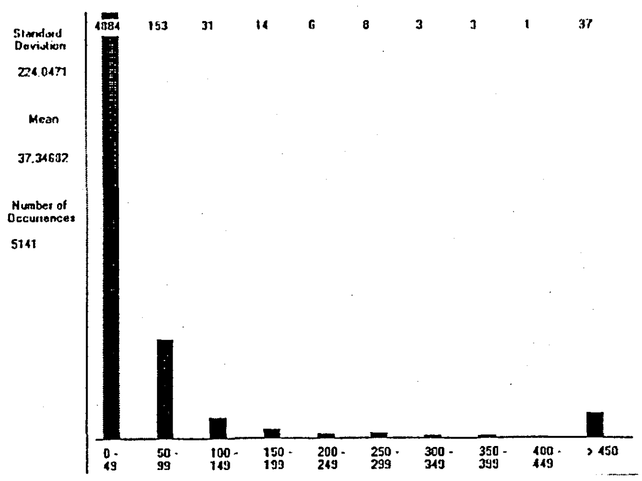

Figure 34: Distribution of Times between Resets for Reference Case – Predefined Rules

However, this observation becomes clearer when the exact procedures taking place are considered. All agents start from symmetric (identical) initial conditions, so that in the first time step all agents have the same preferences and plan to do the same things. Here the symmetry of the configuration is broken when the capacity constraints come into play. As the order in which the agents act is determined randomly, only the first agents up to the defined capacity will receive a payoff from the activity. The rest of the agents will receive nothing, and the symmetry in the budget states is broken. This will lead to different decisions by the agents from day 2.

The other effect leading to unsuccessful behaviour in a homogeneous population is generated by the constraint demanding in the case of shopping and socialising that at least one other agent has to be present working in order to make other agents receive a payoff from that particular activity. It is obvious that a homogeneous rule set for the entire population increases the probability that all (or at least many) agents decide to pick the same activity at the same time, which does not lead to the cooperative pattern needed in this case. However, this is probably a more latent than acute danger, because the proportion of successful agents shows that cooperative patterns come into existence at least to some degree.

For the dynamic environment it can be said that the capacities appear to find an equilibrium point in the course of the run. This equilibrium is far lower than expected, as capacities for recreation rarely pass 16 (in a population of 64) whilst socialising is not exceeding 5. Even for the 25 successful agents this appears to be a very low value, and it has to be taken into account that also the remaining 39 unsuccessful agents generate demand. It has been taken into consideration that in this case the response curve for capacity adaptation might be too restrictive, thereby limiting the maximum number of successful agents by imposing a very low carrying capacity. On the other hand very similar behaviour has been observed in other runs with even more generous capacity constraints (in the extreme case with a static environment with capacities set at the number of agents).

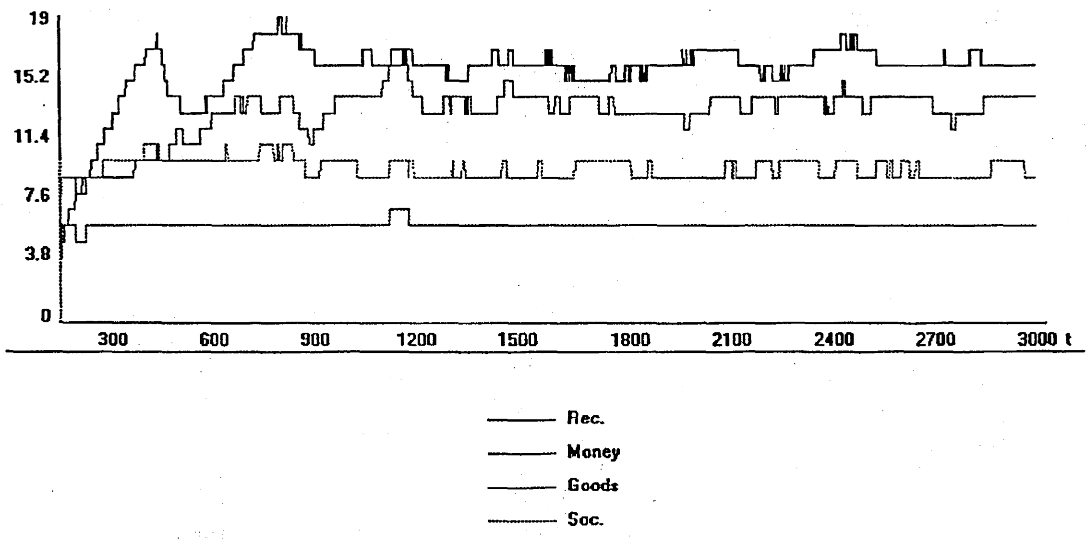

Figure 35: Capacities for Activities for the Reference Run with 64 Agents

The most interesting case of this run configuration is how some previously unsuccessful agents become successful over time. This cannot be due to finding a better rule, because the rules do not change over time. Still, this phenomenon has been observed, and it seems to actually reconfirm the conceptual framework. The discovery of viable niches in the system is here due to the mutual interdependency of agents and environment. If due to high prices and insufficient capacity an agent is reset, it is relaunched after clearing its debts. It might now be a matter of chance for the agent to be drawn early in the allocation process to be able to accumulate a safe budget state, which can be used in times of bad luck. On the other hand all other agents change the landscape constantly during the run, so that the capacities might have increased slightly, in which case it is simpler for a previously unsuccessful agent to act successfully. This interdependency works of course in the opposite direction as well, so that previously successful agents might - for instance through a period of bad luck - be reset. If at the same time the overall behavioural patterns in the population change slightly, there is a risk of reduced capacity, which slims the chances of finding a viable niche in the system.

7.4.2 Random Initial Conditions

In order to be able to estimate the effect of rule mutation of the system, the random initial conditions used throughout all later simulation runs are tested on the static system as well. With this data it was then possible to establish the differences between the model configurations. As it might be expected from the size of the search space in which rules can be defined, the performance of this configuration was rather low. However it turned out that these precise initial conditions lead to one agent (out of 20) behaving successfully, which is a surprising fact already. For the rest of the population, the distribution of times between resets is spread very similarly to the ones obtained from the early stages of adaptive systems.

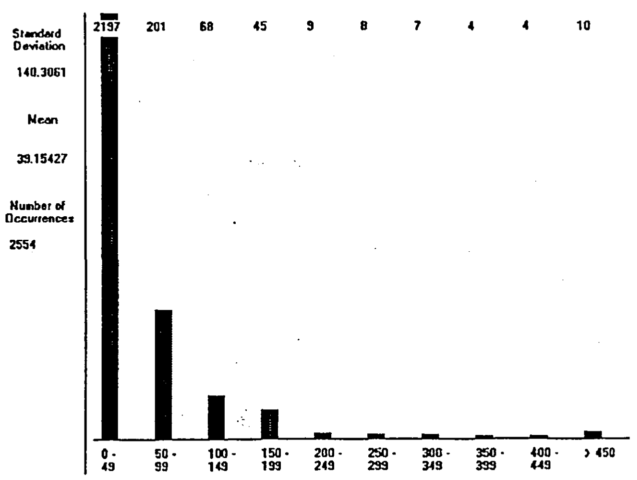

Figure 36: Distribution of Reset Times for the Reference Case – Random Initial Conditions

There are a number of agents, which are reset every 100-150 time steps, which is due to the initial conditions. As opposed to the adaptive runs, these events persist over the entire duration of the run, whereas in the in the adaptive version, unsuccessful agents tend to reset times around 25 time steps after some time. This can be taken as an indication that the rules leading to reset times above this value are already a very rare configuration.

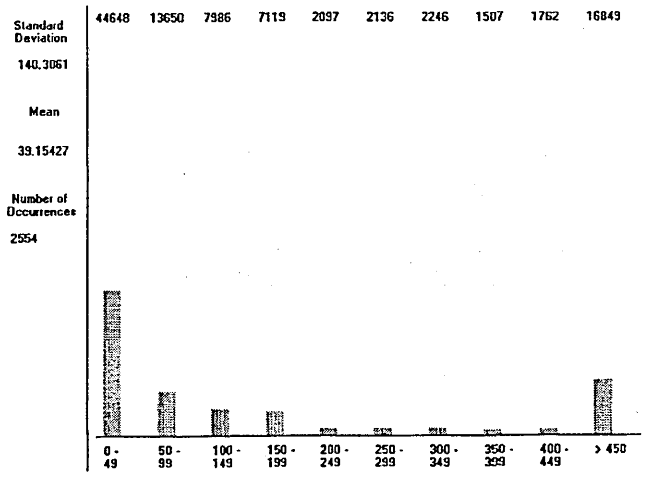

Figure 37: Distribution of Reset Times for the Reference Case – Random Initial Conditions

The average reset time remains more or less constant over the duration of the run, which would have been expected from rules that remain the same over time. The standard deviation of the times between resets is much smaller than that of adaptive runs, which means that the reset times are more evenly distributed than in the adaptive case. Still, this distribution is very much skewed towards low reset times. In the weighted case, only 17% of all reset times (equivalent to only 0.4% of all events) are found in the highest category, whilst the lowest category contains approximately 45% (equivalent to 86% of all events) of all weighted events.

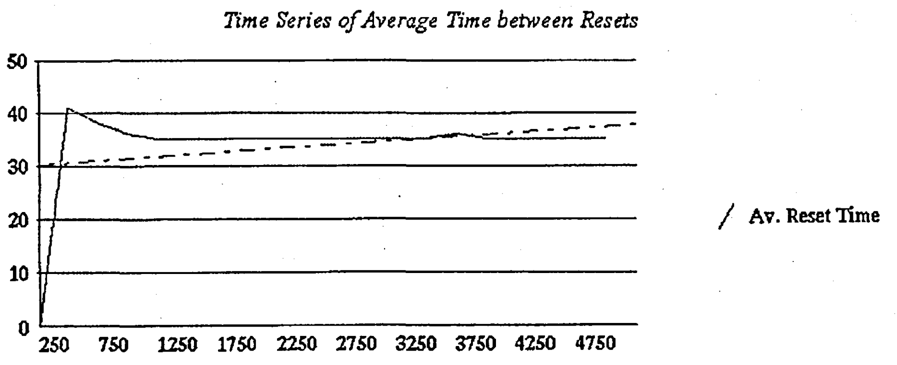

Figure 38: Average Time between Resets for Reference Case – Random Initial Conditions

In summary it can be said that this configuration only confirms what is to be expected from a static rule-based system. The rules have to be defined in a way that requires knowledge on the nature of the system from the very beginning, otherwise the rule base is not capable of governing the agents successfully. It has now to be determined whether an adaptive rule base can perform better. This means that the system has to be able to find working rules on its own.

7.5 Effect of Rule Mutation

Having established the static reference case, the three different rule mutation methods described in Section 6.3.5 were now tested on their own, before in the evolutionary set-up they were put into competition with each other and the static case. The aim of this part of the modelling exercise was to compare the different methods and to establish their efficiency. In comparison with the reference case there had to be an increase of the number of successful agents over time when started from the same initial conditions. This is not obvious from the definition of the mutation techniques, as it is possible that a mutation technique actually decreases the viability of the system by replacing bad with worse rules which lead to even shorter life times.

The basic indicator for the performance has to the number of successful agents (in the best case) respectively the average time between resets (with the objections made in Section 7.2). If a technique results in an increase of the average time between resets, it can be assumed that at least in the set-up tested, the use of this adaptation technique results in a system performing superior to the reference case.

As the principle of optimisation has been rejected in the conceptual framework, the adaptation techniques had to be unbiased in respect to directed search as well. For this reason a random technique which (as already outlined in Section 6.3.5) is easy to implement as well and a correlation technique have been used. Although the correlation technique comes very near to directed optimising search, it has been used as a comparative measure.

The main results of this exercise are that all methods appear to work to some extent. During the runs more and more agents find good rules. However, it cannot be determined at this point whether there are crucial differences between the techniques. From the conceptual point view this is not a significant shortcoming, because the aim of using different adaptation techniques is to introduce a diversity of methods into the system and not to optimise its performance.

The techniques were tested in runs of intervals of 3000 time steps using 64 agents. The main points are summarised below.

7.5.1 Random Mutation of all Rules

The random change of all rules results in a substantial increase in the average lifetime compared to the reference case with random initial conditions. This is actually surprising, not so much in respect to the fact that the random change of rules works, but for the fact that the search space is extremely large, and that it would appear to take some considerable effort to find a complete set of working rules. After the first 3000 time steps the sum of reset times in the highest category (>450 time steps, see Figure 39) is about 3.5 times the sum of the reference case (Figure 33). This can be taken as evidence that the change in the rules of the agents improves the viability of at least a limited number of agents. The average of the category is 913 time steps as opposed to only 575 time steps in the reference case.

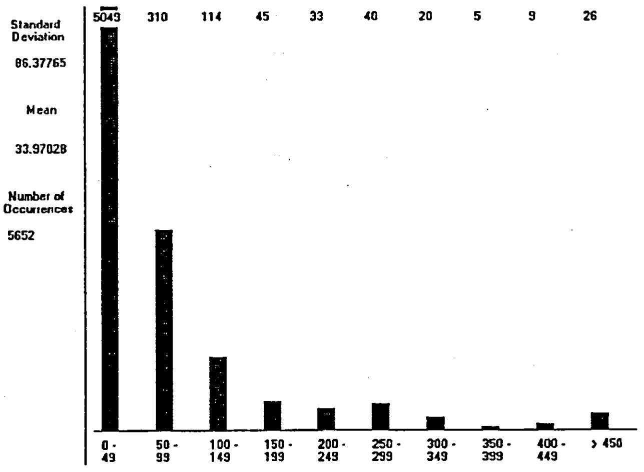

Figure 39: Distribution of Reset Times for Random Mutation after 3000 Time Steps

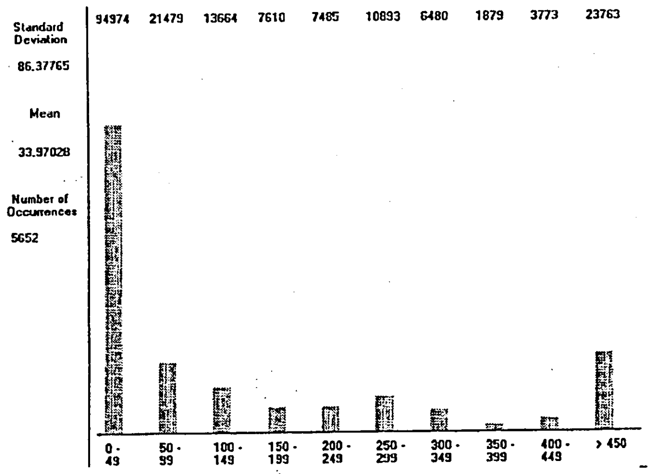

Figure 40: Weighted Distribution of Reset Times for Random Mutation after 3000 Time Steps

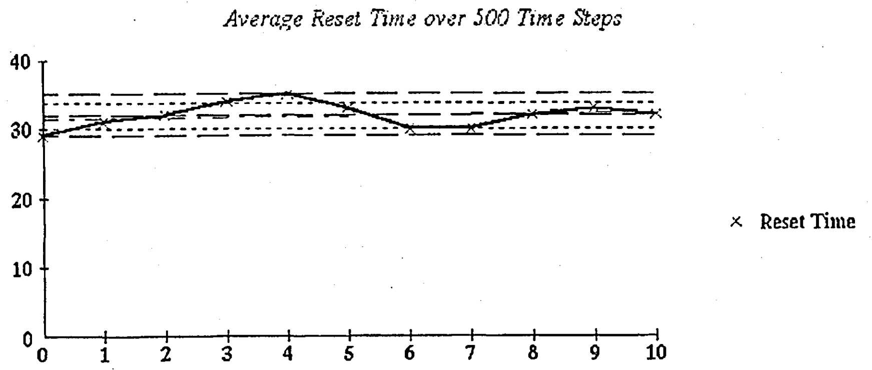

Figure 41: Average Time between Resets for Random Mutation over Intervals of 500 Time Steps after 3000 Time Steps

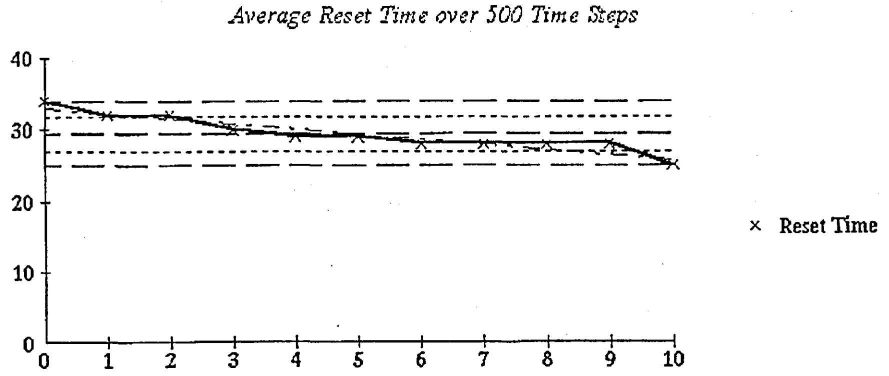

Figure 42: Average Time between Resets for Random Mutation over Intervals of 500 Time Steps after 9000 Time Steps

The tendency of the average lifetime of the agents (Figure 42) is not clear, similar to what has been observed in other run configurations. A rise in the first half of the run is followed by a fall later on. This becomes more clear in the continuation of the run (Figures 43 and 44). It can be observed that the middle categories of the sum of reset times become less and less present over time, whilst the extreme categories (0-49 and >450) are constant respectively rising. It can be concluded that the average time between reset stays more or less constant whilst the population becomes split between very bad and very good performers. The ratio of events between the extreme categories is skewed extremely towards the lower end, so that rare events of improvement are not reflected by an increase in the average.

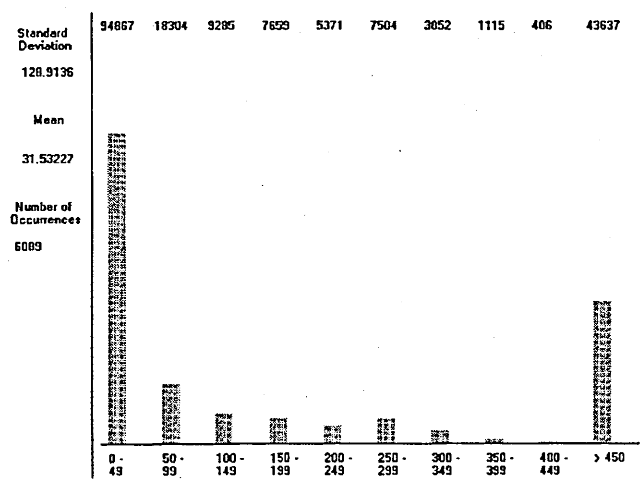

Figure 43: Weighted Distribution of Reset Times for Random Mutation after 9000 Time Steps

The last stage of this run (after 9000 time steps) even has a lower average time between resets than the first stage, but the best category's mean has risen from 913 (with 26 occurrences) to 1504 time steps based on 29 occurrences. The successful agents have become even more successful in a stagnating population.

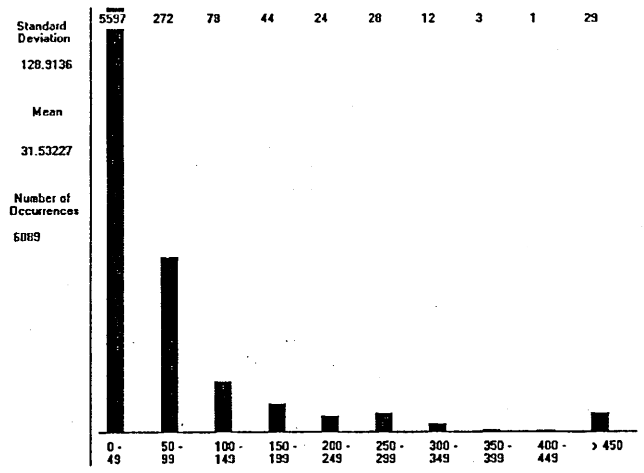

Figure 44: Distribution of Reset Times for Random Mutation after 9000 Time Steps

This is still a far cry from the results obtained with the reference configuration using the predefined "sensible" rules. Here we can observe that after the third stage in this configuration, the best category's mean is 2184 time steps based on 37 events. Although this is lower than the first leg's 2524 (with 37 events as well), it is still significantly better than the results obtained with the adaptive version.

7.5.2 Correlation between Budget State and Tendency

The benefits of the first version of the correlation technique are not as obvious as those of the random rule mutation method. The first stage of this run does not show any apparent improvements over the reference case. In the highest category of the distribution of reset times even fewer occurrences are registered than in the reference case (Figure 36). However, during the second part of the run this is partly reversed.

After two stages, the weighted number of events in the highest category has surpassed the reference case by 35%, but as this number is based on 13 occurrences the mean is only 661 times steps as opposed to 575 in the reference case (Figure 45 and Figure 46).

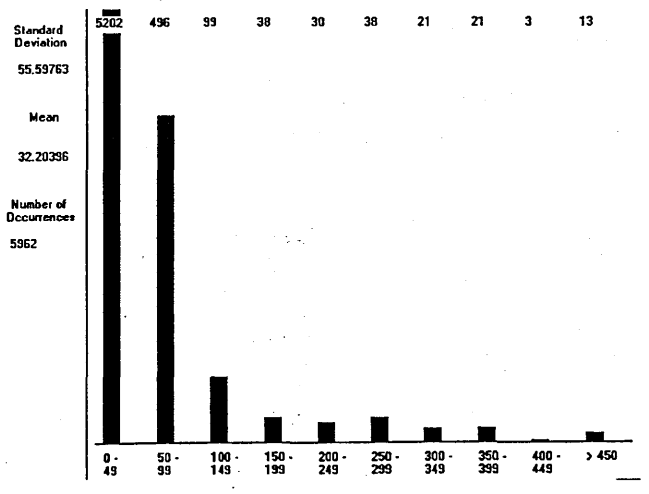

Figure 45: Distribution of Reset Times for the Correlation Technique

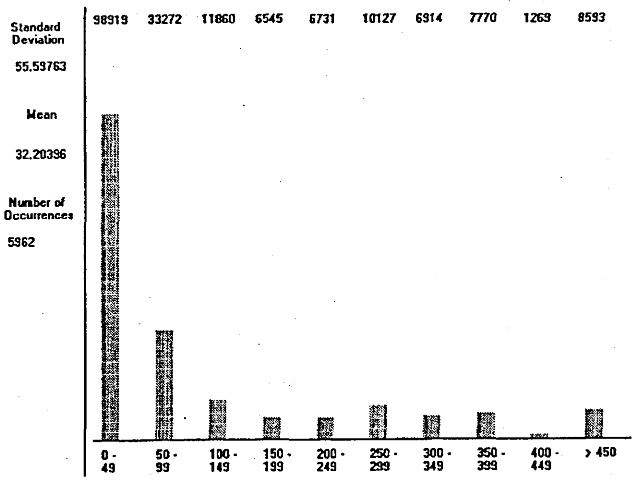

Figure 46: Weighted Reset Times for the Correlation Technique

The manipulation algorithm has subsequently been improved, and this modified version is used in all later runs, where it appears that the performance of this technique is equal to the two random techniques used in these simulations.

7.6 The Evolutionary System - "Learning to Learn"

The results obtained from the previous runs indicate that an adaptive rule based system can eventually reproduce at least some behavioural features of the observed human behaviour. In a next step diversity in the adaptation mechanisms was introduced. This puts the learning strategies in competition with each other. The performance of the non-adaptive configuration was tested as well by initially assigning this strategy to some or all agents in the system. The basic idea was that unsuccessful agents would have a possibility to change to a different strategy upon reset, so that the success of the different strategies could be compared on the basis of how many agents ended up with which strategy.

For this configuration one more strategy was added to the repertoire of the agents. It was now also possible just to get a new random rule set for the badly performing budgets. This results in four different strategies: a) not changing rules at all, b) to get a completely new random rule set, c) to get random new rules for failing budgets and d) to change rules on the basis of budget state and tendencies.

The spread of methods would give an indication of how suited the adaptation strategies would be in order to enable the agents to find adequate rule sets. The ultimate goal of this part of the simulation was to establish whether adaptation - or "learning" - would come into existence and take over from non-adapting behaviour. The term "learning" has to be regarded in a rather loose sense at this point, because the adaptation of agents only has only a very loose similarity with real-life learning of humans.

All runs start from an all non-adapting population. The chance for an agent to switch strategies is limited by introducing a relatively small probability of 1-2% of changing strategy. Adaptation strategies can in this way "diffuse" into the population. This aims at imitating "imperfect reproduction" at the reset point as agents which are reset keep their last rule set (and in the runs in described Chapter 7.5 also their strategy). The rule sets can only be changed though the use of the adaptation strategy. The aim of limiting the chance of switching strategy was to avoid that adaptation strategies are discarded at a too early point, when the strategy might take more time to find an acceptable result.

Two alternative approaches to switching have been used: The first approach uses an even probability for each strategy to be assigned, whilst the second would not reassign the strategy used before with an even probability for the other strategies to be assigned. The first set-up will on average at every one in four events assign the previously used strategy whereas the latter one will take at least two and on average three switching events to arrive at the same strategy again. This limits the possibility of diffusion back into a previously discarded strategy to a 1:300 chance on average (using a 1:100 chance to switch strategies on reset) to maintain an inefficient strategy.

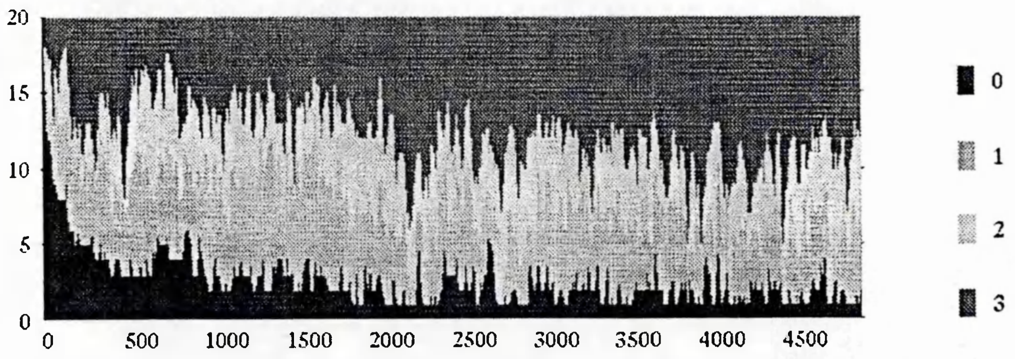

Figure 47: Distribution of Adaptation Strategies starting from an All-non-learning Population with 100% Diffusion of Adaptation Strategies (Data Series from Bottom to Top: Strategy 0: Non-adaptive; 1: Random Mutation of all Rules; 2: Random Mutation of Rules for Low Budgets; 3: Correlation)

In all runs it can be observed that the adaptive strategies take over from the non-adapting one. The speed at which agents adopt the adaptive strategies depends very much on the "diffusion rate", but the trend is common to all configurations. Figure 47 shows as an example the distribution of strategies over time of a configuration which has a 100% "diffusion rate" and Figure an equivalent configuration operating on a 1% rate. Although the fluctuations between the strategies vary considerably with the probability to switch strategy, the general trend is the same.

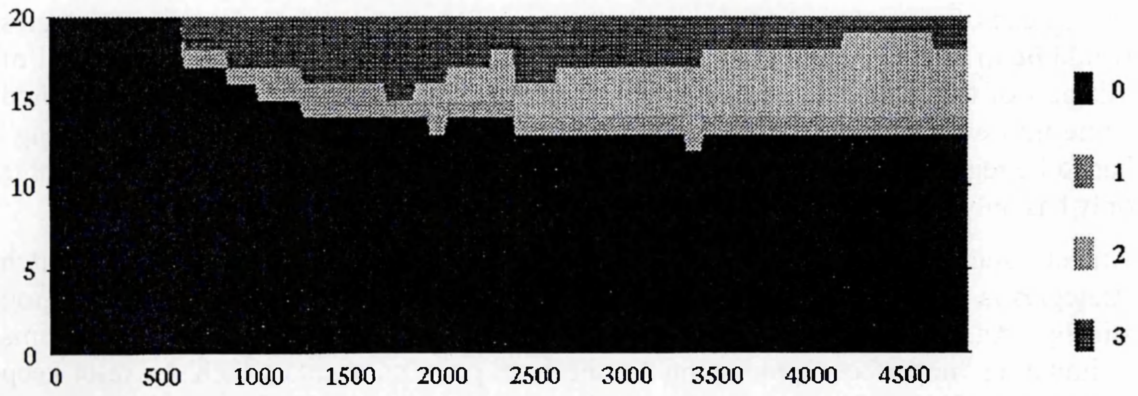

Figure 48: Distribution of Adaptation Strategies starting from an All-non-learning Population with 1% Diffusion of Adaptation Strategies (Strategy 0: Non-adaptive; 1: Random Mutation of all Rules; 2: Random Mutation of Rules for Low Budgets; 3: Correlation)

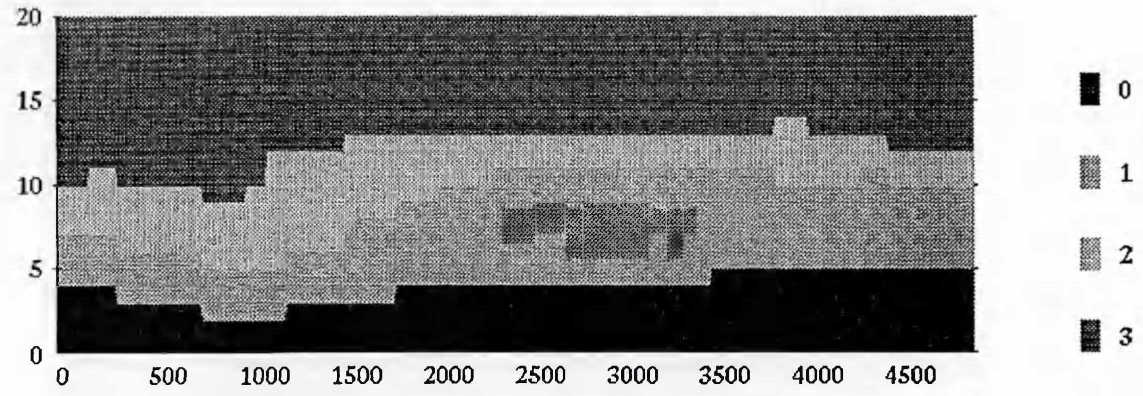

Figure 49: The Distribution for the same Run after 50000 Time Steps

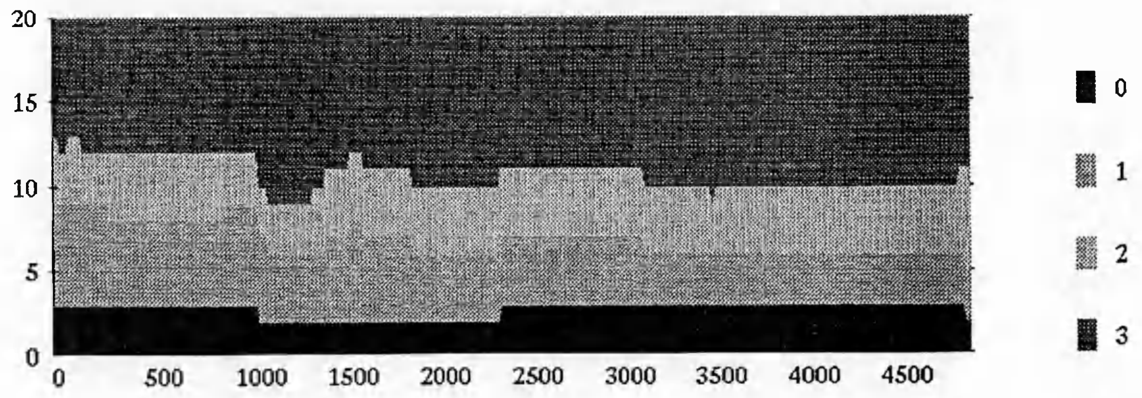

Figure 50: The Distribution after 95000 Time Steps

Age of Agents after ... Time Steps	5000 Time Steps	Adaptation Strategy	10000 Time Steps	Adaptation Strategy	15000 Time Steps	Adaptation Strategy	20000 Time Steps	Adaptation Strategy	25000 Time Steps	Adaptation Strategy	30000 Time Steps	Adaptation Strategy	35000 Time Steps	Adaptation Strategy	40000 Time Steps	Adaptation Strategy	45000 Time Steps	Adaptation Strategy	50000 Time Steps	Adaptation Strategy
Agent 1	6	3	18	3	39	1	16	0	25	0	3	2	9	1	2	2	9	3	1	1
Agent 2	53	0	110	0	9	0	57	0	113	0	23	0	1	0	14	3	20	3	1	3
Agent 3	5	0	4	2	11	0	41	2	19	2	13	0	36	3	64	3	51	3	54	3
Agent 4	61	0	96	2	13	2	162	1	46	1	164	1	551	1	102	1	118	1	115	1
Agent 5	48	2	66	2	12	1	3098	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3
Agent 6	35	0	5	2	22	1	14	1	79	1	50	2	91	2	357	2	4084	2	442	2
Agent 7	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0
Agent 8	265	0	229	0	255	0	191	0	3200	1	2182	1	227	1	1269	1	1177	1	2588	1
Agent 9	326	0	356	0	258	0	442	0	119	0	126	0	25	0	27	0	56	1	110	1
Agent 10	1166	0	590	0	723	0	1073	0	1459	0	439	0	576	0	455	0	201	0	163	0
Agent 11	8	0	11	3	6	3	4	2	6	3	13	3	12	3	10	3	11	3	7	2
Agent 12	14	1	9	2	11	3	21	3	19	2	15	1	17	2	18	2	17	3	14	0
Agent 13	26	2	8	1	37	1	2	3	10	3	14	3	13	3	26	3	17	3	21	3
Agent 14	11	2	13	1	38	1	13	3	15	3	16	3	15	3	25	2	8	2	9	0
Agent 15	5	0	6	3	26	3	10	1	20	3	9	3	15	2	14	2	5	0	7	3
Agent 16	57	0	42	0	24	3	15	0	6	0	1804	1	16	2	1	2	25	2	71	1
Agent 17	8	0	83	0	1272	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3
Agent 18	20	3	15	3	1	0	8	1	12	1	6	2	14	1	22	3	18	3	19	3
Agent 19	2091	0	2303	0	148	0	374	0	818	0	65	0	22	0	113	3	2728	3	5000	3
Agent 20	10	2	10	1	1	3	4	2	12	2	11	3	10	2	5	1	45	0	12	0

Table 10: Time since Last Reset of Agents over Previous 5000 Time Steps

Age of Agents after ... Time Steps	55000 Time Steps	Adaptation Strategy	60000 Time Steps	Adaptation Strategy	65000 Time Steps	Adaptation Strategy	70000 Time Steps	Adaptation Strategy	75000 Time Steps	Adaptation Strategy	80000 Time Steps	Adaptation Strategy	85000 Time Steps	Adaptation Strategy	90000 Time Steps	Adaptation Strategy	95000 Time Steps	Adaptation Strategy
Agent 1	1	3	3	3	12	3	13	1	3	1	10	3	16	3	2	3	11	3
Agent 2	22	3	10	2	13	3	5	3	16	1	13	1	18	1	17	1	17	3
Agent 3	34	3	64	3	31	3	24	0	40	3	41	0	23	1	10	1	16	3
Agent 4	172	1	59	1	402	1	192	1	195	1	55	1	38	1	361	1	328	1
Agent 5	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3
Agent 6	352	2	278	2	195	2	108	2	230	2	73	2	13	1	10	3	28	2
Agent 7	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0
Agent 8	1338	1	279	1	2339	1	208	1	583	1	1505	1	4514	1	3236	1	180	1
Agent 9	243	1	26	2	14	3	3	1	16	1	19	3	20	3	11	0	12	2
Agent 10	1798	0	1998	0	1369	0	191	0	1091	0	939	0	1162	0	1738	0	82	0
Agent 11	10	2	1	1	1	0	9	0	1	1	2	0	13	2	14	3	16	3
Agent 12	252	1	168	1	29	1	2855	3	5000	3	5000	3	5000	3	5000	3	5000	3
Agent 13	13	1	10	1	10	1	3	1	4	0	1	3	47	1	28	1	8	2
Agent 14	8	3	35	3	6	3	11	1	56	1	13	1	32	1	24	2	25	2
Agent 15	12	3	4	1	15	1	15	1	10	1	11	1	7	3	24	2	20	3
Agent 16	9	1	30	2	21	2	5000	2	5000	2	5000	2	5000	2	5000	2	5000	2
Agent 17	5000	3	137	3	2963	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3
Agent 18	17	1	14	2	9	1	28	0	19	2	5	2	10	1	2	2	5	1
Agent 19	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3	5000	3
Agent 20	5	2	12	3	9	1	22	1	3	3	2	1	8	3	17	1	10	2

Table 11: Time since Last Reset of Agents over Previous 5000 Time Steps (continued from Table 10)

It might now be claimed that the adoption of adapting strategies is due to the random fluctuations imposed by the stochastic set-up, but as the runs are extended to longer durations the trend continues towards a near extinction of the non-adaptive strategy. The reason why this strategy does not become completely extinct is that in every case there are agents with good enough rule sets from the outset, so that these agents naturally maintain rule set as well as strategy. Table 10 and Table 11 summarise the development of average time between resets respectively number of resets over time for an example run. In these tables, the time since the last reset is given for all agents after intervals of 5000 time steps. The corresponding adaptation strategy is indicated as well.

The presence of the non-adapting strategy depends above all on the initial conditions. If agents start with satisfactory rules they will not adopt adaptation, because they do not have to find better rules. This is the case for agent 7 in Table 10 and Table 11, which is not reset for the entire duration of the run. On the other hand it is clear that unsuccessful agents will at some point adopt an adaptive strategy, like agent 5, which starts like all agents with the non-adaptive strategy 0, then switches to strategy 3 (correlation), which leads to a successful rule set after 20000 time steps. It is possible that an agent finds a successful rule set with an adaptive strategy, but with "bad luck" is subsequently reset to the non-adaptive strategy which preserves the rule set with which the agent can now satisfy its needs. On the other hand in the case of agent 17, it can be observed that this agent finds a good rule set after 15000 time steps, but fails after 55000 time steps. This agent keeps its strategy throughout the run, and soon (after 65000 time steps) finds another successful rule set.

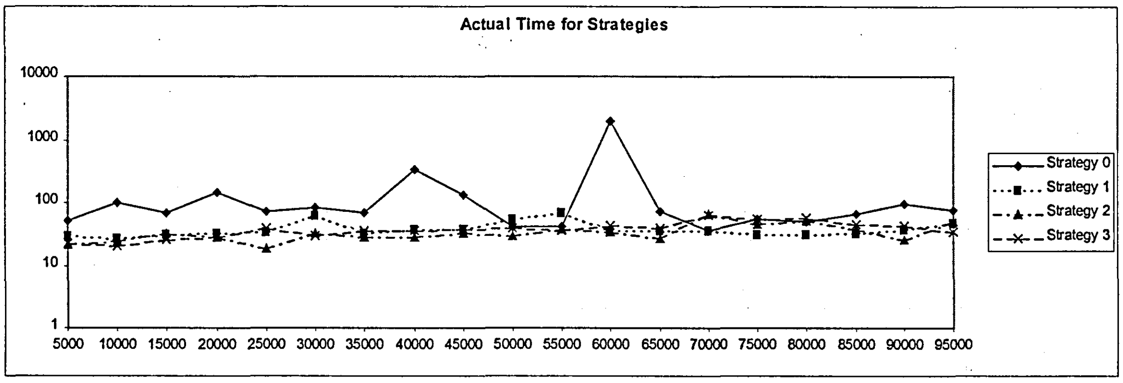

Figure 51: Actual Average Lifetimes by Strategies in Run shown in Table 10 and Table 11

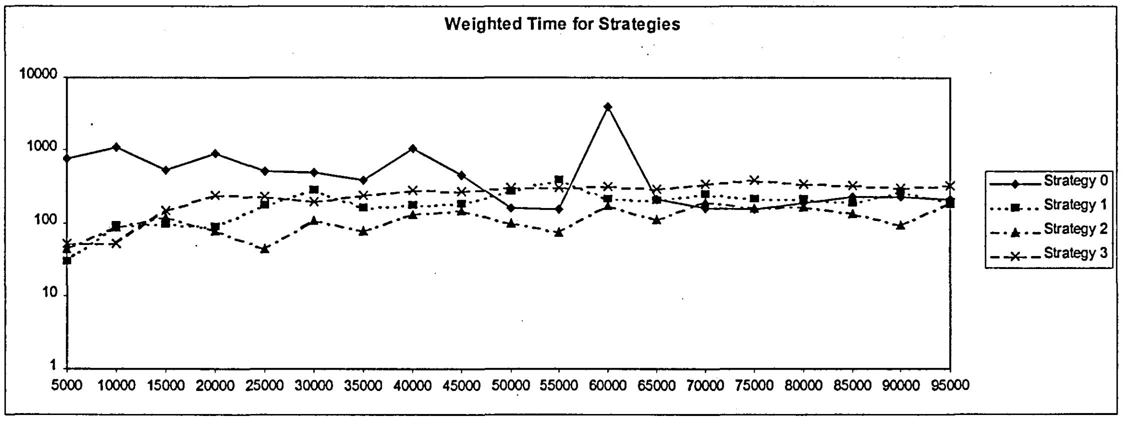

Figure 52: Average Lifetimes by Strategy weighted by Number of Agents in Strategy for Run in Table 10 and Table 11

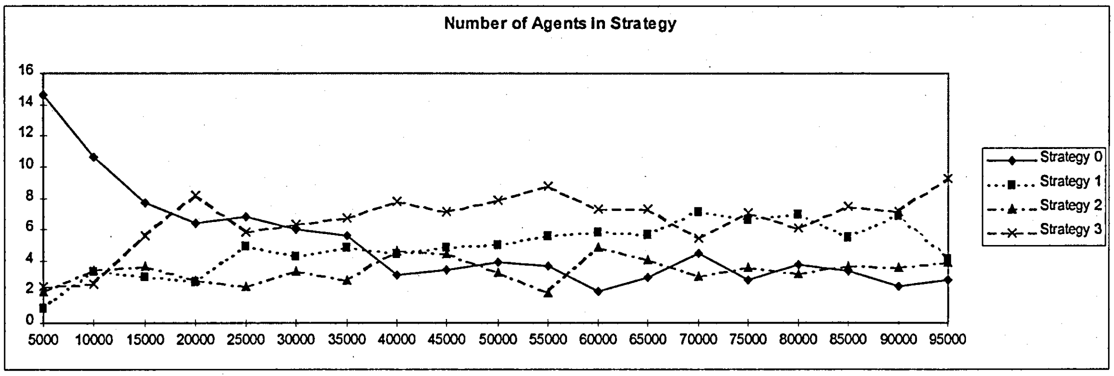

Figure 53: Average Number of Agents per Strategy for Run in Table 10 and Table 11

Although there is only a 1:100 chance that this will happen, adaptation spreads amazingly quickly. Taking into account that there are only around 2500 resets per 5000 time step run, this means that there are only 25 chances to switch strategy for the whole of the population with a 1:100 switching rate. If it is taken into account that successful agents do not change anything (neither rules nor strategy), the chances are still very slim for an agent to change its strategy. Typically about 50% of all agents have switched from non-adaptive to adaptive strategies after the first 5000 time steps when starting from an all non-adaptive population.

The question whether there are significant differences in the effectiveness of the adaptation strategies cannot be answered straightaway with the available data. The performance measure chosen - the average time between resets - is biased because of the non-linear relationship between number of resets and average age (see Section 7.3). As the absolute number of successful agents will - at least in the beginning of a run and especially in the case of a small overall population - be small, the remaining number of unsuccessful agents will depend on the size of the sub-population in question and not on the performance of the strategy. This means that small sub-populations will be favoured, if they contain long-lived individuals. On the other hand the prevalence of a strategy can give some indication on how successful it is in the Darwinian sense. Figures 51, 52, and 53 illustrate this dilemma for one example run, showing the average time between resets for each strategy, the number of agents per strategy and the average time between resets weighted by the number of agents in the strategy over time. Although about 50% of the agents adopt the correlation strategy 3 (Figure 50 and Figure 53), the average time between resets for this strategy is even lower than that of the non-adaptive strategy 0 (Figure 51). This is due to the small population size of strategy 0. Only if the average time between resets is weighted by the population size (Figure 52), the largest population achieves the highest value. In reality, the seemingly large success of the static strategy is due only to the initial conditions, which provided for one successful agent in this strategy.

7.7 The Effect of Knowledge Propagation

The most significant improvement in agent success is achieved not by generating sets of new rules, but by spreading existing, working rule sets within the population. Even the longest runs without knowledge propagation have yielded only a success rate of 6 out 20 within 95000 time steps. The introduction of a method to transmit knowledge (which is encrypted in the rules) improved this ratio in the best case to 19 out of 20 agents being successful within 5000 time steps.

In comparison with the run in Chapter 7.6 the most significant difference between the two example runs is the higher mean time between resets for the copying set-up (Figure 56). This occurs already after 25000 time steps while the average for the non-copying set-up is taken after 95000 time steps. However, the average time between resets in the highest category is only half of what was observed for the non-copying set-up (1091 time steps when copying as opposed to 2211). On the other hand there are much fewer occurrences in the lowest category for the copying set-up (184 as opposed to 2254). In this case the propagation of rules within the population leads to few agents being successful for the entire duration of the run. The majority of agents is not reset for longer periods, but these agents not successful in the sense of the definition, although they are performing much better than the majority of agents in the non-copying set-up.

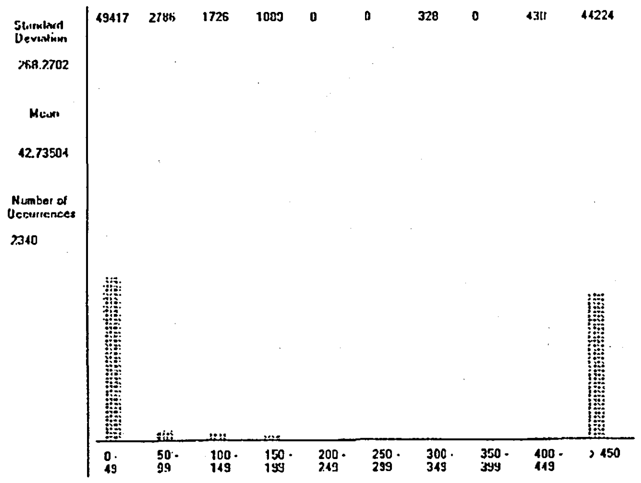

Figure 54: Weighted Distribution of Reset Times for Non-copying Set-up after 95000 Time Steps

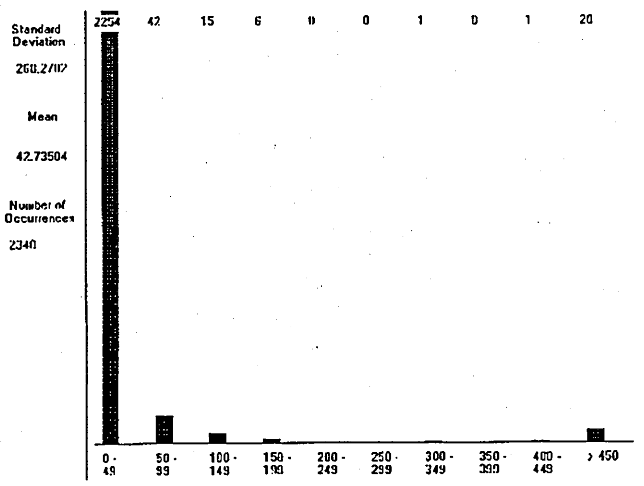

Figure 55: Distribution of Reset Times for Non-copying Set-up after 95000 Time Steps

It can be concluded that knowledge propagation on its own cannot be regarded as the universal solution to creating a successful adaptive system. The method for propagating knowledge used in this example application has its specific limitations. The blackboard of common knowledge will assign a new rule set to an unsuccessful agent when it would otherwise generate a new rule set according to its strategy, but the agent might use a different adaptation strategy to the one with which this rule set was generated. This recombination of strategies with rule sets can have positive as well as negative effects. In the case of a rule set generated with the correlation strategy, this might lead to discarding parts of the rule set which might be crucial if the agent applies one of the random techniques to it. In this case the value of copying an entire new rule set is greatly reduced, because the effect is no different to carrying on with the random strategy.

We have seen already in the reference case (Chapter 7.4) that the success of rule sets is circumstantial. Even rules, which normally work, might not work in the particular situation an agent is in. The effect of this is similar to an unfortunate matching of rule set and adaptation strategy, and will not improve the agent's performance. The same effect can be observed when the blackboard is too small, or by accident just contains one or only very few rule sets.

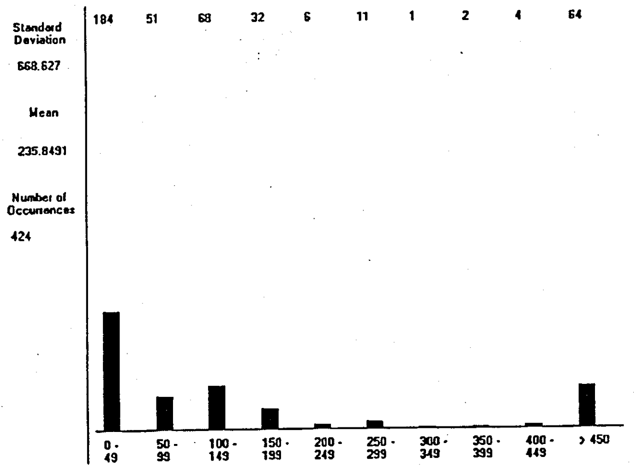

Figure 56: Distribution of Reset Times for Copying Agents after 25000 Time Steps (Example Run)

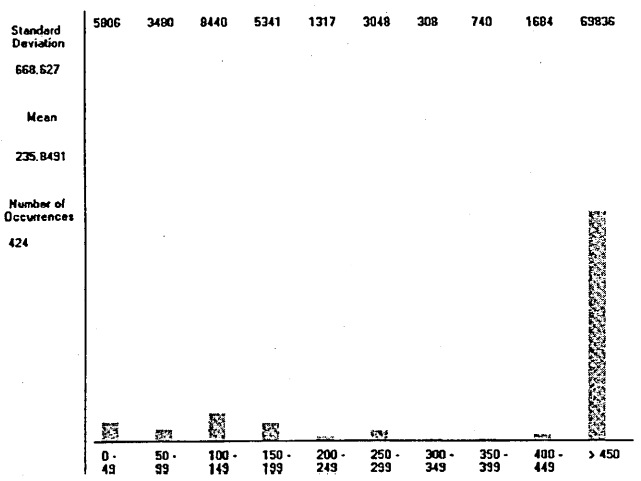

Figure 57: Weighted Distribution of Reset Times for Copying Agents after 25000 Time Steps (Example Run)

Consideration has to be given to the ratio between exploration and exploitation. Exploration means here that it is necessary for the system as a whole to discover new rule sets, at least at the beginning of a model run. Once a number of working rule sets is discovered, they can be exploited by the majority of the population. This reminds us of the discussion of "cartesian" and "stochast" behaviour in Allen and McGlade's (1987) fisheries model (see 4.5.1), where exactly the same problem occurred. Successful behaviours have to be discovered before they can be exploited. It seems trivial to make this point, but it is not obvious in a computer model that these behaviours will only improve the system as a whole when they are combined.

The best results in our case have been obtained with 80-85% of all resets copying from the blackboard. As this is a fixed ratio over the duration of a run, there is no possibility in the current set-up to let the system learn which ratio is advantageous at which point in time. The introduction of copying has also implications on the distribution of adaptation strategies. In Figure 58 the distribution of adaptation strategies for the run shown in Figure 56 and Figure 57 is shown. When compared to the distribution of the non-copying set-up (Figure 50) it becomes clear that the proportion of adapting agents is much smaller than in this configuration. This is due to the fact that the agents can only change their strategy when they are reset. As there is a high proportion of agents copying relatively successful rule sets before they are reset, there is no possibility for them to switch strategy. This leaves the system with many agents relying on the copied rule set and only a few adaptive explorers. Ideally, in the course of a model run the proportion of copying agents would increase over time when starting from random initial conditions, but a certain percentage of exploring agents would have to be retained in order to keep the system flexible to future change.

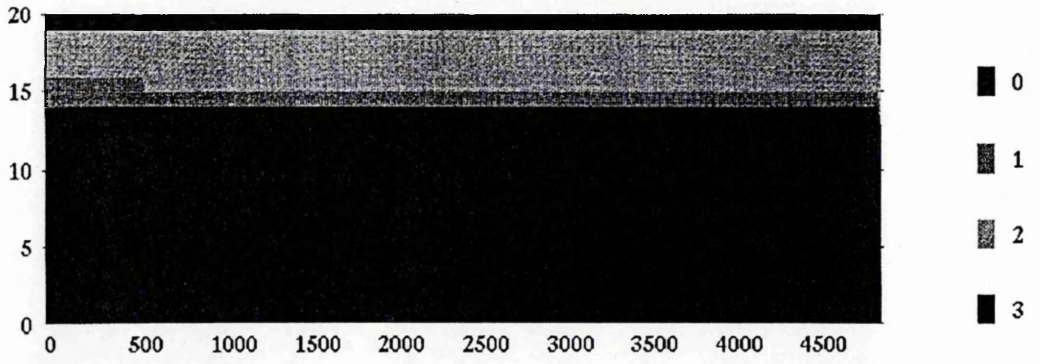

Figure 58: Distribution of Adaptation Strategies after 25000 Time Steps (Strategy 0: Non-adaptive; 1: Random Mutation of all Rules; 2: Random Mutation of Rules for Low Budgets; 3: Correlation)

Like the ratio of exploration to exploitation, it is a matter of definition what a "long lasting rule set" is. This is in our case defined as enabling at least one agent not to be reset for at least 500 time steps. In a truly evolutionary system on the other hand this measure would have to be evolved by the system itself. Because the blackboard is representing a knowledge pool common to the entire population, it is a matter of "public perception" to define what a long lasting / successful rule set is.

7.8 Summary

The results obtained from our agent-based model can give us some insight into the nature of complex adaptive systems. As the model in the form used can only deal with a fraction of the ideas outlined in the conceptual framework it is not truly evolutionary in the sense of the conceptual framework. However, the model has produced some very interesting results, which will be summarised below and discussed for their implications in the next chapter.

The first point to be made is not an obvious one: Even good rules do not guarantee success for an agent, although the environmental constraints were very much relaxed. The rules were defined in a way that common sense would make one think that the nature of the rules would enable an agent to satisfy its needs sufficiently. This point links into the problem of having only one set of homogeneous rules for the entire population in this configuration. The model set-up demands that some cooperative behaviour is needed. Even with the homogeneous rules cooperative behaviour develops to some extent, but could a diverse population make better use of the resources available to them? From this perspective diversity appears to be desirable for a system, because it can make cooperation (apart from the competition, which always exists) happen more easily.

Even in this very simple model there exists a mutual interdependency between environment and agents. The agent's actions transform the environment, which is the constraining factor for the success of the agents. These effects are significant. Unsuccessful agents are forced to find new resources, thereby developing new rule sets which in return indirectly affect all other agents through their influence on the environment. Also the concept of ecological niches can be found in the model, only that the niches exist here only in the sense that temporal / spatial use of facilities is possible only during certain times. In the case of a synchronised population and limited capacity niches open up for activities when most others agents do something else.

The system is extremely complex for its size. Success or failure of an agent depend to a great degree on what the other agents do, and this is not predictable, because it is determined a) by the experience of the other agents and their current rules as well as b) the random arrival order in a place at a time and c) the place's capacity. The prices as well have considerable influence on the individual. Prices are in the end determined by the collective (through demand) and have more influence on "poor" agents than on "rich" ones. This means that not only the environment, rules and experience determine the success of an agent, but its own budget states as well!

In the adaptive case it has been shown that already a small system has an enormous search space in which to find good rules. It is obvious that these rules are rare, and the duration of random searches by the agents shows that this is the case. However, these configurations appear not to be as rare as one would expect from the size of the search space, which allows for $\approx 3.4 \cdot 10^{38}$ permutations for an agent's rule set. Agents are able to find a limited number of successful rule sets very quickly using only the most basic random search techniques.

Comparing adaptation with the case of static rules, the results show that even simple adaptive mechanisms are performing better than a system starting from random static rules in creating long living agents. However, it is not clear whether the adaptive system performs better (in terms of achievable lifetime) than a conventional rule based system with good (or optimal) rules. This question links into the broader topic of whether natural systems optimise on their own. As far as Varela, Thompson and Rosch (1991) and the conceptual framework are concerned, optimisation is the extreme case of evolution, which only applies if the selective pressures are high enough. Natural systems would otherwise underperform. In this view it appears to be questionable that a natural evolutionary system would actually evolve in a way that is superior to a conventional rule based system which uses optimised rules. The difficulty here is just that these rules are very hard to find and that they are able to adapt to the change they inflict upon the environment.

As the adaptation techniques are only very basic, the problem of their effectiveness in the used set-up comes into focus. Agents have only very limited time to try new rules before they are replaced with a new set if there is no immediate improvement in the budget states. It is to be asked how many rules, which might work, are discarded before they can become effective. Other ways of improving the agents' performance include the fine-tuning of rules by changing parameters, which were not used in the example application, because this would further enlarge the search space. It has to be kept in mind that some rules might work only within a certain parameter range, which might not apply when the rule is generated. Ideally, the agents should find the adaptation technique on their own, thereby adding one more adaptation layer to the model. In the example only four predefined techniques have been used, but if one takes the conceptual framework seriously, the way of adaptation of the agents should be a result of their evolution.

However, the main result of the modelling exercise is that learning (or adaptation in general) can be regarded as an emergent property of an evolutionary system, provided that there is a chance for learning to come into existence. Adaptation gave some agents an advantage over the initial population with static rule sets. Even if adaptation proved not to give a permanent improvement, it would happen at least temporarily, but then possibly remain restricted to a small number of individuals, which achieve satisfactory results with this technique.

In our case the long model runs would have provided a chance for diffusion back to non-adaptation (which happens for some individuals, see 7.6). Generally it can be concluded that adaptation in the tested case gives an advantage over non-adaptation. The reason for this is that when starting from random initial conditions there has to be a good rule set in the first place for non-adaptation to work as shown in the comparison between the reference case and the single adaptation techniques.

There is a chance that an adaptation strategy develops a good rule set, which is then used successfully with non-adaptation (or another adaptation technique) so that it becomes also important to match adaptation techniques and specific rule sets. On the other hand it is possible that certain rule set only works in conjunction with a specific adaptation technique. All these factors make success seem impossible to reach, but in all runs a considerable number of agents finds these very narrow sustainable paths on their own without any interference from outside!

In the case of competing adaptation strategies even the most basic of performance measures are extremely difficult to apply to the results because of bias of age towards small populations. This bias can even outweigh weighting with the size of the population, but the population size alone is only an indicator, not a measure for the effectiveness of a strategy. This leads to the question whether there exist any meaningful quantitative measures for complex adaptive systems at all. On the other hand a qualitative lifecycle analysis which refers explicitly to the context of what happened appears to be an appropriate tool for tracing processes in such a complex system, but this does not allow for any aggregate description of a system.

Finally, knowledge propagation (or copying) proved to be a complementary technique to spread successful behaviours in the population. The number of successful agents can in this way be raised to about 95% of the population with the techniques used. In this context the issue of behavioural niches mentioned above becomes extremely important. As the process of copying homogenises the behavioural rules (and with it the behavioural patterns) of the population, rules which might work for some agents, and who subsequently write these rules to the common knowledge base, can fail when used in a different context. Furthermore, knowledge propagation slows down the spread of adaptation and therefore can possibly reduce the resilience of the system as a whole, although the improvements are considerable in the short term. Again, the issue of how far rules sets can be matched with different adaptation strategies as well as different environments than those in which the rules were developed is essential for the understanding of the behaviour of a complex adaptive system.

8 Discussion

In this section we will evaluate the results of the modelling exercise against the objectives with which the research was started, and an interpretation of the results will be given. The results of the modelling exercise have to be compared to some of the models outlined in Chapter 2 as well. The main differences in the possible interpretations of the results and the implications thereof will be clarified, before focusing on the contribution to knowledge this project has made.

The implications of this project lead to some very interesting questions, which are briefly outlined in the last section. This is intended to give an outlook to further research investigating the methodology used in this approach and possible applications of evolutionary models.

8.1 Evaluation and Interpretation of Results

The results reinforce the initial idea that a completely disaggregated model of a city based on the individual inhabitant is feasible. Much more information can be extracted from the disaggregate model regarding the implications of change on the population than by using higher level models. The evolutionary principle (as outlined in Chapter 5.2) - although implemented only in a rudimentary fashion - starts off a series of self-organising processes by letting diversity diffuse into the population. This symmetry breaking of the of the initial conditions is only partially due to the use of random number generation techniques, but also to the rule base set-up used in the model. The low level motivation for the agents - their needs - can lead to a multitude of different co-operative as well as competitive behaviours even with a single rule set, as we have seen in Chapter 7.4.1.

The introduction of the possibility to take up adaptive behaviour vindicates the conceptual framework even further. The agents eventually "learn to learn" by picking a learning strategy which is not necessarily "the best". The fact that sub-optimality is sufficient to create a successful system, but that even in an initially sub-optimal system the competitive pressure might eventually lead to a temporarily optimal system is the essence of Varela, Thompson and Rosch's (1991) framework to evolutionary systems. An indication for the sub-optimality of the systems is that even though adaptation gives these agents an advantage over non-adapting ones when starting from random initial conditions, there are still some agents in the system, which can get along by not adapting dynamically. The general case of an evolutionary system has therefore to be regarded as being satisficing only. This appears to be an important point for future approaches to modelling dynamic systems, because the traditional dynamic techniques usually apply optimisation algorithms, which invariably arrive at stable equilibria without being able to explore other possible, but suboptimal pathways.

The model has of course to provide the possibility for learning to come into existence in the first place, but as this taken into account in its design, adaptation will almost inevitably happen. The different evolutionary strategies originating from this point lead to clusters of rules, which cause different modes of behaviours for certain groups of the population. This is equivalent to the emergence of cultures in the population, all of which are oriented towards the ultimate goal of subsistence (as the satisfaction of needs can be translated). However, the emergence of cultures is not something, which can be regarded as happening always and at any point in time and space. It is the interconnectedness of any individual decision with most other issues in the system, which makes the evolution of cultures specific to the spatial and temporal setting, which in return is at least partly a product of the actions of the agents in the past. Figure 59 illustrates how a seemingly simple decision on how to travel has in fact to be regarded as being really an interpretation of the individual on the "meaning of life". But not only the individual's perception influences the decision, the conventions of society and the historic development of society and the environment play a major part in this decision as well. Although this might be the most consequential conclusion to be drawn from the model's results, it is to be determined to what extent this is to be incorporated into a model, as introducing complexity hinders the analysis of the results, as we have seen in the previous chapter.

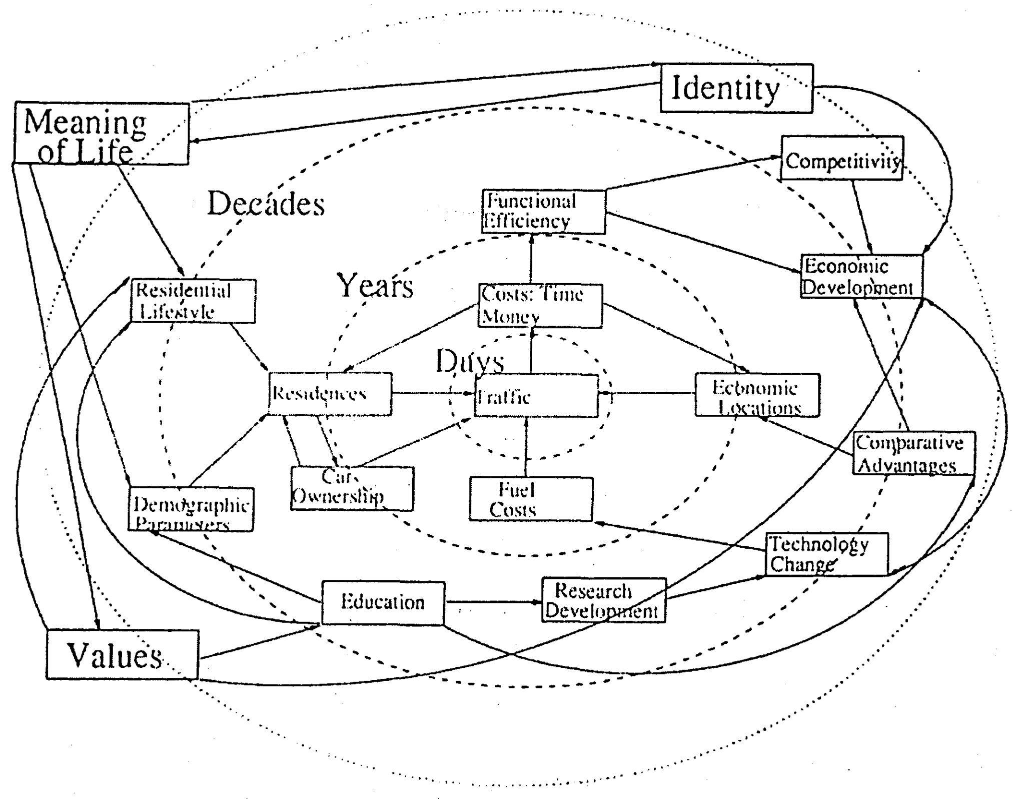

Figure 59: Web of Interactions on Everyday Decisions (after Allen 97b, p. 235)

The central point of the discussion turns out to be the issue of viability, in the first place individual viability, but because of the nature of the system, viability has to be regarded much more as a collective than an individual property. This leads to the issue of collective as opposed to individual utility of an action or a lifestyle. In this model the system was not complete. Resources were in theory unlimited and the only adverse effects of agent actions on their ability to survive were fluctuations in demand for activities. This is very much different from a real ecosystem (if one wants to use this term for a city) where resources are limited, and the system as a whole can degrade to the extent of not supporting its population any more.

Sustainability in the model relates to the way the individual behavioural rules are formed. As the agents do not have a long-term horizon, and the rules do not change any more once a satisficing solution is found, the initial individual viability might lead to adverse effects in the long term. Indication for this wasteful use of resources can be found when looking at the budget states of some successful agents. A big proportion of these agents builds up large surpluses in their goods budget, whilst another group just amasses money. The analogy to real world consumer culture is tempting... This issue is even more serious when one takes into account that the most efficient spread of successful behaviours occurs through copying and not by exploration.

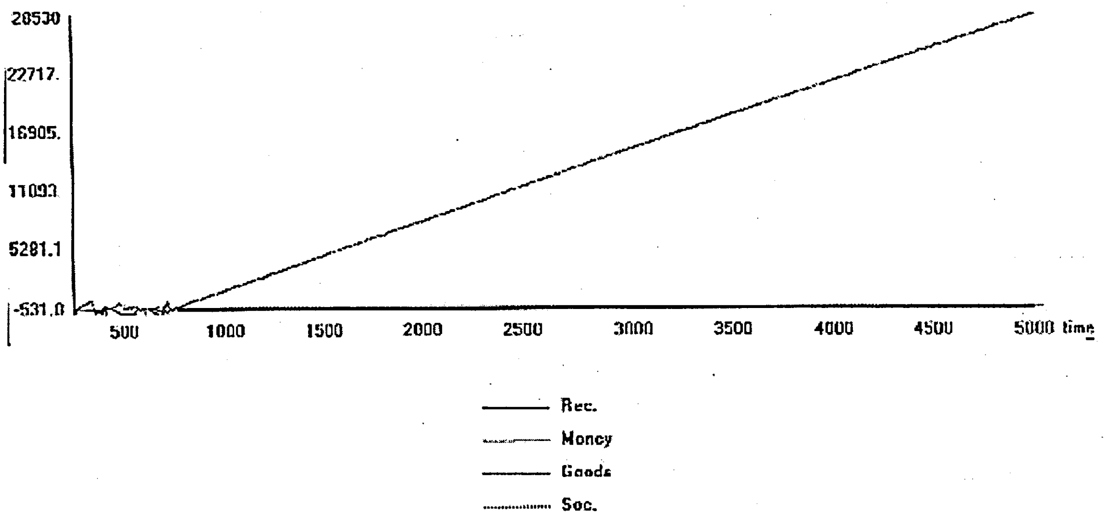

Figure 60: Agent Specialising in Accumulating Goods

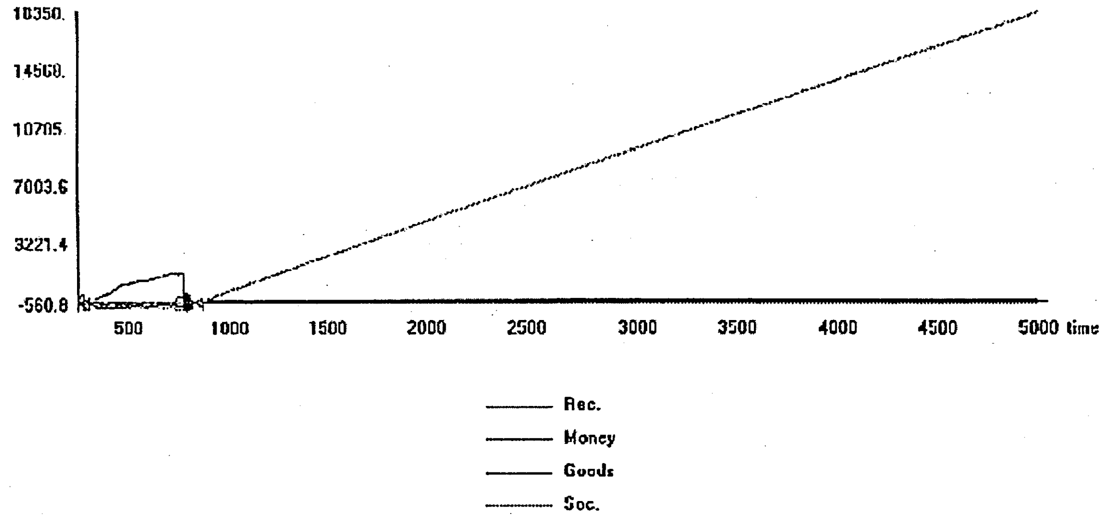

Figure 61: Agent Specialising in Socialising

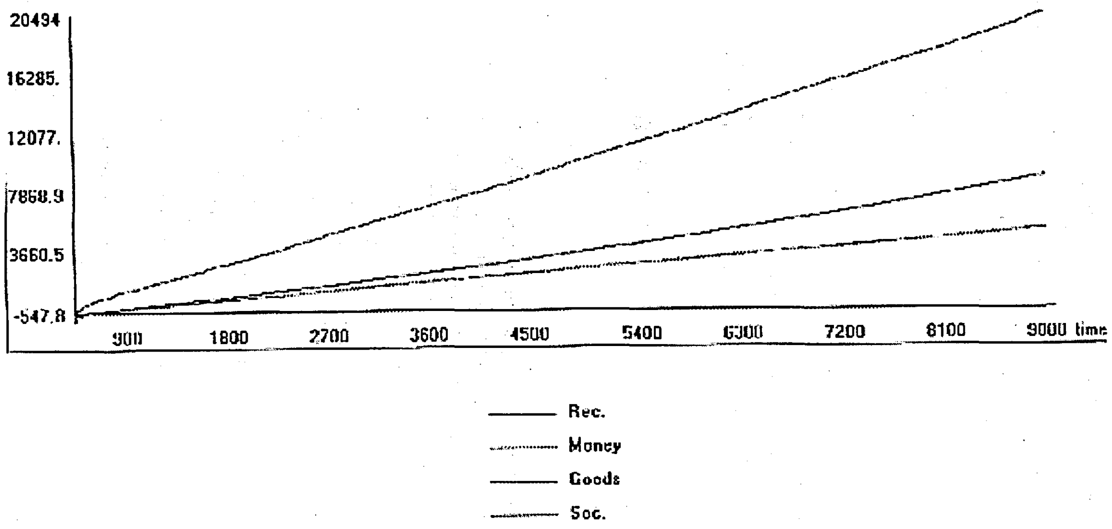

Figure 62: Agent Balancing three of the four Budgets

The spread of individually successful behaviours within the population takes place through imitation of good rule sets. Whilst in the case of individual exploration of the possibility space in the best case only 55% of agents end up with good rule sets after 95000 model days, imitation can lead to 95% of agents having successful rule sets after only 5000 model days. Still, imitation only does not lead to a viable population on its own, as the case of a population with predefined rules shows. The homogeneous rules can be seen as the limiting case of imitation, in which only one rule set is allowed. We have seen that this does not lead to success for the entire population. The conclusion from this is that what is needed for collective survival are "stochasts" (explorers) and "cartesians" (imitators) in the right ratio. As the best results of this mix of strategies have been obtained with a 80-90% proportion of imitation to exploration; "stochasts", many of which are initially unsuccessful, are only an insurance for the collective to a changing future and not of immediate value to the collective. The majority of the population is by far more successful when using the - at least in the short term - most effective way of communicating vital knowledge: copying.

The search for rules resembles the formation of cognition for an individual, because the rules are formed according to (at least some of) the experience of the agent. In the beginning there is no prioritisation of the needs which are expressed through the budget states. In the process of rule formation an individual hierarchy of needs can emerge through the construction of rules which discriminate some budget states for some activities. (If this budget is low/high *and* that budget is low/high *and* that budget is low/high then...) However, it is not obvious from the agent's behaviour whether in the end it is rules or needs which more evident to observe. This cognitive issue is especially evident when the rules have not been found through exploration and failure (leading to some kind of a cognitive picture of the world), but by copying somebody else. In this case it is to be asked whether the rules have anything to do with the defined (objective) needs. The agent's situation might lead to a very much different kind of rule set if experience instead of copying was used. The copied rules set might still work for the individual, but it definitely does not relate to the situation and the previous experience of the individual.

The question of emergence of new needs as seen by Max-Neef (see Section 4.3.3) might be addressed from a different perspective. If an agent copies a rule set which has been formed under different circumstances than those of the individual in question, the intrinsic hierarchy of the underlying needs might not fit directly the circumstances of the agent which copies. This can lead to deficiencies in some of the budgets, but there might be means available to cater for these deficiencies which were not needed before. This can mean that these new means are actually needed only in order to satisfy the budgets using an inadequate rule set. A new "need" for an individual has come into existence. In this case the new need is less an emergent property of the agents than a result of their behaviour. The defined needs of subsistence are still in existence, but in this case other means (or in Max-Neef's words: satisfiers) are required to satisfy the original needs. Although this is not a new need in the original sense, because there are no new budgets, this can amount to relying on more or other satisfiers in order to guarantee viability.

The methods of adaptation used in this example can be grouped into two categories: Mechanisms internal to the agents like rule mutation resemble evolutionary processes more than methods external to the agents - like copying - resemble learning. We have argued above that the methods of knowledge import from outside are much more efficient than the evolutionary process behind the formation of cognition. Furthermore, it has been shown that knowledge import does not necessarily lead to full satisfaction of all needs, but it is quicker and more effective at least in the short term. The question of purposefulness behind learning (and teaching as well) therefore leads to the interpretation that learning (knowledge import from outside) can ensure the immediate survival much better than exploration. However, it does not appear to be possible to relate knowledge import directly to the specific situation of the individual in question, as the knowledge itself has been generated in circumstances different to the ones of the importing individual. Knowledge about oneself cannot be taught, but only discovered in a probably very painful process of exploration.

The model's results reconfirm what has been known through empirical research in planning and the social sciences. The interaction between social and physical systems leads to a mutual interdependency between both systems; and change in one sub-system will affect the other sub-system as well. The methodology presented here can for the first time show this interaction and the implications thereof in a computer model, at least to some extent. An urban model can with this methodology make statements about lifestyles of the inhabitants and the consequences for the modelled urban system. Also it is easier to explain how the inhabitants' behaviour leads to change in the system and vice versa. In the past, statements on the implications of change in an urban system had been restricted to aggregate, mostly economic, parameters. The bottom-up, adaptive, agent-based approach, on the other hand, allows for the exploration of possible lifestyles for sub-populations through simulation on a theoretical level, a method that has not been available before in this form.

8.2 Evaluation of the Model

In spite of the multitude of extremely interesting results gained from running the model, it has become clear that the entirely disaggregated approach has its limits. In the current set-up the model can only represent 64 agents which on the computer platform used leads to run times of around 60 hours for a duration of 5000 time steps, thereby generating a data file of around 190 Mbytes. This means that a model of the size of even a little village is simply not feasible with the program used. Apart from this rather practical objection, the sheer mass of data generated makes any system containing more than about 20 agents extremely hard to analyse, as we have seen that the usual statistical aggregation measures are not applicable to this approach. Much of the logged data has not even been used for this thesis, as it was the primary aim to test the theoretical framework in its implications against reality. This leads on to the question of how far a model should go into detail, as one basic property of a model is that it is simplifying reality. In the case of an agent-based model aiming at the simulation of human behaviour there is the danger that the model becomes as complex as reality, which leads to a model that does not clarify anything because of its complexity, or even equals reality, in which case it would not be a model any more.

The disaggregated approach has its place in simulation, but it has to be determined to what extent it is to be used. For instance, cellular automata are much simpler in their implementation, and save considerably on computer power, when used in a land use model. However, cellular automata in this set-up cannot make any statements about the inhabitants, and usually cannot change their rule base during run time (although this is possible). On the other hand, a combination of models of different aggregation levels results on a philosophical level in a duality for the agents as they are represented several times in several functions, for example as an inhabitant, as part of a company, and as a traffic participant. These would be different entities in a hybrid model, but in reality it is the same person, who carries out different activities. A hybrid model might be more efficient in terms of computing time required, but it does for the reason above as well rely on very exact synchronising between the sub-models, which can otherwise lead to unrealistic fluctuations. As fluctuations are one of the most important elements of self-organising models, lack of synchronisation can cause the model take unrealistic pathways and therefore be misleading in its results.

On the other hand the disaggregated approach using adaptive agents can help here in spite of its demand for computer time. Agents can form "meta-agents" on a higher level of aggregation, which can help eliminate the requirement for synchronisation of sub-models. A "meta-agent" could, for instance, be a company, which is formed by several inhabitants. These inhabitants will have to perform some tasks within the "meta-agent", or in a simpler form, just be required to be in a certain place at a certain time in order to enable the "meta-agent" to perform its tasks. This extension of the methodology used in this project appears to be tempting, but is up to another project to actually implement such a system in an appropriate programming language on a suitable platform.

When comparing the existing model to some of the approaches discussed in Chapter 2, it might at first sight appear to be crude and oversimplifying. However, it can clarify some central points about adaptive systems, which these models cannot. Potentially even more than the results presented can be extracted from the model. Here for instance the question whether viable rules have a certain taxonomy and what the crucial moments for success or failure for the agents are by using lifeline analysis can be the contents of a follow-up project. On the other hand the model has many things in common with earlier approaches.

Allen and McGlade's (1987) fisheries model leads to the same implications on "cartesians" and "stochasts" from a completely different direction. The common problem is in both models that the viability of a community has to be ensured, and the solution to this is in both cases the mix of exploration and exploitation of resources and strategies. The question of the "correct" mix of strategies will probably be the subject of further research, especially since the issue of sustainable development has entered the discussion.

Gärling et al.'s (1998) approach to activity scheduling bears as well strong resemblance to the direction taken in this project. Satisficing according to Simon (1981) is one central element in the theory, and the activity scheduling process seems to be the way forward. In our example this has been neglected for the benefit of simplicity of model (agents making a plan for the day and deciding on the spot), but the process has to be embedded in a dynamic framework of cartesians and stochasts as well as the co-evolution of agents and environment for long term models

Adaptation has been seen as the key to the creation of complexity by Holland (1995). His methodology of classifier systems as an adaptation strategy is certainly a very effective way of implementing a cognitive approach to learning which is as well very much consistent with Varela, Thompson and Rosch's approach which underlies this project. Classifier systems are much more effective than the methods used in the model we are discussing, but the question of how much even classifier systems resemble the real world is still an issue. Classifier systems are in fact optimising to some extent, and they provide a way of learning from experience. As opposed to the model used in this project, they do not provide for copying from others, in fact the word "imitation" does not even appear in the index of Holland et al. (1986). Imitation has to be regarded as a very important factor in real-world adaptive complex systems according to the results of this project and other modelling exercises such as Allen and McGlade (1987). However, there is no objection why classifier systems could not be used in the context of adaptive systems as well if the cut-off criteria for adaptation are set to satisficing solutions. The lack of imitation has to be compensated for in the set-up of future models, because of the effects of un-fitting rule sets discussed above. Still, the use of adaptation strategies based on cognitive science respectively psychology would mean a step forward towards a more realistic model of human learning.

Self-organising zonal models for urban development (such as in Chapter 2.3.15) present a very pragmatic approach to the area of interest discussed in this project. Their relatively simple structure leads to easy-to-analyse results, which can be further refined by using separate sub models. These models use aggregate parameters to express the distribution of behaviours of a collective at some point in time. These parameters do not change over time and they do not have the multicriteria structure of the individual's needs in this project. The zonal models are deemed more appropriate for questions dealing with shorter time scales, as they do not provide a learning population, but on the other hand allow for self-organisation to take place. On top of this, they are much more compact and easier to set up and handle, whereas the completely disaggregate approach will always have a problem in calibrating the model to some initial condition. If it is possible to integrate and compress the essence of individual behaviour and learning into parameters (which then will be valid only for a limited time), the zonal approach has its appeal for models which explore the possibility space of an urban system.

One step even further to practicality are the current models in transportation micro simulation. On the other hand these are built on statistical parameters which, as we have outlined, do not change over time and reflect behavioural patterns of the past. If the time horizon of the model is only short and the interactions within the modelled system are known, there is nothing to say against statistical models, because they might in many cases be easier to build and faster running. On top of this these descriptive models are easier to interpret than explanatory models. If the aim of the modelling exercise is to forecast (actually: extrapolate) quantitative changes to a given system which is regarded as not changing over time, statistical modelling is the most appropriate way of approaching the problem. The evolutionary approach will be more appropriate if the aim of building the model is to learn about a badly understood system.

Learning about the nature of the system was the motivation of this project as well, as the road taken will probably never deliver a quantitative forecast of the system parameters, but it can help us understand more about the interactions in the system and possible pathways the system can take. The exploration of this possibility space is important, because we still do not know very much about the implications of urban change, especially in all those areas where social issues and life styles of the inhabitants and their influence on the city structure are concerned. So when taking up Hägerstrand's (1970) question "What about people in regional science?" again, it is to be said that we have come closer to the long-awaited answer: Here they are!

8.3 The Classification of Urban Models Revisited

Finally the taxonomy of urban models and the critique given in Chapter 2 have to be reassessed in the light of the results obtained. . In the course of this project it had at first to be clarified what an evolutionary model is and what the characteristics of an evolutionary model are. The methodology accounts for features, which have not been dealt with in the modelling community so far. Especially the key element - the mutual specification of system elements - is not found in models of adaptive complex systems so far. Although the computer model does not allow for all features of the conceptual framework to be implemented, the results are promising as outlined above. Tables 12 - 16 give a summary of the main points of critique of the concepts examined and matches these with the approach taken in this project.

It has been said that conceptual models are not a simulation method, but are a useful tool for conceptualising the processes taking place in a system. With the approach taken, it is possible to extend the area of use of a conceptual approach – here the modelling framework, which accounts for the nature of processes in an urban system – to an exploratory dynamic tool, which can generate scenarios on a qualitative level.

Some of the methodologies outlined in Chapter 2.3 already extend the range of phenomena, which can be treated with a model. For instance the implementation of non-linear dynamics into a model automatically discards the assumptions underlying an equilibrium approach as a self-organising system is by definition not at static equilibrium. The dominating methods of implementation for this kind of system are differential or master equations. These are however difficult to define and usually lead to a macro-view of the system with noise replacing fluctuations generated on the micro-scale. This can be overcome by following a micro-scale approach based on agents. Most agent-based models on the other hand are non-adaptive, which can be overcome with the methods outlined in the modelling framework.

Concept / Method	**Description / Critique**	**Approach followed in Project**	**Assessment**
Conceptual Model	Not a simulation method Can be used as a tool to understand system behaviour. The objective of these models is to learn about the system and not to forecast future behaviour	Simulation Model to understand the dynamics of individual behaviour in the urban context	More possibilities for exploration of possible pathways than with a purely conceptual model
Gravity Model	Does not match the structure of real settlements in terms of required symmetry and therefore violates the underlying assumptions Static Descriptive only when calibrated No representation of individual decision making	General preference of agents for nearer opportunities. Potential modification of preferences during run-time of model	The descriptions given by gravity-type models can be explained through the use of micro scale behavioural models
Entropy Maximisation	General form of gravity model Application of Shannon-Weaver entropy to urban systems. Results effectively in static probability distribution of weighting factors for gravity model No representation of individual decision making	No need for derivation of aggregate measures due to disaggregate model set-up Explicit representation of individual decision making	Regional variations of behaviour are an integral part of the approach followed. The aim to explain phenomena qualitatively does not require calibration to observed past behaviour.

Table 12: Comparison of Traditional Methods with the Approach taken

Concept / Method	Description / Critique	Approach followed in Project	Assessment
Optimisation Methods	Central assumption that there exists an optimal configuration for a city which minimises the use of certain resources. This configuration is time invariant Methods calculate the static equilibrium, which is not observed in reality. Optimises technical parameters and not the "quality of life"	Individual satisficing	Traditional optimising model had a different aim to this approach. Search for an optimal configuration for a city as opposed to the exploration of social dynamics
Equilibrium Methods	Pre-complex systems approach to projection of system states Complex systems do not necessarily tend to equilibrium No representation of individual decision making	No assumption about possible equilibria in the system. Disaggregate approach cannot make assumptions on macro behaviour of system	Equilibria occur between periods of structural system change. Assuming equilibrium conditions beforehand removes the chance to simulate structural change.
Discrete Choice	Descriptive measure of average preferences derived from past behaviour Static Relies on Utility function (see below)	Evolving preferences for individuals by rule adaptation Dynamic	Cognitive approach allows for the evolution of lifestyles and local cultures as opposed to static preferences in discrete choice models.
Utility Theory	Defined a utility function in order to describe people's potential gains from certain alternatives Usually linear and additive functions based on technical parameters Single choice criterion Static	System of individual's needs leading to multicriteria evaluation of given alternatives Dynamic Individualised perception of needs	Multiple criteria can be regarded as being more realistic than single function

Table 13: Comparison of Traditional Methods with the Approach taken (continued)

Concept / Method	Description / Critique	Approach followed in Project	Assessment
Statistical / Probabilistic Methods	Description of observed behaviour of the past. Parameter extrapolation Assumption of time invariance of parameters Limited explanatory value No representation of individual decision making	Explanatory approach aiming to generate the descriptive parameters captured in statistics from interaction in the system	Explanation for observed behaviour cannot be gained from descriptions of the past. Extrapolation of past behaviour into the future is methodologically questionable
Time Use / Time Budgets	Systematic description of possible action space of individuals Has been restricted to incorporate only technical parameters such as accessibility No representation of individual preferences	Time availability as constraining factor on behaviour Incorporation of personal preferences	Model extends the principles of time budget approaches from descriptive technical parameters to a dynamic framework which can be used in simulation of spatial behaviour and individual time use
Cognitive Approaches	Incorporation of approaches from cognitive science into models of human behaviour, which account explicitly for features such as incomplete knowledge, error making etc.	Explicit definition of perception in agents. "Learning" based on experience and perception of agent's state Limited knowledge for agents by cognitive map with qualitative attributes of environment	Incorporation of cognitive concepts into computer models leads to qualitatively superior results than purely economic approaches in explaining the behaviour of social systems
System Dynamics	Aggregate dynamic representation of systems Use of averaged parameters results in long term equilibrium Time and space invariant rules No representation of individual decision making	Potential for non-equilibrium dynamics resulting in self-organisation Adaptive behaviour on the micro scale	Aggregate deterministic descriptions are limited in their ability to reproduce the behaviour of complex systems

Table 14: Comparison of Traditional Methods with the Approach taken (continued)

Concept / Method	Description / Critique	Approach followed in Project	Assessment
Cellular Automata	Transition of a "cell's" state according to the state of neighbouring cells Dynamic Rules for all cells are identical and time and state invariant Representation of physical space philosophically questionable No representation of individual decision making	Physical space represented on a grid. No global rules on state change, but demand driven local change of capacities and prices.	Cellular automata are very efficient in their use of computing resources. In land-use modelling applications the representation of physical space containing transition rules by CA should be carefully assessed
Differential Equations	Dynamic method to describe the average system state Can be difficult to operationalise as mathematical descriptions for a system might be difficult to find (for instance for discontinuous systems) No representation of individual decision making	Linguistic rules representing Fuzzy Logic sets and operations	Systems built on differential equations can change their mode of behaviour quite radically, but it is not possible to structurally change the equations describing the system during run-time of a model. This limits the potential for the description of adaptation and learning.
Master Equation	Mathematically most appropriate method to incorporate fluctuations into a dynamic system. Probabilistic method No representation of individual decision making	Rule-based system. Fluctuations through random order of agents, but deterministic operation.	Rule-based systems appear to be easier to implement than mathematical descriptions of systems

Table 15: Comparison of Traditional Methods with the Approach taken (continued)

Concept / Method	Description / Critique	Approach followed in Project	Assessment
Fractals	Extension of Euclidean geometry applied to the growth of built-up areas No representation of economic, demographic and similar processes No representation of individual decision making	Development of city based on individual agents obeying basic economic processes	Fractal growth obeys purely geometric macro rules. There appears to be insufficient correspondence between these rules and the space use of social systems to apply this method to urban growth.
Self-Organisation	Main paradigm of complex systems theory. Results in temporarily stable configuration of dynamic systems Can account for multiple potential pathways described by one single set of equations or rules	No explicit incorporation. Local rules might lead to self-organising effects	Structural change occurs by means of self-organisation. This phenomenon can be observed throughout the living world (as well as in physics and chemistry). Models should provide for self-organisation to occur.
Evolutionary Models	Application of evolutionary theory to non-biological dynamic systems Extension of non-linear dynamic methods to incorporate diversity and emergence of new populations.	Consistent with main ideas of evolutionary models: diversity, change of attributes of agents over time, emergence of lifestyles	Most advanced methodology for modelling complex dynamic systems to data. Combination with agents (see below) leads to appropriate representation of dynamic social systems in computer models
Agents	Disaggregate method of using local rules for individual computational entities Potential for introduction of diversity and adaptation into a computer model	Based on application of this methodology including adaptation	Extremely useful bottom-up methodology for complex systems simulation, although demanding on computer power

Table 16: Comparison of Traditional Methods with the Approach taken (continued)

Other approaches do not bear a significant resemblance to the system. Above all the method of using fractal growth to simulate macroscopic dynamics of urban growth appear to have the same shortcomings of social physics approaches, such as the entropy maximising framework and the gravity model. These models apply analogues to physical laws to urban systems, while attempting to represent social phenomena. It has to be discussed in how far descriptions of the inanimate world can be applied to the animate world without explicit incorporation of cognitive and social theory.

For cellular automata the argument is in how far physical space can be represented by a cell that has the properties of an automaton. The transition rules of cellular automata are fixed in cell space, but in reality they correspond to a collection of entities like companies and infrastructure facilities, which can move from one place to another. Here it seems more appropriate to use agents in physical space instead and to attribute the transition rules to the agents. On the other hand the methods outlined in Chapter 2.3 have been found extremely useful to build the modelling framework on. The cognitive methods used in activity scheduling models feature a set-up very similar to the agent set-up in the model used here. Time Budget approaches have been around for a long time, but most of these treated the constraints imposed by transport and time availability as a description of technical feasibility. The dynamic aspects of change in time use patterns due to cultural change have not been incorporated in detail into these approaches. The road taken in this project results in the incorporation of change to behavioural patterns into the model itself.

Traditional behavioural approaches based on statistics cannot account for change in behavioural patterns either. The same is true for discrete choice models. Once the principles of behaviour are established from statistics, these cannot change during a simulation without contradicting the descriptive principle on which these approaches are built. In the case of utility models, the usually one-dimensional utility function is the same for the time of a simulation, while in this approach the prioritisation of a set of needs through change in an agent's perception is observed. The "utility function" is generated during run time of a simulation and remains changeable throughout.

However, the use of an evolutionary model is subject to some limitations. The evolutionary approach in its present form cannot be used in quantitative forecasting (as far as quantitative forecasting is possible in the first place, see Chapter 2.5), but allows only for explorations of qualitative change in a system. It is a tool to learn about the nature of the modelled system. It appears from the spread of adaptive strategies in the model that learning is an emergent property of a complex system. This means that for future models, adaptation strategies have not necessarily to be predefined but can be generated from within a model. Such a model has only to provide for the possibility that adaptation strategies can be formed, which can lead to either non-adaptive or adaptive systems.

The methodology developed can explore change on the micro-level of urban (and possibly other socio-natural) systems. As opposed to most traditional approaches to urban change which deal with change on a level of land use in spatial zones, it is now possible to explore the implications of this change on people's lifestyles and needs. The reverse (implications of changes in life styles on the urban structure) can be investigated with an evolutionary model as well, because the relationship between these two system elements is one of mutual dependency.

One of the most important results of this project, is however the issue of "cartesian" and "stochast" behaviour as discussed in Section 8.2, as this is a reconfirmation of results gained by earlier work. The implications of this for future models of adaptive behaviour cannot be underestimated. In the light of sustainable communities that recently are discussed ever more often, "cartesians" and "stochasts" are regarded to have a central place in understanding the dynamics of change, sustainability and viability. This result appeared as an unintended by-product of the project, but can provide a starting point for further research into sustainability.

Regarding the developed methodology itself, it is to be said in summary that although many of the points of critique of earlier methods can be eliminated, it appears to be useful at its present stage to complement larger scale models (based on zones in the urban case) rather than a standalone solution due to the computational requirements it imposes. The point of replacing models based on larger aggregates might be reached in the future, when more powerful computers will be available, but the intrinsic problem of model size to system size will remain and with it the difficulties of result analysis.

8.4 Outlook

In summary this project has not only come up with some valuable results, but also has given indication where future research on adaptive complex systems models might be heading. First of all, the conceptual framework has to be implemented to its full extent in a model relying exclusively on agents for all system elements. This will allow for adaptation (and evolution) on all sides, so that for a first time true co-evolution will be found in a computer model. On a small scale the use of "meta-agents" appears to be tempting, so that agents of one type can cluster in a higher level of different agent in order to perform different tasks than their primary aims. In this way the formation and behaviour of organisations, such as companies, co-operatives, or pressure groups can be integrated very elegantly into a single model. The computational requirements for this task should however not be underestimated.

The next step would be to test the methodology (with an appropriate model, of course) against a real-world problem, so that the applicability, which in this project has only been demonstrated on a very abstract level, can be assessed more comprehensively than it was possible in this project. This might answer the question of how realistic the integration of urban and transportation models is in reality (in terms of building a decision support tool), which was one of the motivations for this project. If proven successful such a model could provide a multitude of answers from a single model, for which in the past a number of different models were necessary. Examples for these might include:

- Traffic patterns in space and over time of day.
- Economic development and turnover / profits by sectors and areas.
- Lifestyle patterns for the population.
- Social disparities in the population.
- Stress factors for the population.
- Land values.
- Infrastructure requirements for the community.

Many more technical aspects of the model itself regarding for instance the relation of rules and what rules lead to successful behaviour in which contexts remain to be explored as well. Is there a taxonomy of successful rule sets and if so, how can such rule sets be classified? In this context the question of how rules relate to behaviour might be investigated further than in this place, too.

Finally, the pressing issue of sustainability can be addressed with the methodology outlined here. But not only the long term physical survival, but also how human needs might be satisfied more by the planned and built environment of a city can be the subject of a modelling exercise based on this project. However, the methodology is considered to provide a generic modelling framework for all kinds of socio-natural systems, and the urban context used in this place serves as just one example. Simulations of competing companies, organisations or farming communities and their environment (whether this is a market, society as a whole or the natural environment) appear to be equally feasible within this framework.

9 References

Allen, P. M. (1982). "Evolution, modelling, and design in a complex world". Environment and Planning B, vol. 9, 95 - 111

Allen, P. M. (1988). Evolution: Why the Whole is Greater Than the Sum of the Parts. In: *Wolff, W., Soeder, C.-J. and Drepper, F. R. (Eds.). Ecodynamics,* Springer, Berlin

Allen, P. M. (1997a). "Cities and regions as evolutionary, complex systems". Geographical Systems, vol. 4, 103 - 130

Allen, P. M. (1997b). *Cities and Regions as Self-organizing Systems.* Gordon and Breach, Amsterdam

Allen, P. M. (1998). Evolving Complexity in the Social Sciences. In: *Altmann, G. and Koch, W. A. (Eds.). Systems,* de Gruyter, Berlin

Allen, P. M. and Ebeling, W. (1983). "Evolution and the stochastic description of simple ecosystems". BioSystems, vol. 16, 113 - 126

Allen, P. M., Engelen, G., and Sanglier, M. (1983). Self-Organizing Dynamic Models of Human Systems. In: *Frehland, E. (Ed.). Synergetics: From Microscopic to Makroscopic Order*, Springer, Berlin

Allen, P. M. and McGlade, J. (1987). "Modelling complex human systems: A fisheries example". European Journal of Operations Research, vol. 30, 147 - 167

Allen, P. M. and Sanglier, M. (1981). "Urban evolution, self-organization, and decision making". Environment and Planning A, vol. 13, 163 - 183

Axhausen, K. and Gärling, T. (1992). "Activity-based approaches to travel analysis: conceptual frameworks, models, and research problems". Transport Reviews, vol. 12, no. 4, 323 - 341

Batty, M. (1976). *Urban Modelling.* Cambridge University Press, Cambridge

Batty, M. and Longley, P. A. (1986). "The fractal simulation of urban structure". Environment and Planning A, vol. 18, 1143 – 1179

Batty, M. and Longley, P. A. (1994). *Fractal Cities.* Academic Press, London

Batty, M. and Xie, Y. (1994). "From cells to cities". Environment and Planning B, vol. 21, s31 – s48

Beale, R. and Jackson, R. (1990). *Neural Computing: An Introduction.* IOP Publishing, Bristol

Beaumont, J. R., Clarke, M. and Wilson, A. G. (1981). "The dynamics of urban spatial structure: some exploratory results using difference equations and bifurcation theory". Environment and Planning A, vol. 13, 1473 - 1483

Ben-Akiva, M. and Lerman, S. R. (1985). *Discrete Choice Analysis.* MIT Press, Cambridge, Ma.

Bousquet, F. et al. (1994). Simulating Fisherman's Society. In: *Gilbert, N. and Doran, J. (Eds.). Simulating Societies.* UCL Press, London

Burgess, E.W. (1927). "The determination of gradients of growth of the city". Publications of the American Sociological Society, vol. 21, 178-184

Chapin, F. S.(1968a). "Activity systems and urban structure – a working schema". Journal of the American Institute of Planners, vol. 34, 11-18

Chapin, F. S. (1968b). Activity Systems as a Source of Inputs for Land Use Models. In: *Highway Research Board. Urban Development Models.* Washington D.C.

Chapin, F. S. and Logan, T. H. (1970). Patterns of Time and Space Use. In: *Perloff, H. S. (Ed.). The Quality of Urban Environment.* Resources for the Future Inc., Washington, D. C.

Christaller, W. (1933). *Die zentralen Orte in Südwestdeutschland.* Jena

Clarke, M., Dix, M. and Goodwin, P. (1982). "Some issues in forecasting travel behaviour – a discussion paper". Transportation, vol. 11, 153 - 172

Conte, R. and Castelfranchi, C. (1994). Mind is not Enough: The Precognitive Bases of Social Interaction. In: *Gilbert, N. and Doran, J. (Eds.). Simulating Societies.* UCL Press, London

Cordey-Hayes, M. (1974). On the Feasibility of Simulating the Relationship Between Regional Imbalance and City Growth. In: *Cripps, E. L. (Ed.). Space-Time Concepts in Urban and Regional Models.* Pion, London

Couclelis, H. (1985). "Cellular worlds: a framework for modelling micro-macro dynamics". Environment and Planning A, vol. 17, 585 - 596

Damm, D. and Lerman, S. R. (1981). "A theory of activity scheduling behaviour". Environment and Planning A, vol. 13, 703 - 718

Darwin, C. (1872). *The Origin of Species.* John Murray, London

Dendrinos, D. S. and Mullally, H. (1981). "Fast and slow equations: the development patterns of urban settings". Environment and Planning A, vol. 13, 819 - 827

Dendrinos, D. S. and Mullally, H. (1985). *Urban Evolution.* Oxford University Press, Oxford

Doran, J. and Palmer, M. (1995). The EOS Project: Integrating two Models of Palaeolithic Social Change. In: *Gilbert, N. and Conte, R.. Artificial Societies.* UCL Press, London

Doran, J. et al. (1994). The EOS Project - Modelling Upper Palaeolithic Social Change. In: *Gilbert, N. and Doran, J. (Eds.). Simulating Societies.* UCL Press, London

Ebeling, W. (1989). *Chaos – Ordnung – Information.* Harri Deutsch, Frankfurt am Main

Engelen, G., White, R. and Uljee, I. (1995). Integrating Constrained Cellular Automata Models, GIS and Decision tools to Support Urban Planning and Policy Making. In: *Timmermans, H. J. P. (Ed.). Decision Support Systems for Urban Planning*

Echenique, M. et al. (1974). "A disaggregated model of urban spatial structure: theoretical framework". Environment and Planning A, vol. 6, 33 - 63

Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Brookings Institution Press, Washington D.C., and MIT Press, Cambridge Ma.

Faulkner, H. W. (1978). Locational Stress on Sydney's Metropolitan Fringe. unpublished dissertation, Australian National University, Canberra

Festinger, L. (1957). *A Theory of Cognitive Dissonance*. Stanford University Press, Stanford, Ca.

Fisk, C. (1979). "More paradoxes in the equilibrium assignment problem". Transportation Research B, vol. 13, 305 - 309

Fisk, C. (1980). "Some developments in equilibrium traffic assignment". Transportation Research B, vol. 14, 243 - 255

Forrester, J. W. (1969). *Urban Dynamics*. MIT Press, Cambridge, Ma.

Gärling, T., Axhausen, K. and Brydsten, M. (1996). "Travel choice and the goal / process utility distinction". Applied Cognitive Psychology, vol. 10, 65 - 74

Gärling, T., Kwan, M. and Golledge, R. G. (1994). "Computational-process modelling of household activity schedules". Transportation Research B, vol. 28B, no. 5, 355 - 364

Gärling, T. et al. (1998). "Computer simulation of household activity scheduling". Environment and Planning A, vol. 30, no. 4, 665 - 679

Garner B. (1967). Models of Urban Geography and Settlement Location. In: *Chorley, R. J. and Hagget, P. (Eds.). Models in Geography*. Methuen & Co., London

Giddens, A. (1986). *The Constitution of Society: Outline of the Theory of Structuration*. University of California Press, Berkeley, Ca.

Gleick, J. (1987). *Chaos*. Penguin, New York.

Goodwin, P., Kitamura, R. and Meurs, H. (1990). Some Principles of Dynamic Analysis of Travel Behaviour. In: *Jones, P. (Ed.). Developments in Dynamic and Activity-Based Approaches to Travel Analysis*. Avebury, Aldershot

Haag, G. et al. (1992). Interurban migration and the dynamics of a system of cities: The stochastic framework with an application to the French urban system". Environment and Planning A, vol. 24, 181 - 198

Hägerstrand, T. (1970). "What about People in Regional Science?". Papers of the Regional Science Association, vol. 24, 7 - 21

Harris, C.D. and Ullman, E. L. (1945). "The nature of cities". Annals, American Academy of Political and Social Science, vol. 242, 7-17

Hebb, D. O. (1949). *The Organization of Behaviour*. Wiley, New York

Heuser-Keßler, M. et al. (1994). "Die Architektur des Komplexen". ARCH$^+$ 121, 38 - 42, Aachen

Hiett, P. (1997). On the Nature of Wholes and the Appropriate Way of Approaching Them. unpublished PhD thesis, Cranfield University

Hirsh, M., Prashkea, J. N. and Ben-Akiva, M. (1986). "Dynamic model of weekly activity pattern". Transportation Science, vol. 20, no. 1, 24 - 36

Holland, J. H. (1995). *Hidden Order*. Addison-Wesley, Reading, Ma.

Holland, J. H. et al. (1986). *Induction*. MIT Press Cambridge, Ma.

Hoyt, H. (1939). The Structure and Growth of Residential Neighbourhoods in American Cities. Federal Housing Administration, Washigton D.C.

Jeffrey, P. (1996). "Evolutionary analogies and sustainability: putting a human face on survival". Futures, vol. 28, no. 2, 173 – 187

Jones, P. and Clarke, M. (1988). "The significance and measurement of variability in travel behaviour". Transportation, vol. 15, 65 - 87

Jones, P. et al. (1983). *Understanding Travel Behaviour*. Gower, Aldershot

Kahn, D., Deneubourg, J. L. and de Palma, A. (1981). "Transportation mode choice". Environment and Planning A, vol. 13, 1163 - 1174

Kahn, D., de Palma, A. and Deneubourg, J. L. (1981). "Noisy demand and mode choice". Transportation Research B, vol. 19, 143 - 153

Kimura, M. (1985). Natural Selection and Neutral Evolution. In: *Gofrey, L. R. (Ed.). What Darwin Began: Modern Darwinism and Non-Darwinist Perspectives*, Allyn and Bacon, Boston

Kirman, A. P. (1994). Economies with Interacting Agents. Santa Fe Institute Working Paper 94-05-030, Santa Fe, N.M.

Kitamura, R. (1984). "A model of daily time allocation to discretionary out-of-home activities and trips". Transportation Research B, vol. 18B, no. 3, 255 - 266

Kitamura, R. (1988). "An evaluation of activity-based travel analysis". Transportation, vol. 15, 9 - 34

Kosko, B. (1991). *Neural Networks and Fuzzy Systems*. Prentice-Hall, Englewood Cliffs, N.J.

Kosko, B. (1994). *Fuzzy Thinking*. Flamingo, London

Kutter, E. (1973). "A model for individual travel behaviour", Urban Studies, vol. 10, 235 - 258

Kutter, E. (1984). Integrierte Berechnung Städtischen Personenverkehrs, Dokumentation der Entwicklung eines Verkehrsberechnungsmodells für die Verkehrsentwicklungsplanung Berlin (West). Im Auftrag der Senatoren für Bau- und Wohnungswesen und Stadtentwicklung und Umweltschutz, Berlin

Landauer, T. K. (1972). *Psychology: A Brief Overview*. McGraw-Hill, New York

Le Fur, J. (1995). "Modeling adaptive fishery activities facing fluctuating environments: an artificial intelligence approach". AI Applications in Natural Resources, Agriculture, and Environmental Sciences, vol. 9, no. 1, 85 - 97

Le Fur, J. (in press). Simulating a Fishery Exploitation: Application to the Small-scale Fishery in Senegal. In: *IIFET '96 proc. Symposium,* 7/96, Marrakech

Lee, C. (1973). *Models in Planning*. Pergamon Press, Oxford

Lerman, S. R. (1983). Random Utility Models of Spatial Choice. In: *Hutchinson, B. G., Nijkamp, P. and Batty, M. (Eds.). Optimization and Discrete Choice in Urban Systems. Proceedings, Waterloo, Canada, 1983*. Springer, Berlin

Levins, R. (1966). "The strategy of model building in population biology". American Scientist, vol. 54, no. 4, 421 - 431

Lighthill, M. H. and Whitham, G. B. (1955). "On kinematic waves II". Proc. R. Soc. Ser., vol. A229, no. 1178, 317 - 345

Longley, P. A., Batty, M. and Fotheringham, A. S. (1992). The Geometry of Urban Form and the Fractal Nature of Urban Growth: Presentation Notes. In: *Pumain, D. (Ed.). Spatial Analysis and Population Dynamics*. John Libbey, Montrouge

Lowry, I. S. (1964). A Model of Metropolis. RM-4035-RC, Rand Corporation, Santa Monica, Ca.

Luhmann, N. (1987). *Soziale Systeme*. Suhrkamp, Frankfurt am Main

Mahmassani, H. S. (1988). "Some comments on activity-based approaches to the analysis and prediction of travel behaviour". Transportation, vol. 15, 35 - 40

Maslow, A. H. (1954). *Motivation and Personality*. Harper and Row, New York

Maturana, H. R. and Varela, F. J. (1987). *Der Baum der Erkenntnis*. Goldmann, München

Max-Neef, M. A. (1991). *Human Scale Development*. Apex Press, New York

Minsky, M. (1986). *The Society of Mind*. Simon and Schuster, New York

National Travel Survey 1975/76. Government Statistical Service, HMSO

National Travel Survey 1985/86. Government Statistical Service, HMSO

Nauck, D., Klawonn, F. and Kruse, R. (1994). *Neuronale Netze und Fuzzy-Systeme*. Vieweg, Braunschweig

New Scientist (1997). "The unselfish gene". New Scientist, 25 October 1997

Parisi, D., Cecconi, F. and Cerini, A. (1995). Kin-directed Altruism and Attachment Behaviour in an Evolving Population of Neural Networks. In: *Gilbert, N. and Conte, R.. Artificial Societies*. UCL Press, London

Parr, J. B. (1981). "Temporal change in a central-place system". Environment and Planning A, vol. 13, 97 - 118

Pas, E. I. (1988). "Weekly travel-activity behaviour". Transportation, vol. 15, 89 - 109

Portugali, J. and Benenson, I. (1995). "Artificial planning experience by means of a heuristic cell-space model: simulating international migration in the urban process". Environment and Planning A, vol. 27, 1647 - 1665

Portugali, J. and Benenson, I. (1997). Human Agents between Local and Global Forces in a Self-organizing City. In: *Schweitzer, F. (Ed.). Self-Organization of Complex Structures*. Gordon and Breach, London

Portugali, J., Benenson, I., and Omer, I. (1997). “Spatial cognitive dissonance and socio-spatial emergence in a self-organizing city”. Environment and Planning B, vol. 24, 263 - 285

Pred, A. R. (1973). The Growth and Development of Systems of Cities in Advanced Economies. In: *Pred, A. R. and Törnquist, G.r E.. Systems of Cities and Information Flows*. Lund Studies in Geography. C. W. K. Gleerup, Lund

Rasmussen, S. and Barrett, C. L. (1995). Elements of a Theory of Simulation. In: *ECAL 95, Lecture Notes in Computer Science*, Springer, Berlin

Rechenberg, I. (1973). *Evolutionsstrategie*. Frommann-Holzboog, Stuttgart-Bad Cannstatt

Recker, W.W. and McNally, M. G. (1986a). “A model of complex travel behaviour: part I - theoretical development”. Transportation Research A, vol. 20A, no. 4, 307 - 318

Recker, W. W. and McNally, M. G. (1986b). “A model of complex travel behaviour: part II - an operational model”. Transportation Research A, vol. 20A, no. 4, 319 - 330

Reif, B. (1973). *Models in Urban and Regional Planning*. Leonard Hill, Aylesbury

Sanders, L. et al. (1997). “SIMPOP: a multiagent system for the study of urbanism”. Environment and Planning B, vol.24, no.2, 287 - 305

Shannon, C. and Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Il.

Sharpe, R. (1983). An Optimum Economic/Energy Land-use Transportation Model. In: *Hutchinson, B. G., Nijkamp, P. and Batty, M. (Eds.). Optimization and Discrete Choice in Urban Systems. Proceedings, Waterloo, Canada, 1983*. Springer, Berlin

Simon, H. A. (1981). *The Sciences of the Artificial*. MIT Press, Cambridge, Ma.

Sinha, H. K., Khanna, S. K. and Arora, M. G. (1983). Choice of Urban Transport Modes for Work Trips. In: *Hutchinson, B. G., Nijkamp, P. and Batty, M. (Eds.). Optimization and Discrete Choice in Urban Systems. Proceedings, Waterloo, Canada, 1983*. Springer, Berlin

Smith, M. J. (1979). “The existence, uniqueness and stability of traffic equlibria”, Transportation Research B, vol. 11, 295 - 304

Sorokin, P. A. and Berger, C. Q. (1939). *Time Budgets of Human Behaviour*. Harvard Soc. Studies, vol. 2, Cambridge, Ma.

Szalai, A. (Ed.) (1972). *The Use of Time*. Mouton, The Hague

Varaprasad, N. and Cordey-Hayes, M. (1982). “A dynamic urban growth model for strategic transport planning”. Transportation Planning and Technology, vol. 7, 109 - 120

Varela, F. J., Thompson, E. and Rosch, E. (1991). *The Embodied Mind*. MIT Press, Cambridge, Ma.

von Thünen, J. H. (1826). *Der isolierte Staat in Beziehung auf Landwirtschaft und Nationalökonomie*, Hamburg

Walmsley, D. J. (1988). *Urban Living*. Longman, Harlow

Wardrop, J. G. (1952). "Some theoretical aspects of road traffic research". Proc. Institute of Civil Engineers, Part II, vol. 1, 325 - 378

Weidlich, W. (1997). From Fast to Slow Processes in the Evolution of Urban and Regional Settlement Structures. In: *Schweitzer, F. (Ed.). Self-Organization of Complex Structures*. Gordon and Breach, London

Weidlich, W. and Haag, G. (Eds.) (1988). *Interregional Migration*. Springer, Berlin

Wegener, M. (1983). Description of the Dortmund Model. Arbeitspapier 8, Institut für Raumplanung, Universität Dortmund

Wegener, M. (1985). The Dortmund Housing Market Model: a Monte Carlo Simulation of a Regional Housing Market. In: *Stahl, K. (Ed.). Microeconomic Models of Housing Markets*. Springer, Berlin

Wesson, R. (1991). *Beyond Natural Selection*. MIT Press, Cambridge, Ma.

White, R. and Engelen, G. (1993). "Cellular automata and fractal urban form: a cellular modelling approach to the evolution of urban land-use patterns." Environment and Planning A, vol. 25, 1175 - 1199

Wilson, A. G. (1970). *Entropy in Urban and Regional Modelling*. Pion, London

Wilson, A. G. (1972). *Papers in Urban and Regional Analysis*. Pion, London

Wilson, A. G. (1977). Some Advances in Urban and Regional Modelling. In: *Wilson, A. G., Rees, P. H. and Leigh, C. M.. Models of Cities and Regions*. John Wiley, Chichester

Wolpert, D. H. (1996). An Incompleteness Theorem for Calculating the Future. Santa Fe Institute Working Paper 96-03-008, Santa Fe, N.M.

Zadeh, L. A. (1965). "Fuzzy sets". Information and Control, vol. 8, 338 - 353

Appendix: Listing of the Computer Program Used

Barchart.frm

Sub Command1_Click ()

```
Unload barchart
End Sub
```

Sub Command2_Click ()

```
PrintForm
End Sub
```

Sub Form_Load ()

```
label2.Caption = hscroll1.Value
Label4.Caption = hscroll2.Value
LoadActivity
CalcDem
End Sub
```

Sub HScroll1_Change ()

```
label2.Caption = hscroll1.Value
CalcDem
End Sub
```

Sub HScroll2_Change ()

```
Label4.Caption = hscroll2.Value
CalcDem
End Sub
```

Budgraph.frm

Sub Command1_Click ()

```
Unload BudGraph
End Sub
```

Sub Command2_Click ()

```
PrintForm
End Sub
```

Sub Command3_Click ()

```
RelWealth
DrawRelWealth
BudGraph.Caption = "Relative Welfare of Agent"
End Sub
```

Sub Command4_Click ()

```
FindMinMax
DrawBudgets
BudGraph.Caption = "Budget Graph of Agent"
```

```
End Sub

Sub Command5_Click ()

FindCapPriMinMax
DrawCap
BudGraph.Caption = "Capacities of Cell"
End Sub

Sub Command6_Click ()

FindCapPriMinMax
DrawPri
BudGraph.Caption = "Prices in Cell"
End Sub

Sub Command7_Click ()

RelWealth
FindWealthMinMax
DrawWealth
BudGraph.Caption = "Total Wealth of System"
End Sub

Sub Form_Load ()

Dim i As Integer

For i = 0 To 3
        Line1(i).BorderColor = QBColor(2 * i)
Next i

label1.Caption = Hscroll1.Value
label2.Caption = HScroll2.Value
label5.Caption = "Cell " & HScroll3.Value

LoadBudgets
LoadCapPri
FindMinMax
DrawBudgets
End Sub

Sub HScroll1_Change ()

label1.Caption = Hscroll1.Value
label2.Caption = HScroll2.Value

FindMinMax
DrawBudgets
End Sub

Sub HScroll2_Change ()

label1.Caption = Hscroll1.Value
label2.Caption = HScroll2.Value

FindMinMax
DrawBudgets
End Sub

Sub HScroll3_Change ()
```

```
label5.Caption = "Cell " & HScroll3.Value

FindCapPriMinMax
DrawCap
End Sub
```

Cachange.frm

Sub Command1_Click ()

```
CaChange.Hide
End Sub
```

Capaform.frm

Sub Command1_Click ()

```
capaform.Hide
End Sub
```

Sub Text1_Change (Index As Integer)

```
text1(1).text = Int(Sqr(text1(2).text)) + Int(Sqr(text1(3).text))
End Sub
```

Dataent.frm

Sub Command1_Click ()

```
DATA1.Recordset.Update
End Sub
```

Sub Command2_Click ()

```
Unload NameAct
End Sub
```

Sub Command3_Click ()

```
tableadd
End Sub
```

Decay.frm

Sub Command1_Click ()

```
dec.Hide
End Sub
```

Defuzzy.frm

Sub Command1_Click ()

```
Unload defuzenter
End Sub
```

Sub Command2_Click ()

```
If label2.Visible = True Then
        cmdefstore
```

```
ElseIf label3.Visible = True Then
        defuzstore
End If
End Sub
```

Sub Command3_Click ()

```
label2.Visible = True
label3.Visible = False

cmdefretrieve
End Sub
```

Sub Command4_Click ()

```
label3.Visible = True
label2.Visible = False

defuzretrieve
End Sub
```

Sub Text1_Change ()

```
Dim i As Integer

For i = 0 To 3
        io_edit.Text2(i) = defcen(i)
Next i
End Sub
```

Delform.frm

```
Dim msg As String
```

Sub Command1_Click ()

```
Kill "C:\blah.mdb"
delform.Hide
End Sub
```

Sub Command2_Click ()

```
delform.Hide
End Sub
```

Sub Form_Load ()

```
msg = "This Command deletes all previous Simulation Data unless You have already saved it saved to a
different File !"
label1.Caption = msg
End Sub
```

Displayr.frm

```
Option Explicit
```

Sub Command1_Click ()

```
Unload DisplayRes
End Sub
```

Sub Form_Load ()

```
Dim i As Integer
Dim j As Integer

grid1.Row = 0
grid1.Col = 0
grid1.ColWidth(0) = 800

For i = 1 To 3
        grid1.Row = i
        grid1.Text = 10 * i
        grid1.RowHeight(i) = 1000
Next i

grid1.Row = 0

For j = 1 To 3
        grid1.Col = j
        grid1.ColWidth(j) = 1000
        grid1.Text = 10 * j
Next j
End Sub
```

Sub HScroll1_Change ()

```
text3.Text = Hscroll1.Value
displayactivity
End Sub
```

Sub Option1_Click ()

```
Dim db As database
Dim t As table

Set db = OpenDatabase("C:\blah.mdb", True)
Set t = db.OpenTable("typedescriptor")
        t.MoveFirst
        data1.RecordSource = t("name") & text4.Text
        data1.Refresh
t.Close
db.Close
End Sub
```

Sub Option2_Click ()

```
Dim db As database
Dim t As table

Set db = OpenDatabase("C:\blah.mdb", True)
Set t = db.OpenTable("typedescriptor")
        t.MoveFirst
        t.MoveNext
        data1.RecordSource = t("name") & text4.Text
        data1.Refresh
t.Close
db.Close
End Sub
```

Sub Option3_Click ()

```
Dim db As database
Dim t As table
```

```
Set db = OpenDatabase("C:\blah.mdb", True)
Set t = db.OpenTable("typedescriptor")
        t.MoveFirst
        t.MoveNext
        t.MoveNext
        data1.RecordSource = t("name") & text4.Text
        data1.Refresh
t.Close
db.Close
End Sub
```

Sub Option4_Click ()

```
Dim db As database
Dim t As table

Set db = OpenDatabase("C:\blah.mdb", True)
Set t = db.OpenTable("typedescriptor")
        t.MoveFirst
        t.MoveNext
        data1.RecordSource = t("name") & text4.Text
        data1.Refresh
t.Close
db.Close
End Sub
```

Sub p_res_Click ()

```
PrintForm
End Sub
```

Distribu.frm

Sub Command1_Click ()

```
Unload distribu
End Sub
```

Sub Command2_Click ()

```
PrintForm
End Sub
```

Sub Command3_Click ()

```
ReadWeighRes
End Sub
```

Sub Command4_Click ()

```
ReadResults
CalcStd
DrawDistribu
End Sub
```

Sub Form_Load ()

```
ReadResults
CalcStd
```

```
DrawDistribu
End Sub
```

Fuzzyent.frm

Sub Command1_Click ()

```
Unload fuzzyenter
End Sub
```

Sub Command2_Click ()

```
If label1.Visible = True Then
        cmfuzstore
ElseIf label2.Visible Then
        fuzzstore
End If
End Sub
```

Sub Command3_Click ()

```
label1.Visible = True
label2.Visible = False

CMfuzretrieve
End Sub
```

Sub Command4_Click ()

```
label2.Visible = True
label1.Visible = False

fuzzretrieve
End Sub
```

Inform.frm

Inputf.frm

Sub actcho_Click ()

```
DisplayRes.Show
End Sub
```

Sub actpay_Click ()

```
payoff.Show
End Sub
```

Sub AgBud_Click ()

```
Budgraph.Show
End Sub
```

Sub budname_Click ()

```
NameAct.Show
End Sub
```

Sub capaset_Click ()

```
capaform.Show
End Sub
```

Sub capdis_Click ()

```
barchart.Show
End Sub
```

Sub ChanCa_Click ()

```
CaChange.Show
End Sub
```

Sub ChaPr_Click ()

```
PrChange.Show
End Sub
```

Sub Clear_Res_Click ()

```
Dim db As database
Dim t1 As table

Set db = OpenDatabase("C:\results.mdb", True)
Set t1 = db.OpenTable("Age")
        t1.MoveFirst

                Do Until t1.EOF
                t1.Delete
                t1.MoveNext
                Loop
t1.Close

For i = 1 To 19
        Set t1 = db.OpenTable("A" & i)
                t1.MoveFirst

                Do Until t1.EOF
                        t1.Delete
                        t1.MoveNext
                Loop
        t1.Close
Next i
db.Close
End Sub
```

Sub clearDB_Click ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim t3 As table
Dim i As Integer
Dim j As Integer
Dim k As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t2 = db.OpenTable("typedescriptor")
        t2.MoveFirst
        TotalCount = t2("number")
```

```
t2.Close

For j = 1 To 4
        For i = 0 To TotalCount - 1
                Set t1 = db.OpenTable(j & i)
                t1.MoveFirst
                t1.MoveNext

                Do Until t1.EOF
                        t1.Delete
                        t1.MoveNext
                Loop
                t1.Close
        Next i
Next j

For k = 0 To 8
        Set t3 = db.OpenTable("world" & k)
        t3.MoveFirst
        t3.MoveNext

        Do Until t3.EOF
                t3.Delete
                t3.MoveNext
        Loop
        t3.Close
Next k
db.Close
End Sub
```

Sub CogRan_Click ()

```
RanCogInit
End Sub
```

Sub Command3D1_Click ()

```
runform.Show
End Sub
```

Sub Command3D2_Click ()

```
barchart.Show
End Sub
```

Sub Command3D3_Click ()

```
Budgraph.Show
End Sub
```

Sub Command3D4_Click ()

```
DisplayRes.Show
End Sub
```

Sub Command3D5_Click ()

```
blah
End Sub
```

Sub Command3D6_Click ()

```
delform.Show
End Sub
```

Sub Command3D7_Click ()

```
payoff.Show
End Sub
```

Sub CreDB_Click ()

```
blah
End Sub
```

Sub d5rec_Click ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim t3 As table
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t2 = db.OpenTable("typedescriptor")
        t2.MoveFirst
        TotalCount = t2("number")
t2.Close

For j = 1 To 4
        For i = 0 To TotalCount - 1
        Set t1 = db.OpenTable(j & i)
                t1.MoveFirst
                For l = 0 To 4999
                        t1.Delete
                        t1.MoveNext
                Next l
        t1.Close
        Next i
Next j

For k = 0 To 8
        Set t3 = db.OpenTable("world" & k)
                t3.MoveFirst
                For l = 0 To 4999
                        t3.Delete
                        t3.MoveNext
                Next l
        t3.Close
Next k
db.Close
End Sub
```

Sub deca_Click ()

```
dec.Show
End Sub
```

```
Sub defuz_Click ()

defuzenter.Show
End Sub

Sub DeleteSim_Click ()

delform.Show
End Sub

Sub DTS_Click ()

DrawTSeries
End Sub

Sub fuzzy_Click ()

fuzzyenter.Show
End Sub

Sub grd_Click ()

LogAge
End Sub

Sub ioset_Click ()

io_edit.Show
End Sub

Sub lsmnu_Click ()

get_ls
End Sub

Sub price_Click ()

pricefor.Show
End Sub

Sub quit_Click ()

End
End Sub

Sub randinit_Click ()

RandomInit
End Sub

Sub ranpar_Click ()

ranparfor.Show
End Sub

Sub resag_Click ()

ResetForm.Show
End Sub

Sub rtbar_Click ()
```

```
GetAge
GetTSeries
End Sub
```

Sub save_Click ()

```
Dim DestFile, msg                    ' Declare variables.
On Error GoTo ErrHandler

CMDialog1.Filter = "Databases (*.mdb)|*.mdb"
CMDialog1.Action = 2

DestFile = CMDialog1.Filename
FileCopy "C:\blah.mdb", DestFile   ' Copy file to destination.
Exit Sub

ErrHandler:
If Err = 55 Then                          ' File already open.
        MsgBox "Cannot copy an open file. Close it and try again."
Else
        MsgBox "You must specify a complete destination file name."
End If
Resume Next
End Sub
```

Sub showdist_Click ()

```
distribu.Show
End Sub
```

Sub spr_Click ()

```
If spr.checked = True Then
        spr.checked = False
ElseIf spr.checked = False Then
        spr.checked = True
End If
End Sub
```

Sub start_Click ()

```
runform.Show
End Sub
```

Sub transstart_Click ()

```
Dim db1 As database
Dim db2 As database
Dim t1 As table
Dim t2 As table
Dim i As Integer
Dim j As Integer
Dim k As Integer

Dim a1, a2, a3, a4, a5, a6, a7, a8, a9, a10
Dim a11, a12, a13, a14, a15, a16, a17, a18, a19, a20
Dim a21, a22, a23, a24, a25, a26, a27, a28, a29, a30
Dim a31, a32, a33, a34, a35, a36, a37, a38, a39, a40
Dim a41, a42, a43, a44, a45, a46, a47, a48, a49, a50
Dim a51, a52, a53, a54, a55, a56, a57, a58, a59, a60
Dim a61, a62, a63, a65, a66, a67 a71, a73, a75, a77
Dim a79, a81, a82, a83, a84, a85, a86, a87, a88, a89
Dim a90, a91, a92, a93, a94, a95

Dim b1, b2, b3, b4, b5, b6, b7, b8, b9, b10
Dim b11, b12, b13, b14, b15, b16, b17, b18, b19, b20
Dim b21, b22, b23, b24, b25, b26, b27, b28, b29, b30
Dim b31, b32, b33, b34, b35, b36, b37, b38, b39, b40
Dim b41, b42, b43, b44, b45, b46, b47, b48, b49, b50
Dim b51, b52, b53, b54, b55, b56, b57, b58, b59, b60
Dim b61, b62, b63, b64, b65, b66, b67, b68, b69, b70
Dim b71, b72, b73, b74, b75, b76, b77, b78, b79, b80
Dim b81, b82, b83, b84, b85, b86, b87, b88, b89, b90
Dim b91, b92, b93, b94, b95, b96, b97, b98, b99, b100
Dim b101, b102, b103, b104

Set db1 = OpenDatabase("c:\blah.mdb")
Set t1 = db1.OpenTable("typedescriptor")
        t1.MoveFirst
        TotalCount = t1("number")
t1.Close

Set db2 = OpenDatabase("c:\nextstep.mdb")
For j = 1 To 4
        For i = 0 To TotalCount - 1
        Set t1 = db1.OpenTable(j & i)
                t1.MoveLast

                a1 = t1("st_io_code0")
                a2 = t1("st_io_code1")
                a3 = t1("st_io_code2")
                a4 = t1("st_io_code3")
                a5 = t1("st_io_code4")
                a6 = t1("st_io_code5")
                a7 = t1("st_io_code6")
                a8 = t1("st_io_code7")

                a9 = t1("st_fuz_centre_pos0")
                a11 = t1("st_fuz_centre_pos2")
                a13 = t1("st_fuz_centre_pos4")
                a15 = t1("st_fuz_centre_pos6")

                a17 = t1("st_fuz_sigma0")
                a19 = t1("st_fuz_sigma2")
                a21 = t1("st_fuz_sigma4")
                a23 = t1("st_fuz_sigma6")

                a25 = t1("st_defuz_centre0")
                a26 = t1("st_defuz_centre1")
                a27 = t1("st_defuz_centre2")
                a28 = t1("st_defuz_centre3")

                a29 = t1("st_defuz_sigma0")
                a30 = t1("st_defuz_sigma1")
                a31 = t1("st_defuz_sigma2")
                a32 = t1("st_defuz_sigma3")

                a33 = t1("st_budgetstate0")
                a34 = t1("st_budgetstate1")
```

```
a35 = t1("st_budgetstate2")
a36 = t1("st_budgetstate3")

a37 = t1("st_acttime0")
a38 = t1("st_acttime1")
a39 = t1("st_acttime2")
a40 = t1("st_acttime3")

a41 = t1("st_success00")
a42 = t1("st_success01")
a43 = t1("st_success02")
a44 = t1("st_success03")
a45 = t1("st_success10")
a46 = t1("st_success11")
a47 = t1("st_success12")
a48 = t1("st_success13")
a49 = t1("st_success20")
a50 = t1("st_success21")
a51 = t1("st_success22")
a52 = t1("st_success23")
a53 = t1("st_success30")
a54 = t1("st_success31")
a55 = t1("st_success32")
56 = t1("st_success33")

a57 = t1("CMIOCode0")
a58 = t1("CMIOCode1")

a65 = t1("CM_fuz_centre_pos0")
a67 = t1("CM_fuz_centre_pos2")
a69 = t1("CM_fuz_centre_pos4")
a71 = t1("CM_fuz_centre_pos6")

a73 = t1("CM_fuz_sigma0")
a75 = t1("CM_fuz_sigma2")
a77 = t1("CM_fuz_sigma4")
a79 = t1("CM_fuz_sigma6")

a81 = t1("CM_defuz_centre0")
a82 = t1("CM_defuz_centre1")
a83 = t1("CM_defuz_centre2")
a84 = t1("CM_defuz_centre3")

a85 = t1("CM_defuz_sigma0")
a86 = t1("CM_defuz_sigma1")
a87 = t1("CM_defuz_sigma2")
a88 = t1("CM_defuz_sigma3")

a89 = t1("CM_know0")
a90 = t1("CM_know1")
a91 = t1("CM_know2")
a92 = t1("CM_know3")

a93 = t1("XHome")
a94 = t1("YHome")

a10 = t1("learn_io_code0")
a12 = t1("learn_io_code1")

a14 = t1("learn_fuz_centre_pos2")
a16 = t1("learn_fuz_centre_pos4")
```

```
        a18 = t1("learn_fuz_centre_pos6")
        a20 = t1("learn_fuz_centre_pos8")
        a22 = t1("learn_fuz_centre_pos10")

        a24 = t1("learn_fuz_sigma2")
        a59 = t1("learn_fuz_sigma4")
        a60 = t1("learn_fuz_sigma6")
        a61 = t1("learn_fuz_sigma8")
        a62 = t1("learn_fuz_sigma10")

        a63 = t1("LearnStrategy")
        a95 = t1("Activities")
t1.Close

Set t2 = db2.OpenTable(j & i)
        t2.MoveFirst
        t2.Edit

        t2("st_io_code0") = a1
        t2("st_io_code1") = a2
        t2("st_io_code2") = a3
        t2("st_io_code3") = a4
        t2("st_io_code4") = a5
        t2("st_io_code5") = a6
        t2("st_io_code6") = a7
        t2("st_io_code7") = a8

        t2("st_fuz_centre_pos0") = a9
        t2("st_fuz_centre_pos2") = a11
        t2("st_fuz_centre_pos4") = a13
        t2("st_fuz_centre_pos6") = a15

        t2("st_fuz_sigma0") = a17
        t2("st_fuz_sigma2") = a19
        t2("st_fuz_sigma4") = a21
        t2("st_fuz_sigma6") = a23

        t2("st_defuz_centre0") = a25
        t2("st_defuz_centre1") = a26
        t2("st_defuz_centre2") = a27
        t2("st_defuz_centre3") = a28

        t2("st_defuz_sigma0") = a29
        t2("st_defuz_sigma1") = a30
        t2("st_defuz_sigma2") = a31
        t2("st_defuz_sigma3") = a32

        t2("st_budgetstate0") = a33
        t2("st_budgetstate1") = a34
        t2("st_budgetstate2") = a35
        t2("st_budgetstate3") = a36

        t2("st_acttime0") = a37
        t2("st_acttime1") = a38
        t2("st_acttime2") = a39
        t2("st_acttime3") = a40

        t2("st_success00") = a41
        t2("st_success01") = a42
        t2("st_success02") = a43
        t2("st_success03") = a44
```

```
t2("st_success10") = a45
t2("st_success11") = a46
t2("st_success12") = a47
t2("st_success13") = a48
t2("st_success20") = a49
t2("st_success21") = a50
t2("st_success22") = a51
t2("st_success23") = a52
t2("st_success30") = a53
t2("st_success31") = a54
t2("st_success32") = a55
t2("st_success33") = a56

t2("CMIOCode0") = a57
t2("CMIOCode1") = a58

t2("CM_fuz_centre_pos0") = a65
t2("CM_fuz_centre_pos2") = a67
t2("CM_fuz_centre_pos4") = a69
t2("CM_fuz_centre_pos6") = a71

t2("CM_fuz_sigma0") = a73
t2("CM_fuz_sigma2") = a75
t2("CM_fuz_sigma4") = a77
t2("CM_fuz_sigma6") = a79

t2("CM_defuz_centre0") = a81
t2("CM_defuz_centre1") = a82
t2("CM_defuz_centre2") = a83
t2("CM_defuz_centre3") = a84

t2("CM_defuz_sigma0") = a85
t2("CM_defuz_sigma1") = a86
t2("CM_defuz_sigma2") = a87
t2("CM_defuz_sigma3") = a88

t2("CM_know0") = a89
t2("CM_know1") = a90
t2("CM_know2") = a91
t2("CM_know3") = a92

t2("XHome") = a93
t2("YHome") = a94

t2("learn_io_code0") = a10
t2("learn_io_code1") = a12

t2("learn_fuz_centre_pos2") = a14
t2("learn_fuz_centre_pos4") = a16
t2("learn_fuz_centre_pos6") = a18
t2("learn_fuz_centre_pos8") = a20
t2("learn_fuz_centre_pos10") = a22

t2("learn_fuz_sigma2") = a24
t2("learn_fuz_sigma4") = a59
t2("learn_fuz_sigma6") = a60
t2("learn_fuz_sigma8") = a61
t2("learn_fuz_sigma10") = a62

t2("LearnStrategy") = a63
t2("Activities") = a95
```

```
                t2.Update
        t2.Close
        Next i
Next j

For k = 0 To 8
Set t1 = db1.OpenTable("world" & k)
        t1.MoveLast

        b1 = t1("cap_0")
        b2 = t1("cap_1")
        b3 = t1("cap_2")
        b4 = t1("cap_3")

        b5 = t1("price_0")
        b6 = t1("price_1")
        b7 = t1("price_2")
        b8 = t1("price_3")

        b9 = t1("payoff0_0")
        b10 = t1("payoff0_1")
        b11 = t1("payoff0_2")
        b12 = t1("payoff0_3")
        b13 = t1("payoff0_4")
        b14 = t1("payoff0_5")
        b15 = t1("payoff0_6")
        b16 = t1("payoff0_7")
        b17 = t1("payoff0_8")
        b18 = t1("payoff0_9")
        b19 = t1("payoff0_10")
        b20 = t1("payoff0_11")
        b21 = t1("payoff0_12")
        b22 = t1("payoff0_13")
        b23 = t1("payoff0_14")
        b24 = t1("payoff0_15")
        b25 = t1("payoff0_16")
        b26 = t1("payoff0_17")
        b27 = t1("payoff0_18")
        b28 = t1("payoff0_19")
        b29 = t1("payoff0_20")
        b30 = t1("payoff0_21")
        b31 = t1("payoff0_22")
        b32 = t1("payoff0_23")

        b33 = t1("payoff1_0")
        b34 = t1("payoff1_1")
        b35 = t1("payoff1_2")
        b36 = t1("payoff1_3")
        b37 = t1("payoff1_4")
        b38 = t1("payoff1_5")
        b39 = t1("payoff1_6")
        b40 = t1("payoff1_7")
        b41 = t1("payoff1_8")
        b42 = t1("payoff1_9")
        b43 = t1("payoff1_10")
        b44 = t1("payoff1_11")
        b45 = t1("payoff1_12")
        b46 = t1("payoff1_13")
```

```
b47 = t1("payoff1_14")
b48 = t1("payoff1_15")
b49 = t1("payoff1_16")
b50 = t1("payoff1_17")
b51 = t1("payoff1_18")
b52 = t1("payoff1_19")
b53 = t1("payoff1_20")
b54 = t1("payoff1_21")
b55 = t1("payoff1_22")
b56 = t1("payoff1_23")

b57 = t1("payoff2_0")
b58 = t1("payoff2_1")
b59 = t1("payoff2_2")
b60 = t1("payoff2_3")
b61 = t1("payoff2_4")
b62 = t1("payoff2_5")
b63 = t1("payoff2_6")
b64 = t1("payoff2_7")
b65 = t1("payoff2_8")
b66 = t1("payoff2_9")
b67 = t1("payoff2_10")
b68 = t1("payoff2_11")
b69 = t1("payoff2_12")
b70 = t1("payoff2_13")
b71 = t1("payoff2_14")
b72 = t1("payoff2_15")
b73 = t1("payoff2_16")
b74 = t1("payoff2_17")
b75 = t1("payoff2_18")
b76 = t1("payoff2_19")
b77 = t1("payoff2_20")
b78 = t1("payoff2_21")
b79 = t1("payoff2_22")
b80 = t1("payoff2_23")

b81 = t1("payoff3_0")
b82 = t1("payoff3_1")
b83 = t1("payoff3_2")
b84 = t1("payoff3_3")
b85 = t1("payoff3_4")
b86 = t1("payoff3_5")
b87 = t1("payoff3_6")
b88 = t1("payoff3_7")
b89 = t1("payoff3_8")
b90 = t1("payoff3_9")
b91 = t1("payoff3_10")
b92 = t1("payoff3_11")
b93 = t1("payoff3_12")
b94 = t1("payoff3_13")
b95 = t1("payoff3_14")
b96 = t1("payoff3_15")
b97 = t1("payoff3_16")
b98 = t1("payoff3_17")
b99 = t1("payoff3_18")
b100 = t1("payoff3_19")
b101 = t1("payoff3_20")
```

```
          b102 = t1("payoff3_21")
          b103 = t1("payoff3_22")
          b104 = t1("payoff3_23")

t1.Close
Set t2 = db2.OpenTable("world" & k)
          t2.MoveFirst
          t2.Edit

          t2("cap_0") = b1
          t2("cap_1") = b2
          t2("cap_2") = b3
          t2("cap_3") = b4

          t2("price_0") = b5
          t2("price_1") = b6
          t2("price_2") = b7
          t2("price_3") = b8

          t2("payoff0_0") = b9
          t2("payoff0_1") = b10
          t2("payoff0_2") = b11
          t2("payoff0_3") = b12
          t2("payoff0_4") = b13
          t2("payoff0_5") = b14
          t2("payoff0_6") = b15
          t2("payoff0_7") = b16
          t2("payoff0_8") = b17
          t2("payoff0_9") = b18
          t2("payoff0_10") = b19
          t2("payoff0_11") = b20
          t2("payoff0_12") = b21
          t2("payoff0_13") = b22
          t2("payoff0_14") = b23
          t2("payoff0_15") = b24
          t2("payoff0_16") = b25
          t2("payoff0_17") = b26
          t2("payoff0_18") = b27
          t2("payoff0_19") = b28
          t2("payoff0_20") = b29
          t2("payoff0_21") = b30
          t2("payoff0_22") = b31
          t2("payoff0_23") = b32

          t2("payoff1_0") = b33
          t2("payoff1_1") = b34
          t2("payoff1_2") = b35
          t2("payoff1_3") = b36
          t2("payoff1_4") = b37
          t2("payoff1_5") = b38
          t2("payoff1_6") = b39
          t2("payoff1_7") = b40
          t2("payoff1_8") = b41
          t2("payoff1_9") = b42
          t2("payoff1_10") = b43
          t2("payoff1_11") = b44
          t2("payoff1_12") = b45
          t2("payoff1_13") = b46
```

```
t2("payoff1_14") = b47
t2("payoff1_15") = b48
t2("payoff1_16") = b49
t2("payoff1_17") = b50
t2("payoff1_18") = b51
t2("payoff1_19") = b52
t2("payoff1_20") = b53
t2("payoff1_21") = b54
t2("payoff1_22") = b55
t2("payoff1_23") = b56

t2("payoff2_0") = b57
t2("payoff2_1") = b58
t2("payoff2_2") = b59
t2("payoff2_3") = b60
t2("payoff2_4") = b61
t2("payoff2_5") = b62
t2("payoff2_6") = b63
t2("payoff2_7") = b64
t2("payoff2_8") = b65
t2("payoff2_9") = b66
t2("payoff2_10") = b67
t2("payoff2_11") = b68
t2("payoff2_12") = b69
t2("payoff2_13") = b70
t2("payoff2_14") = b71
t2("payoff2_15") = b72
t2("payoff2_16") = b73
t2("payoff2_17") = b74
t2("payoff2_18") = b75
t2("payoff2_19") = b76
t2("payoff2_20") = b77
t2("payoff2_21") = b78
t2("payoff2_22") = b79
t2("payoff2_23") = b80

t2("payoff3_0") = b81
t2("payoff3_1") = b82
t2("payoff3_2") = b83
t2("payoff3_3") = b84
t2("payoff3_4") = b85
t2("payoff3_5") = b86
t2("payoff3_6") = b87
t2("payoff3_7") = b88
t2("payoff3_8") = b89
t2("payoff3_9") = b90
t2("payoff3_10") = b91
t2("payoff3_11") = b92
t2("payoff3_12") = b93
t2("payoff3_13") = b94
t2("payoff3_14") = b95
t2("payoff3_15") = b96
t2("payoff3_16") = b97
t2("payoff3_17") = b98
t2("payoff3_18") = b99
t2("payoff3_19") = b100
t2("payoff3_20") = b101
```

```
        t2("payoff3_21") = b102
        t2("payoff3_22") = b103
        t2("payoff3_23") = b104

        t2.Update
t2.Close
Next k
db1.Close
db2.Close
End Sub
```

Meangrap.frm

Sub Command1_Click ()

```
Unload MeanGraph
End Sub
```

Sub Command2_Click ()

```
PrintForm
End Sub
```

On_off.frm

```
Option Explicit
```

Sub Combo1_Click ()

```
If Combo1.Text = output1 Then
        test1 = ipt0
        test2 = ipt1
ElseIf Combo1.Text = output2 Then
        test1 = ipt2
        test2 = ipt3
ElseIf Combo1.Text = output3 Then
        test1 = ipt4
        test2 = ipt5
ElseIf Combo1.Text = output4 Then
        test1 = ipt6
        test2 = ipt7
End If

bittest
End Sub
```

Sub Command1_Click ()

```
Unload io_edit
End Sub
```

Sub Command2_Click ()

```
defuzenter.Show
fuzzyenter.Show
End Sub
```

Sub Command3_Click ()

```
Save
```

```
End Sub
```

Sub Option1_DblClick (Index As Integer)

```
If io_edit.Combo1.Text = output1 Then
        ipt0 = (ipt0 Or 2 ^ Index)
ElseIf io_edit.Combo1.Text = output2 Then
        ipt1 = (ipt1 Or 2 ^ Index)
ElseIf io_edit.Combo1.Text = output3 Then
        ipt2 = (ipt2 Or 2 ^ Index)
ElseIf io_edit.Combo1.Text = output4 Then
        ipt3 = (ipt3 Or 2 ^ Index)
End If
End Sub
```

Sub Option2_DblClick (Index As Integer)

```
If option2(Index).Value = True And Combo1.Text = output1 Then
        ipt0 = (ipt0 Xor 2 ^ Index)
End If
If option2(Index).Value = True And Combo1.Text = output2 Then
        ipt1 = (ipt1 Xor 2 ^ Index)
End If
If option2(Index).Value = True And Combo1.Text = output3 Then
        ipt2 = (ipt2 Xor 2 ^ Index)
End If
If option2(Index).Value = True And Combo1.Text = output4 Then
        ipt3 = (ipt3 Xor 2 ^ Index)
End If
End Sub
```

Sub Option3_DblClick (Index As Integer)

```
If io_edit.Combo1.Text = output1 Then
        ipt4 = (ipt4 Or 2 ^ Index)
ElseIf io_edit.Combo1.Text = output2 Then
        ipt5 = (ipt5 Or 2 ^ Index)
ElseIf io_edit.Combo1.Text = output3 Then
        ipt6 = (ipt6 Or 2 ^ Index)
ElseIf io_edit.Combo1.Text = output4 Then
        ipt7 = (ipt7 Or 2 ^ Index)
End If
End Sub
```

Sub Option4_DblClick (Index As Integer)

```
If option2(Index).Value = True And Combo1.Text = output1 Then
        ipt4 = (ipt4 Xor 2 ^ Index)
End If
If option2(Index).Value = True And Combo1.Text = output2 Then
        ipt5 = (ipt5 Xor 2 ^ Index)
End If
If option2(Index).Value = True And Combo1.Text = output3 Then
        ipt6 = (ipt6 Xor 2 ^ Index)
End If
If option2(Index).Value = True And Combo1.Text = output4 Then
        ipt7 = (ipt7 Xor 2 ^ Index)
End If
End Sub
```

Sub Text3_Change ()

```
retrieve
End Sub
```

Payoff.frm

Sub Command1_Click ()

```
Unload payoff
End Sub
```

Sub Command2_Click ()

```
data1.Recordset.Update
End Sub
```

Sub Form_Load ()

```
data1.RecordSource = "world0"
End Sub
```

Sub HScroll1_Change ()

```
data1.RecordSource = "world" & hscroll1.Value
End Sub
```

Prchange.frm

Sub Command1_Click ()

```
PrChange.Hide
End Sub
```

pricefor.frm

Sub Command1_Click ()

```
pricefor.Hide
End Sub
```

Ranparfor.frm

Sub Command1_Click ()

```
ranparfor.Hide
End Sub
```

Reset.frm

Sub Command1_Click ()

```
Hide
End Sub
```

Runform.frm

```
Sub Command1_Click ()

MainLoop
End Sub

Sub Command3_Click ()

RunForm.Hide
End Sub
```

Admin.bas

```
Option Explicit

Type Actor
        tablenumber As Variant
        ShortTermIOCode(7) As Long
        ShortTermBudgets(3) As Single
        ShortTermDefuzCentre(3) As Integer
        ShortTermDefuzSigma(3) As Integer
        ShortTermFuzCentre(7) As Single
        ShortTermFuzSigma(7) As Single
        ShortTermImportance(3) As Single
        ShortTermActivities(3) As Single
        PreviousActivity(1) As Integer          'activity number and hrs. spent
        DailyActivities As String               'documentation string for the day
        CurrentActivity(3) As Integer           'gives current activity and arrival number and
                                                alternative chosen

        RemTime(3) As Single
        XPos As Integer                         'current coordinates
        Ypos As Integer
        FinalDecisionVector(4, 3) As Single     'Utility,Reward,Xpos,Ypos*Activities
        CurrentCell As Integer                  'Number of Current cell position
        XHome As Integer                        'home coordinates
        YHome As Integer
        CMknow(3, 3, 2) As Integer              'activities, alternatives, 2 coordinates + success
                                                count for alternatives

        CmIOCode(7) As Long
        CMDefuzCentre(3) As Integer
        CMDefuzSigma(3) As Integer
        CMFuzCentre(7) As Single
        CMFuzSigma(7) As Single
        Utility(3, 3) As Single                 'activitites, alternatives
        CentreMatrix(299, 3) As Single          'matrix containing the last ten values for the centre
                                                of the st_fuzzy sets to calculate moving average

        MutationTag As Integer
        LearnIOCode(7) As Long
        LearnSigma(2 To 11) As Single
        LearnCentre(2 To 11) As Single
        Age As Integer
        LearnS As Integer
End Type

Global Const NumOfActors = 16
Dim Agent1() As Actor
Dim Agent2() As Actor
Dim Agent3() As Actor
Dim Agent4() As Actor
```

```
Global duration As Integer
Global TotalCount As Integer
Global RobotNum As Integer
Global Actinum As Integer
Global ItNum As Integer
Global importance(3) As Single
Global Factor(3) As Single
Global WriteNum As Integer
Global LearnAlt(3) As Single
Global TTC As Integer
Global MuteAlt As Integer
Global CopyProb As Single
Global AndOrProb As Single
Global Capfac As Single
Global CMASize As Integer
Global MutRate As Integer
Global LThres As Single
Global RepFil As Single

Global db As database
Dim number As Integer
Global RandArray() As Single
Global Resetvalue As Single
```

Sub ForwardValues (Robot() As Actor)

```
'this is supposed to decode IO Values and to write all stuff to
'the fuzzy rule base fuzzrule.bas

Dim k As Integer
Dim l As Integer
Dim test1 As Double
Dim test2 As Double

test1 = 0
test2 = 0
If Actinum = 0 Then
        test1 = Robot(RobotNum).ShortTermIOCode(0)
        test2 = Robot(RobotNum).ShortTermIOCode(1)
ElseIf Actinum = 1 Then
        test1 = Robot(RobotNum).ShortTermIOCode(2)
        test2 = Robot(RobotNum).ShortTermIOCode(3)
ElseIf Actinum = 2 Then
        test1 = Robot(RobotNum).ShortTermIOCode(4)
        test2 = Robot(RobotNum).ShortTermIOCode(5)
ElseIf Actinum = 3 Then
        test1 = Robot(RobotNum).ShortTermIOCode(6)
        test2 = Robot(RobotNum).ShortTermIOCode(7)
End If

For k = 0 To 15
        If (test1 And 2 ^ k) = 2 ^ k Then
                ioarray(Actinum, k) = 1
        Else ioarray(Actinum, k) = 0
        End If
Next k

For l = 0 To 15
        If (test2 And 2 ^ l) = 2 ^ l Then
```

```
                ioarray(Actinum, l + 16) = 1
        Else ioarray(Actinum, l + 16) = 0
        End If
Next l
End Sub
```

Sub LoadValues ()

```
'loads initial Conditions from Blah.MDB

Dim t1 As table
Dim t2 As table
Dim t3 As table
Dim t4 As table

Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim m As Integer
Dim n As Integer
Dim o As Integer
Dim p As Integer
Dim q As Integer
Dim r As Integer
Dim s As Integer
Dim u As Integer
Dim v As Integer
Dim cnt As Integer
Dim cnt2 As Integer
Dim Nextcnt As Integer
Dim interim As Integer
Dim cellnumber As Integer

Set t1 = db.OpenTable("typedescriptor")
        t1.MoveFirst
        TotalCount = t1("number")
        For i = 0 To TotalCount - 1
                Agent1(i).tablenumber = i
                Agent2(i).tablenumber = i
                Agent3(i).tablenumber = i
                Agent4(i).tablenumber = i
        Next i
t1.Close

For RobotNum = 0 To TotalCount - 1              'loop over number of robots of type
        Set t2 = db.OpenTable("1" & Agent1(RobotNum).tablenumber)
        t2.MoveFirst
        For k = 0 To 7
                Agent1(RobotNum).ShortTermIOCode(k) = t2("st_io_code" & k)
                Agent1(RobotNum).CmIOCode(k) = t2("CMIOCode" & k Mod 2)
                Agent1(RobotNum).LearnIOCode(k) = t2("learn_io_code" & k Mod 2)
        Next k
        For l = 0 To 3
                Agent1(RobotNum).ShortTermFuzCentre(2 * l) = t2("st_fuz_centre_pos" & 2 * l)
                Agent1(RobotNum).ShortTermFuzCentre(2 * l + 1) = t2("st_fuz_centre_pos" & 2 * l)
                Agent1(RobotNum).ShortTermFuzSigma(2 * l) = t2("st_fuz_sigma" & 2 * l)
                Agent1(RobotNum).ShortTermFuzSigma(2 * l + 1) = t2("st_fuz_sigma" & 2 * l)
                Agent1(RobotNum).CMFuzCentre(2 * l) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent1(RobotNum).CMFuzCentre(2 * l + 1) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent1(RobotNum).CMFuzSigma(2 * l) = t2("CM_fuz_sigma" & 2 * l)
        Agent1(RobotNum).CMFuzSigma(2 * l + 1) = t2("CM_fuz_sigma" & 2 * l)

        Agent1(RobotNum).ShortTermBudgets(l) = t2("st_budgetstate" & l)
        Agent1(RobotNum).ShortTermDefuzCentre(l) = t2("st_defuz_centre" & l)
        Agent1(RobotNum).ShortTermDefuzSigma(l) = t2("st_defuz_sigma" & l)
        Agent1(RobotNum).CMDefuzCentre(l) = t2("CM_defuz_centre" & l)
        Agent1(RobotNum).CMDefuzSigma(l) = t2("CM_defuz_sigma" & l)
Next l
For v = 1 To 5
        Agent1(RobotNum).LearnCentre(2 * v) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent1(RobotNum).LearnCentre(2 * v + 1) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent1(RobotNum).LearnSigma(2 * v) = t2("learn_fuz_sigma" & 2 * v)
        Agent1(RobotNum).LearnSigma(2 * v + 1) = t2("learn_fuz_sigma" & 2 * v)
Next v
For j = 0 To 3
        For o = 0 To 3
                For q = 0 To 1
                        Agent1(RobotNum).CMknow(j, o, q) = gridfactor *
                        Mid(t2("CM_know" & j), (2 * o) + q + 1, 1)
                Next q
                Agent1(RobotNum).CMknow(j, o, 2) = t2("st_success" & j & o)
                Next o
Next j

Agent1(RobotNum).XPos = t2("XHome")
Agent1(RobotNum).Ypos = t2("YHome")
Agent1(RobotNum).XHome = t2("XHome")
Agent1(RobotNum).YHome = t2("YHome")
Agent1(RobotNum).Age = 0

Select Case startform.spr.Checked
Case True
        Agent1(RobotNum).LearnS = t2("LearnStrategy")
Case Else
        Agent1(RobotNum).LearnS = 0 'Int(4 * Rnd)
End Select
t2.Close

Set t2 = db.OpenTable("2" & Agent2(RobotNum).tablenumber)
t2.MoveFirst
For k = 0 To 7
        Agent2(RobotNum).ShortTermIOCode(k) = t2("st_io_code" & k)
        Agent2(RobotNum).CmIOCode(k) = t2("CMIOCode" & k Mod 2)
        Agent2(RobotNum).LearnIOCode(k) = t2("learn_io_code" & k Mod 2)
Next k
For l = 0 To 3
        Agent2(RobotNum).ShortTermFuzCentre(2 * l) = t2("st_fuz_centre_pos" & 2 * l)
        Agent2(RobotNum).ShortTermFuzCentre(2 * l + 1) = t2("st_fuz_centre_pos" & 2 * l)
        Agent2(RobotNum).ShortTermFuzSigma(2 * l) = t2("st_fuz_sigma" & 2 * l)
        Agent2(RobotNum).ShortTermFuzSigma(2 * l + 1) = t2("st_fuz_sigma" & 2 * l)
        Agent2(RobotNum).CMFuzCentre(2 * l) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent2(RobotNum).CMFuzCentre(2 * l + 1) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent2(RobotNum).CMFuzSigma(2 * l) = t2("CM_fuz_sigma" & 2 * l)
        Agent2(RobotNum).CMFuzSigma(2 * l + 1) = t2("CM_fuz_sigma" & 2 * l)

        Agent2(RobotNum).ShortTermBudgets(l) = t2("st_budgetstate" & l)
        Agent2(RobotNum).ShortTermDefuzCentre(l) = t2("st_defuz_centre" & l)
        Agent2(RobotNum).ShortTermDefuzSigma(l) = t2("st_defuz_sigma" & l)
        Agent2(RobotNum).CMDefuzCentre(l) = t2("CM_defuz_centre" & l)
        Agent2(RobotNum).CMDefuzSigma(l) = t2("CM_defuz_sigma" & l)
Next l
For v = 1 To 5
        Agent2(RobotNum).LearnCentre(2 * v) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent2(RobotNum).LearnCentre(2 * v + 1) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent2(RobotNum).LearnSigma(2 * v) = t2("learn_fuz_sigma" & 2 * v)
        Agent2(RobotNum).LearnSigma(2 * v + 1) = t2("learn_fuz_sigma" & 2 * v)
Next v
For j = 0 To 3
        For o = 0 To 3
                For q = 0 To 1
                        Agent2(RobotNum).CMknow(j, o, q) = gridfactor *
                        Mid(t2("CM_know" & j), (2 * o) + q + 1, 1)
                Next q
                Agent2(RobotNum).CMknow(j, o, 2) = 0
        Next o
Next j

Agent2(RobotNum).XPos = t2("XHome")
Agent2(RobotNum).Ypos = t2("YHome")
Agent2(RobotNum).XHome = t2("XHome")
Agent2(RobotNum).YHome = t2("YHome")
Agent2(RobotNum).Age = 0

Select Case startform.spr.Checked
Case True
        Agent2(RobotNum).LearnS = t2("LearnStrategy")
Case Else
        Agent2(RobotNum).LearnS = 0 'Int(4 * Rnd)
End Select
t2.Close

Set t2 = db.OpenTable("3" & Agent3(RobotNum).tablenumber)
t2.MoveFirst
For k = 0 To 7
        Agent3(RobotNum).ShortTermIOCode(k) = t2("st_io_code" & k)
        Agent3(RobotNum).CmIOCode(k) = t2("CMIOCode" & k Mod 2)
        Agent3(RobotNum).LearnIOCode(k) = t2("learn_io_code" & k Mod 2)
Next k
For l = 0 To 3
        Agent3(RobotNum).ShortTermFuzCentre(2 * l) = t2("st_fuz_centre_pos" & 2 * l)
        Agent3(RobotNum).ShortTermFuzCentre(2 * l + 1) = t2("st_fuz_centre_pos" & 2 * l)
        Agent3(RobotNum).ShortTermFuzSigma(2 * l) = t2("st_fuz_sigma" & 2 * l)
        Agent3(RobotNum).ShortTermFuzSigma(2 * l + 1) = t2("st_fuz_sigma" & 2 * l)
        Agent3(RobotNum).CMFuzCentre(2 * l) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent3(RobotNum).CMFuzCentre(2 * l + 1) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent3(RobotNum).CMFuzSigma(2 * l) = t2("CM_fuz_sigma" & 2 * l)
        Agent3(RobotNum).CMFuzSigma(2 * l + 1) = t2("CM_fuz_sigma" & 2 * l)

        Agent3(RobotNum).ShortTermBudgets(l) = t2("st_budgetstate" & l)
        Agent3(RobotNum).ShortTermDefuzCentre(l) = t2("st_defuz_centre" & l)
        Agent3(RobotNum).ShortTermDefuzSigma(l) = t2("st_defuz_sigma" & l)
        Agent3(RobotNum).CMDefuzCentre(l) = t2("CM_defuz_centre" & l)
        Agent3(RobotNum).CMDefuzSigma(l) = t2("CM_defuz_sigma" & l)
Next l
```

```
For v = 1 To 5
        Agent3(RobotNum).LearnCentre(2 * v) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent3(RobotNum).LearnCentre(2 * v + 1) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent3(RobotNum).LearnSigma(2 * v) = t2("learn_fuz_sigma" & 2 * v)
        Agent3(RobotNum).LearnSigma(2 * v + 1) = t2("learn_fuz_sigma" & 2 * v)
Next v
For j = 0 To 3
        For o = 0 To 3
                For q = 0 To 1
                        Agent3(RobotNum).CMknow(j, o, q) = gridfactor *
                        Mid(t2("CM_know" & j), (2 * o) + q + 1, 1)
                Next q
                Agent3(RobotNum).CMknow(j, o, 2) = 0
        Next o
Next j

Agent3(RobotNum).XPos = t2("XHome")
Agent3(RobotNum).Ypos = t2("YHome")
Agent3(RobotNum).XHome = t2("XHome")
Agent3(RobotNum).YHome = t2("YHome")
Agent3(RobotNum).Age = 0

Select Case startform.spr.Checked
Case True
        Agent3(RobotNum).LearnS = t2("LearnStrategy")
Case Else
        Agent3(RobotNum).LearnS = 0 'Int(4 * Rnd)
End Select
t2.Close

Set t2 = db.OpenTable("4" & Agent4(RobotNum).tablenumber)
t2.MoveFirst
For k = 0 To 7
        Agent4(RobotNum).ShortTermIOCode(k) = t2("st_io_code" & k)
        Agent4(RobotNum).CmIOCode(k) = t2("CMIOCode" & k Mod 2)
        Agent4(RobotNum).LearnIOCode(k) = t2("learn_io_code" & k Mod 2)
Next k
For l = 0 To 3
        Agent4(RobotNum).ShortTermFuzCentre(2 * l) = t2("st_fuz_centre_pos" & 2 * l)
        Agent4(RobotNum).ShortTermFuzCentre(2 * l + 1) = t2("st_fuz_centre_pos" & 2 * l)
        Agent4(RobotNum).ShortTermFuzSigma(2 * l) = t2("st_fuz_sigma" & 2 * l)
        Agent4(RobotNum).ShortTermFuzSigma(2 * l + 1) = t2("st_fuz_sigma" & 2 * l)
        Agent4(RobotNum).CMFuzCentre(2 * l) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent4(RobotNum).CMFuzCentre(2 * l + 1) = t2("CM_fuz_centre_pos" & 2 * l)
        Agent4(RobotNum).CMFuzSigma(2 * l) = t2("CM_fuz_sigma" & 2 * l)
        Agent4(RobotNum).CMFuzSigma(2 * l + 1) = t2("CM_fuz_sigma" & 2 * l)

        Agent4(RobotNum).ShortTermBudgets(l) = t2("st_budgetstate" & l)
        Agent4(RobotNum).ShortTermDefuzCentre(l) = t2("st_defuz_centre" & l)
        Agent4(RobotNum).ShortTermDefuzSigma(l) = t2("st_defuz_sigma" & l)
        Agent4(RobotNum).CMDefuzCentre(l) = t2("CM_defuz_centre" & l)
        Agent4(RobotNum).CMDefuzSigma(l) = t2("CM_defuz_sigma" & l)
Next l
For v = 1 To 5
        Agent4(RobotNum).LearnCentre(2 * v) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent4(RobotNum).LearnCentre(2 * v + 1) = t2("learn_fuz_centre_pos" & 2 * v)
        Agent4(RobotNum).LearnSigma(2 * v) = t2("learn_fuz_sigma" & 2 * v)
        Agent4(RobotNum).LearnSigma(2 * v + 1) = t2("learn_fuz_sigma" & 2 * v)
        Next v
        For j = 0 To 3
                For o = 0 To 3
                        For q = 0 To 1
                                Agent4(RobotNum).CMknow(j, o, q) = gridfactor *
                                Mid(t2("CM_know" & j), (2 * o) + q + 1, 1)
                        Next q
                        Agent4(RobotNum).CMknow(j, o, 2) = 0
                Next o
        Next j

        Agent4(RobotNum).XPos = t2("XHome")
        Agent4(RobotNum).Ypos = t2("YHome")
        Agent4(RobotNum).XHome = t2("XHome")
        Agent4(RobotNum).YHome = t2("YHome")
        Agent4(RobotNum).Age = 0

        Select Case startform.spr.Checked
        Case True
                Agent4(RobotNum).LearnS = t2("LearnStrategy")
        Case Else
                Agent4(RobotNum).LearnS = 0 'Int(4 * Rnd)
        End Select
        t2.Close

        For r = 0 To 299
                For s = 0 To 3
                        Agent1(RobotNum).CentreMatrix(r, s) = (Agent1(RobotNum).
                        ShortTermFuzCentre(2 * s) + Agent1(RobotNum).ShortTermFuzCentre(2 * s + 1))
                / 2
                        Agent2(RobotNum).CentreMatrix(r, s) =  (Agent2(RobotNum).
                        ShortTermFuzCentre(2 * s) + Agent2(RobotNum).ShortTermFuzCentre(2 * s + 1))
                / 2
                        Agent3(RobotNum).CentreMatrix(r, s) =  (Agent3(RobotNum).
                        ShortTermFuzCentre(2 * s) + Agent3(RobotNum).ShortTermFuzCentre(2 * s + 1))
                / 2
                        Agent4(RobotNum).CentreMatrix(r, s) = (Agent4(RobotNum).
                        ShortTermFuzCentre(2 * s) + Agent4(RobotNum).ShortTermFuzCentre(2 * s + 1))
                / 2
                Next s
        Next r
Next RobotNum

Capfac = CaChange.Text3.Text
CMASize = Ranparfor.Text4.Text
MutRate = Ranparfor.Text3.Text
RepFil = Ranparfor.Text6.Text

Set t3 = db.OpenTable("world")
n = 0
t3.MoveFirst
Do Until t3.EOF
        ThisWorld.XPos(n) = t3("x_pos")
        ThisWorld.Ypos(n) = t3("y_pos")
        n = n + 1
        t3.MoveNext
Loop
t3.MoveLast
```

```
cellnumber = t3("count")
ThisWorld.tablenumber = t3("count")
t3.Close
For u = 0 To cellnumber - 1
        Set t4 = db.OpenTable("world" & u)
        t4.MoveFirst
        For m = 3 To 0 Step -1
                Select Case m
                Case 3
                        ThisWorld.Capacity(m, u) = t4("cap_" & m)
                        ThisWorld.MaxCapacity(m, u) = capaform.Text1(m).Text
                Case 2
                        ThisWorld.Capacity(m, u) = t4("cap_" & m)
                        ThisWorld.MaxCapacity(m, u) = capaform.Text1(m).Text
                Case 1
                        ThisWorld.Capacity(m, u) = Int(Capfac * Sqr(ThisWorld.Capacity(2, u))) +
                Int(Capfac * Sqr(ThisWorld.Capacity(3, u)))
                        ThisWorld.MaxCapacity(m, u) = Int(Capfac * Sqr(ThisWorld.MaxCapacity(2, u)))
                + Int(Capfac * Sqr(ThisWorld.MaxCapacity(3, u)))
                Case 0
                        ThisWorld.Capacity(m, u) = t4("cap_" & m)
                        ThisWorld.MaxCapacity(m, u) = capaform.Text1(m).Text
                End Select
                ThisWorld.Price(m, u) = t4("price_" & m)
                ThisWorld.MaxPrice(m, u) = pricefor.Text2(m).Text
                BasePrice(m, u) = pricefor.Text1(m).Text
        Next m
        For m = 0 To 23
                ThisWorld.Act(0, m, u) = t4("payoff0_" & m)
                ThisWorld.Act(1, m, u) = t4("payoff1_" & m)
                ThisWorld.Act(2, m, u) = t4("payoff2_" & m)
                ThisWorld.Act(3, m, u) = t4("payoff3_" & m)
        Next m
        t4.Close
Next u
For cnt = 0 To 3                        'preset long term average demand for change of capacity
        For cnt2 = 0 To CellNos - 1
                For Nextcnt = 0 To 299
                        LtAvDemand(cnt, cnt2, Nextcnt) = ((-Log((ThisWorld.MaxPrice(cnt, cnt2) *
                        BasePrice(cnt, cnt2)) / (ThisWorld.Price(cnt, cnt2)) - 1) / CaChange.Text2.Text) +
                CaChange.Text1.Text)
                Next Nextcnt
        Next cnt2
Next cnt
For cnt = 0 To CMASize - 1
        For cnt2 = 0 To 7
                ThisWorld.CommonRules(cnt, cnt2) = Int(Rnd * 65536)
        Next cnt2
Next cnt
Resetvalue = ResetForm.Text1.Text
AndOrProb = Ranparfor.Text1.Text
CopyProb = Ranparfor.Text2.Text
LThres = Ranparfor.Text5.Text
End Sub
```

```
Sub MainLoop ()
'this defines the number of iterations

Dim msg As String
Dim k As Integer
Dim l As Integer
Dim n As Integer

duration = runform.Text1.Text
Set db = OpenDatabase("C:\blah.mdb", True)
ReDim Agent1(NumOfActors)  As Actor
ReDim Agent2(NumOfActors)  As Actor
ReDim Agent3(NumOfActors)  As Actor
ReDim Agent4(NumOfActors)  As Actor

Select Case runform.Check1(6).Value
Case 1
        Randomize
End Select

LoadValues
inform.Show
WriteNum = 0

ReDim Preserve Agent1(TotalCount - 1) As Actor
ReDim Preserve Agent2(TotalCount - 1) As Actor
ReDim Preserve Agent3(TotalCount - 1) As Actor
ReDim Preserve Agent4(TotalCount - 1) As Actor

For ItNum = 0 To duration - 1
        inform.Label1.Caption = "Currently working, Iteration " & (ItNum + 1)
        inform.Label1.Refresh
        For RobotNum = 0 To TotalCount - 1                    ' loop over all robots
                ResetAgent Agent1()
                Select Case Agent1(RobotNum).LearnS
                Case Is <> 0
                        Select Case ItNum Mod MutRate = 0
                        Case True
                                Select Case Agent1(RobotNum).LearnS
                                Case 2
                                        RndChngAgent Agent1()
                                End Select
                                Select Case Agent1(RobotNum).LearnS
                                Case 3
                                        RndChngAgent1 Agent1()
                                End Select
                                Select Case Agent1(RobotNum).LearnS
                                Case 1
                                For Actinum = 0 To 3             'loop over budgets
                                        DecodeLearnRules Agent1()
                                                For MuteAlt = 0 To 3       'loop over other budgets
                                                        Select Case MuteAlt
                                                        Case Is <> Actinum
                                                                GetLearnValues Agent1()
                                                                LearnFuzzify
                                                                LearnDefuzzify
                                                                Mutate Agent1()
                                                        Case Else
                                                        End Select
                                Next MuteAlt
                        Next Actinum
                        End Select
                End Select
        End Select
        For Actinum = 0 To 3'                   loop over activities to load Rule Code
                ForwardValues Agent1()
        Next Actinum
        For Actinum = 0 To 3
                GetValues Agent1()
                Fuzzify
                Defuzzify
                Agent1(RobotNum).ShortTermImportance(Actinum) = importance(Actinum)
        Next Actinum
        Activity Agent1()

        ResetAgent Agent2()
        Select Case Agent2(RobotNum).LearnS
        Case Is <> 0
                Select Case ItNum Mod MutRate = 0
                Case True
                Select Case Agent2(RobotNum).LearnS
                Case 2
                RndChngAgent Agent2()
        End Select
        Select Case Agent2(RobotNum).LearnS
        Case 3
                RndChngAgent1 Agent2()
        End Select
        Select Case Agent2(RobotNum).LearnS
        Case 1
                For Actinum = 0 To 3                    'loop over budgets
                        DecodeLearnRules Agent2()
                        For MuteAlt = 0 To 3            'loop over other budgets
                                Select Case MuteAlt
                                Case Is <> Actinum
                                        GetLearnValues Agent2()
                                        LearnFuzzify
                                        LearnDefuzzify
                                        Mutate Agent2()
                                Case Else
                                End Select
                        Next MuteAlt
                Next Actinum
        End Select
End Select
End Select
For Actinum = 0 To 3                            'loop over activities
        ForwardValues Agent2()
Next Actinum
For Actinum = 0 To 3
        GetValues Agent2()
        Fuzzify
        Defuzzify
        Agent2(RobotNum).ShortTermImportance(Actinum) = importance(Actinum)
Next Actinum
```

```
Activity Agent2()

ResetAgent Agent3()
Select Case Agent3(RobotNum).LearnS
Case Is <> 0
        Select Case ItNum Mod MutRate = 0
        Case True
                Select Case Agent3(RobotNum).LearnS
                Case 2
                        RndChngAgent Agent3()
                End Select
                Select Case Agent3(RobotNum).LearnS
                Case 3
                        RndChngAgent1 Agent3()
                End Select
                Select Case Agent3(RobotNum).LearnS
                Case 1
                        For Actinum = 0 To 3                'loop over budgets
                                DecodeLearnRules Agent3()
                                For MuteAlt = 0 To 3        'loop over other budgets
                                        Select Case MuteAlt
                                        Case Is <> Actinum
                                                GetLearnValues Agent3()
                                                LearnFuzzify
                                                LearnDefuzzify
                                                Mutate Agent3()
                                        Case Else
                                        End Select
                                Next MuteAlt
                        Next Actinum
                End Select
        End Select
End Select
For Actinum = 0 To 3                        'loop over activities
        ForwardValues Agent3()
Next Actinum
For Actinum = 0 To 3
        GetValues Agent3()
        Fuzzify
        Defuzzify
        Agent3(RobotNum).ShortTermImportance(Actinum) = importance(Actinum)
Next Actinum
Activity Agent3()

ResetAgent Agent4()
Select Case Agent4(RobotNum).LearnS
Case Is <> 0
        Select Case ItNum Mod MutRate = 0
        Case True
                Select Case Agent4(RobotNum).LearnS
                Case 3
                        RndChngAgent1 Agent4()
                End Select
                Select Case Agent4(RobotNum).LearnS
                Case 2
                        RndChngAgent Agent4()
                End Select
                Select Case Agent4(RobotNum).LearnS
                Case 1
                For Actinum = 0 To 3                    'loop over budgets
                        DecodeLearnRules Agent4()
                        For MuteAlt = 0 To 3                    'loop over other budgets
                                Select Case MuteAlt
                                Case Is <> Actinum
                                        GetLearnValues Agent4()
                                        LearnFuzzify
                                        LearnDefuzzify
                                        Mutate Agent4()
                                Case Else
                                End Select
                        Next MuteAlt
                Next Actinum
                End Select
        End Select
End Select
For Actinum = 0 To 3                    'loop over activities
        ForwardValues Agent4()
Next Actinum
For Actinum = 0 To 3
        GetValues Agent4()
        Fuzzify
        Defuzzify
        Agent4(RobotNum).ShortTermImportance(Actinum) = importance(Actinum)
Next Actinum
Activity Agent4()
Next RobotNum

For TimeOfDay = 0 To 23                 'loop over the hours of the day
        For k = 0 To 3
                For l = 0 To CellNos - 1
                        CellCapacityCount(k, l, 0) = ThisWorld.Capacity(k, l)
                        CellCapacityCount(k, l, 1) = 0
                Next l
        Next k
        For RobotCount = 0 To TotalCount - 1            'loop over robots

                DecodeRules Agent1()
                FeedValues Agent1()
                ResetFDV Agent1()
                For ActCount = 0 To 3                   'loop over activitites
                        CalcParameters Agent1()
                Next ActCount

                DecodeRules Agent2()
                FeedValues Agent2()
                ResetFDV Agent2()
                For ActCount = 0 To 3                   'loop over activitites
                        CalcParameters Agent2()
                Next ActCount

                DecodeRules Agent3()
                FeedValues Agent3()
                ResetFDV Agent3()
                For ActCount = 0 To 3                   'loop over activitites
                        CalcParameters Agent3()
                Next ActCount

                DecodeRules Agent4()
                FeedValues Agent4()
                ResetFDV Agent4()
                For ActCount = 0 To 3  'loop over activitites
                        CalcParameters Agent4()
                Next ActCount
        Next RobotCount

        SortRandArray
        For TTC = 0 To (4 * TotalCount) - 1
                Select Case Int(RandArray(1, TTC) / TotalCount) = 0
                Case True
                        SelAct Agent1()
                End Select
                Select Case Int(RandArray(1, TTC) / TotalCount) = 1
                Case True
                        SelAct Agent2()
                End Select
                Select Case Int(RandArray(1, TTC) / TotalCount) = 2
                Case True
                        SelAct Agent3()
                End Select
                Select Case Int(RandArray(1, TTC) / TotalCount) = 3
                Case True
                        SelAct Agent4()
                End Select
        Next TTC

        For RobotCount = 0 To TotalCount - 1                    'loop over agents
                GetSomething Agent1()
                GetSomething Agent2()
                GetSomething Agent3()
                GetSomething Agent4()
        Next RobotCount
        GetTurnover
Next TimeOfDay

For RobotCount = 0 To TotalCount - 1
        Select Case runform.Check1(0).Value
        Case 1
                MoveAvg Agent1()
                MoveAvg Agent2()
                MoveAvg Agent3()
                MoveAvg Agent4()
        End Select
        Agent1(RobotCount).Age = Agent1(RobotCount).Age + 1
        Agent2(RobotCount).Age = Agent2(RobotCount).Age + 1
        Agent3(RobotCount).Age = Agent3(RobotCount).Age + 1
        Agent4(RobotCount).Age = Agent4(RobotCount).Age + 1
Next RobotCount

For RobotNum = 0 To TotalCount - 1
        WriteValues
Next RobotNum
For number = 0 To ThisWorld.tablenumber - 1
        WriteWorld
        Next number
        GetDemand
        Select Case runform.Check1(3).Value
        Case 1
                ChangePrice
        End Select
        Select Case runform.Check1(4).Value
        Case 1
                ChangeCap
        End Select
        WriteNum = WriteNum + 1
        If WriteNum > 299 Then
                WriteNum = WriteNum - 300
        End If
Next ItNum
db.Close
inform.Hide
msg = "FINISHED !"
MsgBox msg, 64, "Simulation Finished"
End Sub
```

Sub RanCogInit ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim TypeNum As Integer
Dim i As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.MoveFirst
        TotalCount = t1("number")
t1.Close
For RobotNum = 0 To TotalCount - 1                'loop over number of robots of type
        For TypeNum = 1 To 4
                Set t2 = db.OpenTable(TypeNum & RobotNum)
                        t2.MoveFirst
                        t2.Edit
                        For i = 0 To 3
                                t2("CM_know" & i) = Int(3 * Rnd) & Int(3 * Rnd) & Int(3 * Rnd) &
                        Int(3 * Rnd) & Int(3 * Rnd) & Int(3 * Rnd) & Int(3 * Rnd) & Int(3 * Rnd)
        & Int(3 * Rnd)
                        Next i
                        t2.Update
                        t2.Close
        Next TypeNum
Next RobotNum
db.Close
End Sub
```

Sub RandomInit ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim TypeNum As Integer
Dim i As Integer
Dim k As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.MoveFirst
        TotalCount = t1("number")
t1.Close
For RobotNum = 0 To TotalCount - 1              'loop over number of robots of type
        For TypeNum = 1 To 4
                Set t2 = db.OpenTable(TypeNum & RobotNum)
                        t2.MoveFirst
                        t2.Edit
                        For k = 0 To 7
                                t2("st_io_code" & k) = Int(Rnd * 32768)
                        Next k
                t2.Update
                t2.Close
        Next TypeNum
Next RobotNum
db.Close
End Sub
```

Sub SortRandArray ()

```
Dim i As Integer
Dim IR As Integer
Dim j As Integer
Dim l As Integer
Dim k As Integer
Dim RRA As Single

ReDim RandArray(1, 4 * TotalCount - 1)
For k = 0 To (4 * TotalCount) - 1
        RandArray(0, k) = Rnd
        RandArray(1, k) = k
Next k
l = Int((4 * TotalCount - 1) / 2) + 1
IR = 4 * TotalCount - 1
Do
        If l > 1 Then
                l = l - 1
                RRA = RandArray(0, l)
        Else
                RRA = RandArray(0, IR)
                RandArray(0, IR) = RandArray(0, 1)
                IR = IR - 1
                If IR = 1 Then
                        RandArray(0, 1) = RRA
                        Exit Sub
                End If
        End If
        i = l
        j = l + l
        While j <= IR
                If j < IR Then
                If RandArray(0, j) < RandArray(0, j + 1) Then
```

```
            j = j + 1
        End If
        If RRA < RandArray(0, j) Then
            RandArray(0, i) = RandArray(0, j)
            i = j
            j = j + j
        Else
            j = IR + 1
        End If
    Wend
    RandArray(0, i) = RRA
Loop
End Sub
```

Sub WriteValues ()

```
' supposed to write the results to database

Dim t1 As table
Dim i As Integer

Set t1 = db.OpenTable("1" & Agent1(RobotNum).tablenumber)
    t1.AddNew
    If ((duration - 1 > ItNum) And (Agent1(RobotNum).MutationTag <> 1)) Then
        t1("st_budgetstate0") = Agent1(RobotNum).ShortTermBudgets(0)
        t1("st_budgetstate1") = Agent1(RobotNum).ShortTermBudgets(1)
        t1("st_budgetstate2") = Agent1(RobotNum).ShortTermBudgets(2)
        t1("st_budgetstate3") = Agent1(RobotNum).ShortTermBudgets(3)

        t1("LearnStrategy") = Agent1(RobotNum).LearnS
        t1("Activities") = Agent1(RobotNum).DailyActivities
    Else
        t1("st_io_code0") = Agent1(RobotNum).ShortTermIOCode(0)
        t1("st_io_code1") = Agent1(RobotNum).ShortTermIOCode(1)
        t1("st_io_code2") = Agent1(RobotNum).ShortTermIOCode(2)
        t1("st_io_code3") = Agent1(RobotNum).ShortTermIOCode(3)
        t1("st_io_code4") = Agent1(RobotNum).ShortTermIOCode(4)
        t1("st_io_code5") = Agent1(RobotNum).ShortTermIOCode(5)
        t1("st_io_code6") = Agent1(RobotNum).ShortTermIOCode(6)
        t1("st_io_code7") = Agent1(RobotNum).ShortTermIOCode(7)

        t1("st_fuz_centre_pos0") = Agent1(RobotNum).ShortTermFuzCentre(0)
        t1("st_fuz_centre_pos2") = Agent1(RobotNum).ShortTermFuzCentre(2)
        t1("st_fuz_centre_pos4") = Agent1(RobotNum).ShortTermFuzCentre(4)
        t1("st_fuz_centre_pos6") = Agent1(RobotNum).ShortTermFuzCentre(6)

        t1("st_fuz_sigma0") = Agent1(RobotNum).ShortTermFuzSigma(0)
        t1("st_fuz_sigma2") = Agent1(RobotNum).ShortTermFuzSigma(2)
        t1("st_fuz_sigma4") = Agent1(RobotNum).ShortTermFuzSigma(4)
        t1("st_fuz_sigma6") = Agent1(RobotNum).ShortTermFuzSigma(6)

        t1("st_defuz_centre0") = Agent1(RobotNum).ShortTermDefuzCentre(0)
        t1("st_defuz_centre1") = Agent1(RobotNum).ShortTermDefuzCentre(1)
        t1("st_defuz_centre2") = Agent1(RobotNum).ShortTermDefuzCentre(2)
        t1("st_defuz_centre3") = Agent1(RobotNum).ShortTermDefuzCentre(3)

        t1("st_defuz_sigma0") = Agent1(RobotNum).ShortTermDefuzSigma(0)
        t1("st_defuz_sigma1") = Agent1(RobotNum).ShortTermDefuzSigma(1)
        t1("st_defuz_sigma2") = Agent1(RobotNum).ShortTermDefuzSigma(2)
        t1("st_defuz_sigma3") = Agent1(RobotNum).ShortTermDefuzSigma(3)
        t1("st_budgetstate0") = Agent1(RobotNum).ShortTermBudgets(0)
        t1("st_budgetstate1") = Agent1(RobotNum).ShortTermBudgets(1)
        t1("st_budgetstate2") = Agent1(RobotNum).ShortTermBudgets(2)
        t1("st_budgetstate3") = Agent1(RobotNum).ShortTermBudgets(3)

        t1("st_acttime0") = Agent1(RobotNum).ShortTermActivities(0)
        t1("st_acttime1") = Agent1(RobotNum).ShortTermActivities(1)
        t1("st_acttime2") = Agent1(RobotNum).ShortTermActivities(2)
        t1("st_acttime3") = Agent1(RobotNum).ShortTermActivities(3)

        t1("st_success00") = Agent1(RobotNum).CMknow(0, 0, 2)
        t1("st_success01") = Agent1(RobotNum).CMknow(0, 1, 2)
        t1("st_success02") = Agent1(RobotNum).CMknow(0, 2, 2)
        t1("st_success03") = Agent1(RobotNum).CMknow(0, 3, 2)
        t1("st_success10") = Agent1(RobotNum).CMknow(1, 0, 2)
        t1("st_success11") = Agent1(RobotNum).CMknow(1, 1, 2)
        t1("st_success12") = Agent1(RobotNum).CMknow(1, 2, 2)
        t1("st_success13") = Agent1(RobotNum).CMknow(1, 3, 2)
        t1("st_success20") = Agent1(RobotNum).CMknow(2, 0, 2)
        t1("st_success21") = Agent1(RobotNum).CMknow(2, 1, 2)
        t1("st_success22") = Agent1(RobotNum).CMknow(2, 2, 2)
        t1("st_success23") = Agent1(RobotNum).CMknow(2, 3, 2)
        t1("st_success30") = Agent1(RobotNum).CMknow(3, 0, 2)
        t1("st_success31") = Agent1(RobotNum).CMknow(3, 1, 2)
        t1("st_success32") = Agent1(RobotNum).CMknow(3, 2, 2)
        t1("st_success33") = Agent1(RobotNum).CMknow(3, 3, 2)

        t1("CMIOCode0") = Agent1(RobotNum).CmIOCode(0)
        t1("CMIOCode1") = Agent1(RobotNum).CmIOCode(1)

        t1("CM_fuz_centre_pos0") = Agent1(RobotNum).CMFuzCentre(0)
        t1("CM_fuz_centre_pos2") = Agent1(RobotNum).CMFuzCentre(2)
        t1("CM_fuz_centre_pos4") = Agent1(RobotNum).CMFuzCentre(4)
        t1("CM_fuz_centre_pos6") = Agent1(RobotNum).CMFuzCentre(6)

        t1("CM_fuz_sigma0") = Agent1(RobotNum).CMFuzSigma(0)
        t1("CM_fuz_sigma2") = Agent1(RobotNum).CMFuzSigma(2)
        t1("CM_fuz_sigma4") = Agent1(RobotNum).CMFuzSigma(4)
        t1("CM_fuz_sigma6") = Agent1(RobotNum).CMFuzSigma(6)

        t1("CM_defuz_centre0") = Agent1(RobotNum).CMDefuzCentre(0)
        t1("CM_defuz_centre1") = Agent1(RobotNum).CMDefuzCentre(1)
        t1("CM_defuz_centre2") = Agent1(RobotNum).CMDefuzCentre(2)
        t1("CM_defuz_centre3") = Agent1(RobotNum).CMDefuzCentre(3)

        t1("CM_defuz_sigma0") = Agent1(RobotNum).CMDefuzSigma(0)
        t1("CM_defuz_sigma1") = Agent1(RobotNum).CMDefuzSigma(1)
        t1("CM_defuz_sigma2") = Agent1(RobotNum).CMDefuzSigma(2)
        t1("CM_defuz_sigma3") = Agent1(RobotNum).CMDefuzSigma(3)

        t1("CM_know0") = Left(Agent1(RobotNum).CMknow(0, 0, 0), 1) &
    Left(Agent1(RobotNum).CMknow(0, 0, 1), 1) & Left(Agent1(RobotNum).CMknow(0, 1,
    0), 1) & Left(Agent1(RobotNum).CMknow(0, 1, 1), 1) & Left(Agent1(RobotNum).
    CMknow(0, 2, 0), 1) & Left(Agent1(RobotNum).CMknow(0, 2, 1), 1) &
    Left(Agent1(RobotNum).CMknow(0, 3, 0), 1) & Left(Agent1(RobotNum).CMknow(0, 3,
    1), 1)
        t1("CM_know1") = Left(Agent1(RobotNum).CMknow(1, 0, 0), 1) &
    Left(Agent1(RobotNum).CMknow(1, 0, 1), 1) & Left(Agent1(RobotNum).CMknow(1, 1,
    0), 1) & Left(Agent1(RobotNum).CMknow(1, 1, 1), 1) & Left(Agent1(RobotNum).
    CMknow(1, 2, 0), 1) &      Left(Agent1(RobotNum).CMknow(1, 2, 1), 1) &
```

```
        Left(Agent1(RobotNum).CMknow(1, 3, 0), 1) & Left(Agent1(RobotNum).CMknow(1, 3,
        1), 1)
                t1("CM_know2") = Left(Agent1(RobotNum).CMknow(2, 0, 0), 1) &
        Left(Agent1(RobotNum).CMknow(2, 0, 1), 1) & Left(Agent1(RobotNum).CMknow(2, 1,
        0), 1) & Left(Agent1(RobotNum).CMknow(2, 1, 1), 1) & Left(Agent1(RobotNum).
        CMknow(2, 2, 0), 1) & Left(Agent1(RobotNum).CMknow(2, 2, 1), 1) &
        Left(Agent1(RobotNum).CMknow(2, 3, 0), 1) & Left(Agent1(RobotNum).CMknow(2, 3,
        1), 1)
                t1("CM_know3") = Left(Agent1(RobotNum).CMknow(3, 0, 0), 1) &
        Left(Agent1(RobotNum).CMknow(3, 0, 1), 1) & Left(Agent1(RobotNum).CMknow(3, 1,
        0), 1) & Left(Agent1(RobotNum).CMknow(3, 1, 1), 1) & Left(Agent1(RobotNum).
        CMknow(3, 2, 0), 1) & Left(Agent1(RobotNum).CMknow(3, 2, 1), 1) &
        Left(Agent1(RobotNum).CMknow(3, 3, 0), 1) & Left(Agent1(RobotNum).CMknow(3, 3,
        1), 1)

                t1("XHome") = Agent1(RobotNum).XHome
                t1("YHome") = Agent1(RobotNum).YHome

                t1("learn_io_code0") = Agent1(RobotNum).LearnIOCode(0)
                t1("learn_io_code1") = Agent1(RobotNum).LearnIOCode(1)
                t1("LearnStrategy") = Agent1(RobotNum).LearnS
                t1("learn_fuz_centre_pos2") = Agent1(RobotNum).LearnSigma(2)
                t1("learn_fuz_centre_pos4") = Agent1(RobotNum).LearnSigma(4)
                t1("learn_fuz_centre_pos6") = Agent1(RobotNum).LearnSigma(6)
                t1("learn_fuz_centre_pos8") = Agent1(RobotNum).LearnSigma(8)
                t1("learn_fuz_centre_pos10") = Agent1(RobotNum).LearnSigma(10)

                t1("learn_fuz_sigma2") = Agent1(RobotNum).LearnSigma(2)
                t1("learn_fuz_sigma4") = Agent1(RobotNum).LearnSigma(4)
                t1("learn_fuz_sigma6") = Agent1(RobotNum).LearnSigma(6)
                t1("learn_fuz_sigma8") = Agent1(RobotNum).LearnSigma(8)
                t1("learn_fuz_sigma10") = Agent1(RobotNum).LearnSigma(10)

                t1("Activities") = Agent1(RobotNum).DailyActivities
        End If
        t1.Update
t1.Close

Set t1 = db.OpenTable("2" & Agent2(RobotNum).tablenumber)
        t1.AddNew
        If ((duration - 1 > ItNum) And (Agent2(RobotNum).MutationTag <> 1)) Then
                t1("st_budgetstate0") = Agent2(RobotNum).ShortTermBudgets(0)
                t1("st_budgetstate1") = Agent2(RobotNum).ShortTermBudgets(1)
                t1("st_budgetstate2") = Agent2(RobotNum).ShortTermBudgets(2)
                t1("st_budgetstate3") = Agent2(RobotNum).ShortTermBudgets(3)

                t1("LearnStrategy") = Agent2(RobotNum).LearnS
                t1("Activities") = Agent2(RobotNum).DailyActivities
        Else
                t1("st_io_code0") = Agent2(RobotNum).ShortTermIOCode(0)
                t1("st_io_code1") = Agent2(RobotNum).ShortTermIOCode(1)
                t1("st_io_code2") = Agent2(RobotNum).ShortTermIOCode(2)
                t1("st_io_code3") = Agent2(RobotNum).ShortTermIOCode(3)
                t1("st_io_code4") = Agent2(RobotNum).ShortTermIOCode(4)
                t1("st_io_code5") = Agent2(RobotNum).ShortTermIOCode(5)
                t1("st_io_code6") = Agent2(RobotNum).ShortTermIOCode(6)
                t1("st_io_code7") = Agent2(RobotNum).ShortTermIOCode(7)

                t1("st_fuz_centre_pos0") = Agent2(RobotNum).ShortTermFuzCentre(0)
```

```
t1("st_fuz_centre_pos2") = Agent2(RobotNum).ShortTermFuzCentre(2)
t1("st_fuz_centre_pos4") = Agent2(RobotNum).ShortTermFuzCentre(4)
t1("st_fuz_centre_pos6") = Agent2(RobotNum).ShortTermFuzCentre(6)

t1("st_fuz_sigma0") = Agent2(RobotNum).ShortTermFuzSigma(0)
t1("st_fuz_sigma2") = Agent2(RobotNum).ShortTermFuzSigma(2)
t1("st_fuz_sigma4") = Agent2(RobotNum).ShortTermFuzSigma(4)
t1("st_fuz_sigma6") = Agent2(RobotNum).ShortTermFuzSigma(6)

t1("st_defuz_centre0") = Agent2(RobotNum).ShortTermDefuzCentre(0)
t1("st_defuz_centre1") = Agent2(RobotNum).ShortTermDefuzCentre(1)
t1("st_defuz_centre2") = Agent2(RobotNum).ShortTermDefuzCentre(2)
t1("st_defuz_centre3") = Agent2(RobotNum).ShortTermDefuzCentre(3)

t1("st_defuz_sigma0") = Agent2(RobotNum).ShortTermDefuzSigma(0)
t1("st_defuz_sigma1") = Agent2(RobotNum).ShortTermDefuzSigma(1)
t1("st_defuz_sigma2") = Agent2(RobotNum).ShortTermDefuzSigma(2)
t1("st_defuz_sigma3") = Agent2(RobotNum).ShortTermDefuzSigma(3)

t1("st_budgetstate0") = Agent2(RobotNum).ShortTermBudgets(0)
t1("st_budgetstate1") = Agent2(RobotNum).ShortTermBudgets(1)
t1("st_budgetstate2") = Agent2(RobotNum).ShortTermBudgets(2)
t1("st_budgetstate3") = Agent2(RobotNum).ShortTermBudgets(3)

t1("st_acttime0") = Agent2(RobotNum).ShortTermActivities(0)
t1("st_acttime1") = Agent2(RobotNum).ShortTermActivities(1)
t1("st_acttime2") = Agent2(RobotNum).ShortTermActivities(2)
t1("st_acttime3") = Agent2(RobotNum).ShortTermActivities(3)

t1("st_success00") = Agent2(RobotNum).CMknow(0, 0, 2)
t1("st_success01") = Agent2(RobotNum).CMknow(0, 1, 2)
t1("st_success02") = Agent2(RobotNum).CMknow(0, 2, 2)
t1("st_success03") = Agent2(RobotNum).CMknow(0, 3, 2)
t1("st_success10") = Agent2(RobotNum).CMknow(1, 0, 2)
t1("st_success11") = Agent2(RobotNum).CMknow(1, 1, 2)
t1("st_success12") = Agent2(RobotNum).CMknow(1, 2, 2)
t1("st_success13") = Agent2(RobotNum).CMknow(1, 3, 2)
t1("st_success20") = Agent2(RobotNum).CMknow(2, 0, 2)
t1("st_success21") = Agent2(RobotNum).CMknow(2, 1, 2)
t1("st_success22") = Agent2(RobotNum).CMknow(2, 2, 2)
t1("st_success23") = Agent2(RobotNum).CMknow(2, 3, 2)
t1("st_success30") = Agent2(RobotNum).CMknow(3, 0, 2)
t1("st_success31") = Agent2(RobotNum).CMknow(3, 1, 2)
t1("st_success32") = Agent2(RobotNum).CMknow(3, 2, 2)
t1("st_success33") = Agent2(RobotNum).CMknow(3, 3, 2)

t1("CMIOCode0") = Agent2(RobotNum).CmIOCode(0)
t1("CMIOCode1") = Agent2(RobotNum).CmIOCode(1)

t1("CM_fuz_centre_pos0") = Agent2(RobotNum).CMFuzCentre(0)
t1("CM_fuz_centre_pos2") = Agent2(RobotNum).CMFuzCentre(2)
t1("CM_fuz_centre_pos4") = Agent2(RobotNum).CMFuzCentre(4)
t1("CM_fuz_centre_pos6") = Agent2(RobotNum).CMFuzCentre(6)

t1("CM_fuz_sigma0") = Agent2(RobotNum).CMFuzSigma(0)
t1("CM_fuz_sigma2") = Agent2(RobotNum).CMFuzSigma(2)
t1("CM_fuz_sigma4") = Agent2(RobotNum).CMFuzSigma(4)
t1("CM_fuz_sigma6") = Agent2(RobotNum).CMFuzSigma(6)

t1("CM_defuz_centre0") = Agent2(RobotNum).CMDefuzCentre(0)
t1("CM_defuz_centre1") = Agent2(RobotNum).CMDefuzCentre(1)
```

```
        t1("CM_defuz_centre2") = Agent2(RobotNum).CMDefuzCentre(2)
        t1("CM_defuz_centre3") = Agent2(RobotNum).CMDefuzCentre(3)

        t1("CM_defuz_sigma0") = Agent2(RobotNum).CMDefuzSigma(0)
        t1("CM_defuz_sigma1") = Agent2(RobotNum).CMDefuzSigma(1)
        t1("CM_defuz_sigma2") = Agent2(RobotNum).CMDefuzSigma(2)
        t1("CM_defuz_sigma3") = Agent2(RobotNum).CMDefuzSigma(3)

        t1("CM_know0") = Left(Agent2(RobotNum).CMknow(0, 0, 0), 1) &
    Left(Agent2(RobotNum).CMknow(0, 0, 1), 1) & Left(Agent2(RobotNum).CMknow(0, 1,
    0), 1) & Left(Agent2(RobotNum).CMknow(0, 1, 1), 1) & Left(Agent2(RobotNum).
    CMknow(0, 2, 0), 1) & Left(Agent2(RobotNum).CMknow(0, 2, 1), 1) &
    Left(Agent2(RobotNum).CMknow(0, 3, 0), 1) & Left(Agent2(RobotNum).CMknow(0, 3,
    1), 1)
        t1("CM_know1") = Left(Agent2(RobotNum).CMknow(1, 0, 0), 1) &
    Left(Agent2(RobotNum).CMknow(1, 0, 1), 1) & Left(Agent2(RobotNum).CMknow(1, 1,
    0), 1) & Left(Agent2(RobotNum).CMknow(1, 1, 1), 1) & Left(Agent2(RobotNum).
    CMknow(1, 2, 0), 1) & Left(Agent2(RobotNum).CMknow(1, 2, 1), 1) &
    Left(Agent2(RobotNum).CMknow(1, 3, 0), 1) & Left(Agent2(RobotNum).CMknow(1, 3,
    1), 1)
        t1("CM_know2") = Left(Agent2(RobotNum).CMknow(2, 0, 0), 1) &
    Left(Agent2(RobotNum).CMknow(2, 0, 1), 1) & Left(Agent2(RobotNum).CMknow(2, 1,
    0), 1) & Left(Agent2(RobotNum).CMknow(2, 1, 1), 1) & Left(Agent2(RobotNum).
    CMknow(2, 2, 0), 1) & Left(Agent2(RobotNum).CMknow(2, 2, 1), 1) &
    Left(Agent2(RobotNum).CMknow(2, 3, 0), 1) & Left(Agent2(RobotNum).CMknow(2, 3,
    1), 1)
        t1("CM_know3") = Left(Agent2(RobotNum).CMknow(3, 0, 0), 1) &
    Left(Agent2(RobotNum).CMknow(3, 0, 1), 1) & Left(Agent2(RobotNum).CMknow(3, 1,
    0), 1) & Left(Agent2(RobotNum).CMknow(3, 1, 1), 1) & Left(Agent2(RobotNum).
    CMknow(3, 2, 0), 1) & Left(Agent2(RobotNum).CMknow(3, 2, 1), 1) &
    Left(Agent2(RobotNum).CMknow(3, 3, 0), 1) & Left(Agent2(RobotNum).CMknow(3, 3,
    1), 1)

        t1("XHome") = Agent2(RobotNum).XHome
        t1("YHome") = Agent2(RobotNum).YHome

        t1("learn_io_code0") = Agent2(RobotNum).LearnIOCode(0)
        t1("learn_io_code1") = Agent2(RobotNum).LearnIOCode(1)
        t1("LearnStrategy") = Agent2(RobotNum).LearnS

        t1("learn_fuz_centre_pos2") = Agent2(RobotNum).LearnSigma(2)
        t1("learn_fuz_centre_pos4") = Agent2(RobotNum).LearnSigma(4)
        t1("learn_fuz_centre_pos6") = Agent2(RobotNum).LearnSigma(6)
        t1("learn_fuz_centre_pos8") = Agent2(RobotNum).LearnSigma(8)
        t1("learn_fuz_centre_pos10") = Agent2(RobotNum).LearnSigma(10)

        t1("learn_fuz_sigma2") = Agent2(RobotNum).LearnSigma(2)
        t1("learn_fuz_sigma4") = Agent2(RobotNum).LearnSigma(4)
        t1("learn_fuz_sigma6") = Agent2(RobotNum).LearnSigma(6)
        t1("learn_fuz_sigma8") = Agent2(RobotNum).LearnSigma(8)
        t1("learn_fuz_sigma10") = Agent2(RobotNum).LearnSigma(10)

        t1("Activities") = Agent2(RobotNum).DailyActivities
    End If
    t1.Update
    t1.Close

Set t1 = db.OpenTable("3" & Agent3(RobotNum).tablenumber)
    t1.AddNew
    If ((duration - 1 > ItNum) And (Agent3(RobotNum).MutationTag <> 1)) Then
```

```
        t1("st_budgetstate0") = Agent3(RobotNum).ShortTermBudgets(0)
        t1("st_budgetstate1") = Agent3(RobotNum).ShortTermBudgets(1)
        t1("st_budgetstate2") = Agent3(RobotNum).ShortTermBudgets(2)
        t1("st_budgetstate3") = Agent3(RobotNum).ShortTermBudgets(3)

        t1("LearnStrategy") = Agent3(RobotNum).LearnS
        t1("Activities") = Agent3(RobotNum).DailyActivities
    Else

        t1("st_io_code0") = Agent3(RobotNum).ShortTermIOCode(0)
        t1("st_io_code1") = Agent3(RobotNum).ShortTermIOCode(1)
        t1("st_io_code2") = Agent3(RobotNum).ShortTermIOCode(2)
        t1("st_io_code3") = Agent3(RobotNum).ShortTermIOCode(3)
        t1("st_io_code4") = Agent3(RobotNum).ShortTermIOCode(4)
        t1("st_io_code5") = Agent3(RobotNum).ShortTermIOCode(5)
        t1("st_io_code6") = Agent3(RobotNum).ShortTermIOCode(6)
        t1("st_io_code7") = Agent3(RobotNum).ShortTermIOCode(7)

        t1("st_fuz_centre_pos0") = Agent3(RobotNum).ShortTermFuzCentre(0)
        t1("st_fuz_centre_pos2") = Agent3(RobotNum).ShortTermFuzCentre(2)
        t1("st_fuz_centre_pos4") = Agent3(RobotNum).ShortTermFuzCentre(4)
        t1("st_fuz_centre_pos6") = Agent3(RobotNum).ShortTermFuzCentre(6)

        t1("st_fuz_sigma0") = Agent3(RobotNum).ShortTermFuzSigma(0)
        t1("st_fuz_sigma2") = Agent3(RobotNum).ShortTermFuzSigma(2)
        t1("st_fuz_sigma4") = Agent3(RobotNum).ShortTermFuzSigma(4)
        t1("st_fuz_sigma6") = Agent3(RobotNum).ShortTermFuzSigma(6)

        t1("st_defuz_centre0") = Agent3(RobotNum).ShortTermDefuzCentre(0)
        t1("st_defuz_centre1") = Agent3(RobotNum).ShortTermDefuzCentre(1)
        t1("st_defuz_centre2") = Agent3(RobotNum).ShortTermDefuzCentre(2)
        t1("st_defuz_centre3") = Agent3(RobotNum).ShortTermDefuzCentre(3)

        t1("st_defuz_sigma0") = Agent3(RobotNum).ShortTermDefuzSigma(0)
        t1("st_defuz_sigma1") = Agent3(RobotNum).ShortTermDefuzSigma(1)
        t1("st_defuz_sigma2") = Agent3(RobotNum).ShortTermDefuzSigma(2)
        t1("st_defuz_sigma3") = Agent3(RobotNum).ShortTermDefuzSigma(3)

        t1("st_budgetstate0") = Agent3(RobotNum).ShortTermBudgets(0)
        t1("st_budgetstate1") = Agent3(RobotNum).ShortTermBudgets(1)
        t1("st_budgetstate2") = Agent3(RobotNum).ShortTermBudgets(2)
        t1("st_budgetstate3") = Agent3(RobotNum).ShortTermBudgets(3)

        t1("st_acttime0") = Agent3(RobotNum).ShortTermActivities(0)
        t1("st_acttime1") = Agent3(RobotNum).ShortTermActivities(1)
        t1("st_acttime2") = Agent3(RobotNum).ShortTermActivities(2)
        t1("st_acttime3") = Agent3(RobotNum).ShortTermActivities(3)

        t1("st_success00") = Agent3(RobotNum).CMknow(0, 0, 2)
        t1("st_success01") = Agent3(RobotNum).CMknow(0, 1, 2)
        t1("st_success02") = Agent3(RobotNum).CMknow(0, 2, 2)
        t1("st_success03") = Agent3(RobotNum).CMknow(0, 3, 2)
        t1("st_success10") = Agent3(RobotNum).CMknow(1, 0, 2)
        t1("st_success11") = Agent3(RobotNum).CMknow(1, 1, 2)
        t1("st_success12") = Agent3(RobotNum).CMknow(1, 2, 2)
        t1("st_success13") = Agent3(RobotNum).CMknow(1, 3, 2)
        t1("st_success20") = Agent3(RobotNum).CMknow(2, 0, 2)
        t1("st_success21") = Agent3(RobotNum).CMknow(2, 1, 2)
        t1("st_success22") = Agent3(RobotNum).CMknow(2, 2, 2)
        t1("st_success23") = Agent3(RobotNum).CMknow(2, 3, 2)
        t1("st_success30") = Agent3(RobotNum).CMknow(3, 0, 2)
        t1("st_success31") = Agent3(RobotNum).CMknow(3, 1, 2)
        t1("st_success32") = Agent3(RobotNum).CMknow(3, 2, 2)
        t1("st_success33") = Agent3(RobotNum).CMknow(3, 3, 2)

        t1("CMIOCode0") = Agent3(RobotNum).CmIOCode(0)
        t1("CMIOCode1") = Agent3(RobotNum).CmIOCode(1)

        t1("CM_fuz_centre_pos0") = Agent3(RobotNum).CMFuzCentre(0)
        t1("CM_fuz_centre_pos2") = Agent3(RobotNum).CMFuzCentre(2)
        t1("CM_fuz_centre_pos4") = Agent3(RobotNum).CMFuzCentre(4)
        t1("CM_fuz_centre_pos6") = Agent3(RobotNum).CMFuzCentre(6)

        t1("CM_fuz_sigma0") = Agent3(RobotNum).CMFuzSigma(0)
        t1("CM_fuz_sigma2") = Agent3(RobotNum).CMFuzSigma(2)
        t1("CM_fuz_sigma4") = Agent3(RobotNum).CMFuzSigma(4)
        t1("CM_fuz_sigma6") = Agent3(RobotNum).CMFuzSigma(6)

        t1("CM_defuz_centre0") = Agent3(RobotNum).CMDefuzCentre(0)
        t1("CM_defuz_centre1") = Agent3(RobotNum).CMDefuzCentre(1)
        t1("CM_defuz_centre2") = Agent3(RobotNum).CMDefuzCentre(2)
        t1("CM_defuz_centre3") = Agent3(RobotNum).CMDefuzCentre(3)

        t1("CM_defuz_sigma0") = Agent3(RobotNum).CMDefuzSigma(0)
        t1("CM_defuz_sigma1") = Agent3(RobotNum).CMDefuzSigma(1)
        t1("CM_defuz_sigma2") = Agent3(RobotNum).CMDefuzSigma(2)
        t1("CM_defuz_sigma3") = Agent3(RobotNum).CMDefuzSigma(3)

        t1("CM_know0") = Left(Agent3(RobotNum).CMknow(0, 0, 0), 1) &
Left(Agent3(RobotNum).CMknow(0, 0, 1), 1) & Left(Agent3(RobotNum).CMknow(0, 1,
0), 1) & Left(Agent3(RobotNum).CMknow(0, 1, 1), 1) & Left(Agent3(RobotNum).
CMknow(0, 2, 0), 1) & Left(Agent3(RobotNum).CMknow(0, 2, 1), 1) &
Left(Agent3(RobotNum).CMknow(0, 3, 0), 1) & Left(Agent3(RobotNum).CMknow(0, 3,
1), 1)
        t1("CM_know1") = Left(Agent3(RobotNum).CMknow(1, 0, 0), 1) &
Left(Agent3(RobotNum).CMknow(1, 0, 1), 1) & Left(Agent3(RobotNum).CMknow(1, 1,
0), 1) & Left(Agent3(RobotNum).CMknow(1, 1, 1), 1) & Left(Agent3(RobotNum).
CMknow(1, 2, 0), 1) & Left(Agent3(RobotNum).CMknow(1, 2, 1), 1) &
Left(Agent3(RobotNum).CMknow(1, 3, 0), 1) & Left(Agent3(RobotNum).CMknow(1, 3,
1), 1)
        t1("CM_know2") = Left(Agent3(RobotNum).CMknow(2, 0, 0), 1) &
Left(Agent3(RobotNum).CMknow(2, 0, 1), 1) & Left(Agent3(RobotNum).CMknow(2, 1,
0), 1) & Left(Agent3(RobotNum).CMknow(2, 1, 1), 1) & Left(Agent3(RobotNum).
CMknow(2, 2, 0), 1) & Left(Agent3(RobotNum).CMknow(2, 2, 1), 1) &
Left(Agent3(RobotNum).CMknow(2, 3, 0), 1) & Left(Agent3(RobotNum).CMknow(2, 3,
1), 1)
        t1("CM_know3") = Left(Agent3(RobotNum).CMknow(3, 0, 0), 1) &
Left(Agent3(RobotNum).CMknow(3, 0, 1), 1) & Left(Agent3(RobotNum).CMknow(3, 1,
0), 1) & Left(Agent3(RobotNum).CMknow(3, 1, 1), 1) & Left(Agent3(RobotNum).
CMknow(3, 2, 0), 1) & Left(Agent3(RobotNum).CMknow(3, 2, 1), 1) &
Left(Agent3(RobotNum).CMknow(3, 3, 0), 1) & Left(Agent3(RobotNum).CMknow(3, 3,
1), 1)

        t1("XHome") = Agent3(RobotNum).XHome
        t1("YHome") = Agent3(RobotNum).YHome

        t1("learn_io_code0") = Agent3(RobotNum).LearnIOCode(0)
        t1("learn_io_code1") = Agent3(RobotNum).LearnIOCode(1)
        t1("LearnStrategy") = Agent3(RobotNum).LearnS
```

```
                t1("learn_fuz_centre_pos2") = Agent3(RobotNum).LearnSigma(2)
                t1("learn_fuz_centre_pos4") = Agent3(RobotNum).LearnSigma(4)
                t1("learn_fuz_centre_pos6") = Agent3(RobotNum).LearnSigma(6)
                t1("learn_fuz_centre_pos8") = Agent3(RobotNum).LearnSigma(8)
                t1("learn_fuz_centre_pos10") = Agent3(RobotNum).LearnSigma(10)

                t1("learn_fuz_sigma2") = Agent3(RobotNum).LearnSigma(2)
                t1("learn_fuz_sigma4") = Agent3(RobotNum).LearnSigma(4)
                t1("learn_fuz_sigma6") = Agent3(RobotNum).LearnSigma(6)
                t1("learn_fuz_sigma8") = Agent3(RobotNum).LearnSigma(8)
                t1("learn_fuz_sigma10") = Agent3(RobotNum).LearnSigma(10)

                t1("Activities") = Agent3(RobotNum).DailyActivities
        End If
        t1.Update
        t1.Close

Set t1 = db.OpenTable("4" & Agent4(RobotNum).tablenumber)
        t1.AddNew
        If ((duration - 1 > ItNum) And (Agent4(RobotNum).MutationTag <> 1)) Then
                t1("st_budgetstate0") = Agent4(RobotNum).ShortTermBudgets(0)
                t1("st_budgetstate1") = Agent4(RobotNum).ShortTermBudgets(1)
                t1("st_budgetstate2") = Agent4(RobotNum).ShortTermBudgets(2)
                t1("st_budgetstate3") = Agent4(RobotNum).ShortTermBudgets(3)

                t1("LearnStrategy") = Agent4(RobotNum).LearnS
                t1("Activities") = Agent4(RobotNum).DailyActivities
        Else
                t1("st_io_code0") = Agent4(RobotNum).ShortTermIOCode(0)
                t1("st_io_code1") = Agent4(RobotNum).ShortTermIOCode(1)
                t1("st_io_code2") = Agent4(RobotNum).ShortTermIOCode(2)
                t1("st_io_code3") = Agent4(RobotNum).ShortTermIOCode(3)
                t1("st_io_code4") = Agent4(RobotNum).ShortTermIOCode(4)
                t1("st_io_code5") = Agent4(RobotNum).ShortTermIOCode(5)
                t1("st_io_code6") = Agent4(RobotNum).ShortTermIOCode(6)
                t1("st_io_code7") = Agent4(RobotNum).ShortTermIOCode(7)

                t1("st_fuz_centre_pos0") = Agent4(RobotNum).ShortTermFuzCentre(0)
                t1("st_fuz_centre_pos2") = Agent4(RobotNum).ShortTermFuzCentre(2)
                t1("st_fuz_centre_pos4") = Agent4(RobotNum).ShortTermFuzCentre(4)
                t1("st_fuz_centre_pos6") = Agent4(RobotNum).ShortTermFuzCentre(6)

                t1("st_fuz_sigma0") = Agent4(RobotNum).ShortTermFuzSigma(0)
                t1("st_fuz_sigma2") = Agent4(RobotNum).ShortTermFuzSigma(2)
                t1("st_fuz_sigma4") = Agent4(RobotNum).ShortTermFuzSigma(4)
                t1("st_fuz_sigma6") = Agent4(RobotNum).ShortTermFuzSigma(6)

                t1("st_defuz_centre0") = Agent4(RobotNum).ShortTermDefuzCentre(0)
                t1("st_defuz_centre1") = Agent4(RobotNum).ShortTermDefuzCentre(1)
                t1("st_defuz_centre2") = Agent4(RobotNum).ShortTermDefuzCentre(2)
                t1("st_defuz_centre3") = Agent4(RobotNum).ShortTermDefuzCentre(3)

                t1("st_defuz_sigma0") = Agent4(RobotNum).ShortTermDefuzSigma(0)
                t1("st_defuz_sigma1") = Agent4(RobotNum).ShortTermDefuzSigma(1)
                t1("st_defuz_sigma2") = Agent4(RobotNum).ShortTermDefuzSigma(2)
                t1("st_defuz_sigma3") = Agent4(RobotNum).ShortTermDefuzSigma(3)

                t1("st_budgetstate0") = Agent4(RobotNum).ShortTermBudgets(0)
                t1("st_budgetstate1") = Agent4(RobotNum).ShortTermBudgets(1)
                t1("st_budgetstate2") = Agent4(RobotNum).ShortTermBudgets(2)
```

```
t1("st_budgetstate3") = Agent4(RobotNum).ShortTermBudgets(3)

t1("st_acttime0") = Agent4(RobotNum).ShortTermActivities(0)
t1("st_acttime1") = Agent4(RobotNum).ShortTermActivities(1)
t1("st_acttime2") = Agent4(RobotNum).ShortTermActivities(2)
t1("st_acttime3") = Agent4(RobotNum).ShortTermActivities(3)

t1("st_success00") = Agent4(RobotNum).CMknow(0, 0, 2)
t1("st_success01") = Agent4(RobotNum).CMknow(0, 1, 2)
t1("st_success02") = Agent4(RobotNum).CMknow(0, 2, 2)
t1("st_success03") = Agent4(RobotNum).CMknow(0, 3, 2)
t1("st_success10") = Agent4(RobotNum).CMknow(1, 0, 2)
t1("st_success11") = Agent4(RobotNum).CMknow(1, 1, 2)
t1("st_success12") = Agent4(RobotNum).CMknow(1, 2, 2)
t1("st_success13") = Agent4(RobotNum).CMknow(1, 3, 2)
t1("st_success20") = Agent4(RobotNum).CMknow(2, 0, 2)
t1("st_success21") = Agent4(RobotNum).CMknow(2, 1, 2)
t1("st_success22") = Agent4(RobotNum).CMknow(2, 2, 2)
t1("st_success23") = Agent4(RobotNum).CMknow(2, 3, 2)
t1("st_success30") = Agent4(RobotNum).CMknow(3, 0, 2)
t1("st_success31") = Agent4(RobotNum).CMknow(3, 1, 2)
t1("st_success32") = Agent4(RobotNum).CMknow(3, 2, 2)
t1("st_success33") = Agent4(RobotNum).CMknow(3, 3, 2)

t1("CMIOCode0") = Agent4(RobotNum).CmIOCode(0)
t1("CMIOCode1") = Agent4(RobotNum).CmIOCode(1)

t1("CM_fuz_centre_pos0") = Agent4(RobotNum).CMFuzCentre(0)
t1("CM_fuz_centre_pos2") = Agent4(RobotNum).CMFuzCentre(2)
t1("CM_fuz_centre_pos4") = Agent4(RobotNum).CMFuzCentre(4)
t1("CM_fuz_centre_pos6") = Agent4(RobotNum).CMFuzCentre(6)

t1("CM_fuz_sigma0") = Agent4(RobotNum).CMFuzSigma(0)
t1("CM_fuz_sigma2") = Agent4(RobotNum).CMFuzSigma(2)
t1("CM_fuz_sigma4") = Agent4(RobotNum).CMFuzSigma(4)
t1("CM_fuz_sigma6") = Agent4(RobotNum).CMFuzSigma(6)

t1("CM_defuz_centre0") = Agent4(RobotNum).CMDefuzCentre(0)
t1("CM_defuz_centre1") = Agent4(RobotNum).CMDefuzCentre(1)
t1("CM_defuz_centre2") = Agent4(RobotNum).CMDefuzCentre(2)
t1("CM_defuz_centre3") = Agent4(RobotNum).CMDefuzCentre(3)

t1("CM_defuz_sigma0") = Agent4(RobotNum).CMDefuzSigma(0)
t1("CM_defuz_sigma1") = Agent4(RobotNum).CMDefuzSigma(1)
t1("CM_defuz_sigma2") = Agent4(RobotNum).CMDefuzSigma(2)
t1("CM_defuz_sigma3") = Agent4(RobotNum).CMDefuzSigma(3)

t1("CM_know0") = Left(Agent4(RobotNum).CMknow(0, 0, 0), 1) &
Left(Agent4(RobotNum).CMknow(0, 0, 1), 1) & Left(Agent4(RobotNum).CMknow(0, 1,
0), 1) & Left(Agent4(RobotNum).CMknow(0, 1, 1), 1) & Left(Agent4(RobotNum).
CMknow(0, 2, 0), 1) & Left(Agent4(RobotNum).CMknow(0, 2, 1), 1) &
Left(Agent4(RobotNum).CMknow(0, 3, 0), 1) & Left(Agent4(RobotNum).CMknow(0, 3,
1), 1)
t1("CM_know1") = Left(Agent4(RobotNum).CMknow(1, 0, 0), 1) &
Left(Agent4(RobotNum).CMknow(1, 0, 1), 1) & Left(Agent4(RobotNum).CMknow(1, 1,
0), 1) & Left(Agent4(RobotNum).CMknow(1, 1, 1), 1) & Left(Agent4(RobotNum).
CMknow(1, 2, 0), 1) & Left(Agent4(RobotNum).CMknow(1, 2, 1), 1) &
Left(Agent4(RobotNum).CMknow(1, 3, 0), 1) & Left(Agent4(RobotNum).CMknow(1, 3,
1), 1)
```

```
        t1("CM_know2") = Left(Agent4(RobotNum).CMknow(2, 0, 0), 1) &
    Left(Agent4(RobotNum).CMknow(2, 0, 1), 1) & Left(Agent4(RobotNum).CMknow(2, 1,
    0), 1) & Left(Agent4(RobotNum).CMknow(2, 1, 1), 1) & Left(Agent4(RobotNum).
    CMknow(2, 2, 0), 1) & Left(Agent4(RobotNum).CMknow(2, 2, 1), 1) &
    Left(Agent4(RobotNum).CMknow(2, 3, 0), 1) & Left(Agent4(RobotNum).CMknow(2, 3,
    1), 1)
        t1("CM_know3") = Left(Agent4(RobotNum).CMknow(3, 0, 0), 1) &
    Left(Agent4(RobotNum).CMknow(3, 0, 1), 1) & Left(Agent4(RobotNum).CMknow(3, 1,
    0), 1) & Left(Agent4(RobotNum).CMknow(3, 1, 1), 1) & Left(Agent4(RobotNum).
    CMknow(3, 2, 0), 1) & Left(Agent4(RobotNum).CMknow(3, 2, 1), 1) &
    Left(Agent4(RobotNum).CMknow(3, 3, 0), 1) & Left(Agent4(RobotNum).CMknow(3, 3,
    1), 1)

        t1("XHome") = Agent4(RobotNum).XHome
        t1("YHome") = Agent4(RobotNum).YHome

        t1("learn_io_code0") = Agent4(RobotNum).LearnIOCode(0)
        t1("learn_io_code1") = Agent4(RobotNum).LearnIOCode(1)
        t1("LearnStrategy") = Agent4(RobotNum).LearnS

        t1("learn_fuz_centre_pos2") = Agent4(RobotNum).LearnSigma(2)
        t1("learn_fuz_centre_pos4") = Agent4(RobotNum).LearnSigma(4)
        t1("learn_fuz_centre_pos6") = Agent4(RobotNum).LearnSigma(6)
        t1("learn_fuz_centre_pos8") = Agent4(RobotNum).LearnSigma(8)
        t1("learn_fuz_centre_pos10") = Agent4(RobotNum).LearnSigma(10)

        t1("learn_fuz_sigma2") = Agent4(RobotNum).LearnSigma(2)
        t1("learn_fuz_sigma4") = Agent4(RobotNum).LearnSigma(4)
        t1("learn_fuz_sigma6") = Agent4(RobotNum).LearnSigma(6)
        t1("learn_fuz_sigma8") = Agent4(RobotNum).LearnSigma(8)
        t1("learn_fuz_sigma10") = Agent4(RobotNum).LearnSigma(10)

        t1("Activities") = Agent4(RobotNum).DailyActivities
    End If
    t1.Update
t1.Close
End Sub
```

Sub WriteWorld ()

```
Dim t1 As table
Dim i As Integer

Set t1 = db.OpenTable("world" & number)
    t1.AddNew
    Select Case ItNum
    Case Is <> duration - 1
        t1("cap_0") = ThisWorld.Capacity(0, number)
        t1("cap_1") = ThisWorld.Capacity(1, number)
        t1("cap_2") = ThisWorld.Capacity(2, number)
        t1("cap_3") = ThisWorld.Capacity(3, number)

        t1("price_0") = ThisWorld.Price(0, number)
        t1("price_1") = ThisWorld.Price(1, number)
        t1("price_2") = ThisWorld.Price(2, number)
        t1("price_3") = ThisWorld.Price(3, number)
        For i = 0 To 23
            t1("payoff1_" & i) = ThisWorld.Act(1, i, number)
        Next i
    Case Else
```

```
            t1("cap_0") = ThisWorld.Capacity(0, number)
            t1("cap_1") = ThisWorld.Capacity(1, number)
            t1("cap_2") = ThisWorld.Capacity(2, number)
            t1("cap_3") = ThisWorld.Capacity(3, number)

            t1("price_0") = ThisWorld.Price(0, number)
            t1("price_1") = ThisWorld.Price(1, number)
            t1("price_2") = ThisWorld.Price(2, number)
            t1("price_3") = ThisWorld.Price(3, number)
            For i = 0 To 23
                t1("payoff0_" & i) = ThisWorld.Act(0, i, number)
                t1("payoff1_" & i) = ThisWorld.Act(1, i, number)
                t1("payoff2_" & i) = ThisWorld.Act(2, i, number)
                t1("payoff3_" & i) = ThisWorld.Act(3, i, number)
            Next i
        End Select
        t1.Update
t1.Close
End Sub
```

Analysis.bas

```
Option Explicit

Dim lsArray()
Dim AgeArray()
Dim Occurrence As Integer
Dim Mean As Single
Dim StdDev As Single
Dim Value(9) As Long
Const Scalefac = 50
Dim t As table
Dim TypeNum As Integer
```

Sub CalcStd ()

```
Dim i As Integer
Dim sum As Single

sum = 0
For i = 0 To Occurrence
        sum = AgeArray(i, 0) + sum
Next i
Mean = sum / (Occurrence + 1)
distribu.Label5.Caption = Mean
For i = 0 To Occurrence
        AgeArray(i, 1) = (AgeArray(i, 0) - Mean) ^ 2
Next i
sum = 0
For i = 0 To Occurrence
        sum = AgeArray(i, 1) + sum
Next i
StdDev = Sqr(sum / (Occurrence + 1))
distribu.Label3.Caption = StdDev
distribu.Label9.Caption = Occurrence + 1
End Sub
```

Sub Classify ()

```
If t("All") < Scalefac Then
        Value(0) = Value(0) + 1
ElseIf ((t("All") >= Scalefac) And (t("All") < 2 * Scalefac)) Then
        Value(1) = Value(1) + 1
ElseIf ((t("All") >= 2 * Scalefac) And (t("All") < 3 * Scalefac)) Then
        Value(2) = Value(2) + 1
ElseIf ((t("All") >= 3 * Scalefac) And (t("All") < 4 * Scalefac)) Then
        Value(3) = Value(3) + 1
ElseIf ((t("All") >= 4 * Scalefac) And (t("All") < 5 * Scalefac)) Then
        Value(4) = Value(4) + 1
ElseIf ((t("All") >= 5 * Scalefac) And (t("All") < 6 * Scalefac)) Then
        Value(5) = Value(5) + 1
ElseIf ((t("All") >= 6 * Scalefac) And (t("All") < 7 * Scalefac)) Then
        Value(6) = Value(6) + 1
ElseIf ((t("All") >= 7 * Scalefac) And (t("All") < 8 * Scalefac)) Then
        Value(7) = Value(7) + 1
ElseIf ((t("All") >= 8 * Scalefac) And (t("All") < 9 * Scalefac)) Then
        Value(8) = Value(8) + 1
ElseIf ((t("All") >= 9 * Scalefac)) Then
        Value(9) = Value(9) + 1
End If
End Sub
```

Sub DrawDistribu ()

```
Dim i As Integer

distribu.Picture1.Cls
For i = 0 To 9
        distribu.Picture1.Line (240 + (i * 840), 6600)-(500 + (i * 840), 6600 - Value(i) * 10),
        QBColor(1), BF
        distribu.Label1(i).Caption = Scalefac * i & " - " & Scalefac * (i + 1) - 1
        distribu.Label7(i).Caption = Value(i)
Next i
distribu.Label1(9).Caption = "> " & Scalefac * 9
End Sub
```

Sub DrawTSeries ()

```
Dim db As database
Dim t As table
Dim i As Integer
Static TMean(19, 1) As Long
Dim count As Integer
Dim tname As String
Dim j As Integer

TMean(0, 0) = 0
Set db = OpenDatabase("c:\results.mdb", True)
For i = 1 To 19
        Set t = db.OpenTable("A" & i)
                count = 1
                j = i * 250 + 250
                tname = CStr(j)
                t.MoveFirst
                TMean(i, 0) = t(tname)
                t.MoveNext
                        Do Until t.EOF
```

```
                    TMean(i, 0) = TMean(i, 0) + t(tname)
                    t.MoveNext
                    count = count + 1
                Loop
        t.Close
        TMean(i, 0) = TMean(i, 0) / count
Next i
db.Close

Meangraph.Graph1.AutoInc = 1
Meangraph.Graph1.NumSets = 1
Meangraph.Graph1.NumPoints = 20
Meangraph.Graph1.LineStats = 8
Meangraph.Graph1.LabelEvery = 2
Meangraph.Graph1.GraphTitle = "Time Series of Average Time between Resets"
Meangraph.Graph1.GraphType = 6

Meangraph.Graph1.LegendText = "Av. Reset Time"
Meangraph.Show
For i = 0 To 19
        TMean(i, 1) = 250 * (i + 1)
        Meangraph.Graph1.ThisPoint = i + 1
        Meangraph.Graph1.LabelText = TMean(i, 1)
Next i
For i = 0 To Meangraph.Graph1.NumPoints - 1
        Meangraph.Graph1.ThisPoint = i + 1
        Meangraph.Graph1.GraphData = TMean(i, 0)
Next i
End Sub
```

Sub get_ls ()

```
Dim db As database
Dim t As table
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim TypeNum As Integer
Dim AgentNum As Integer
Dim NumRecords As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t = db.OpenTable("Typedescriptor")
        t.MoveFirst
        AgentNum = t("number")
t.Close
NumRecords = 0
Set t = db.OpenTable("10")
        t.MoveFirst
        Do Until t.EOF
                t.MoveNext
                NumRecords = NumRecords + 1
        Loop
t.Close
NumRecords = NumRecords / 10
ReDim lsArray(NumRecords - 1, 3)
For TypeNum = 1 To 4
        For i = 0 To AgentNum - 1
        Set t = db.OpenTable(TypeNum & i)
            t.MoveFirst
            t.MoveNext
            For j = 0 To NumRecords - 1
                Select Case t("LearnStrategy")
                Case 0
                    lsArray(j, 0) = lsArray(j, 0) + 1
                Case 1
                    lsArray(j, 1) = lsArray(j, 1) + 1
                Case 2
                    lsArray(j, 2) = lsArray(j, 2) + 1
                Case 3
                    lsArray(j, 3) = lsArray(j, 3) + 1
                End Select
                For k = 0 To 9
                    t.MoveNext
                Next k
            Next j
        t.Close
    Next i
Next TypeNum

Meangraph.Graph1.AutoInc = 1
Meangraph.Graph1.NumSets = 4
Meangraph.Graph1.NumPoints = NumRecords - 1
Meangraph.Graph1.LineStats = 0
Meangraph.Graph1.LabelEvery = 50
Meangraph.Graph1.GraphTitle = "Distribution of Learning Strategies"
Meangraph.Graph1.GraphType = 8

For i = 1 To Meangraph.Graph1.NumPoints
    Meangraph.Graph1.LabelText = (i * 10) - 10
Next i
For i = 0 To 3
    Meangraph.Graph1.LegendText = i
Next i
For i = 0 To Meangraph.Graph1.NumSets - 1
    For j = 1 To Meangraph.Graph1.NumPoints
        Meangraph.Graph1.GraphData = lsArray(j, Meangraph.Graph1.ThisSet - 1)
    Next j
Next i
Meangraph.Show
Meangraph.Graph1.DrawMode = 2
End Sub
```

Sub GetAge ()

```
Dim db1 As database
Dim db2 As database
Dim t1 As table
Dim t2 As table
Dim AgeCount As Integer
Dim indicator As Integer
Dim store As Integer
Dim i As Integer
Dim TotNumber As Integer
Dim TypeNum As Integer
```

```
Set db1 = OpenDatabase("C:\blah.mdb")
Set db2 = OpenDatabase("C:\results.mdb")
Set t2 = db2.OpenTable("Age")
    Set t1 = db1.OpenTable("Typedescriptor")
        t1.MoveFirst
        TotNumber = t1("number") - 1
    t1.Close
    For TypeNum = 1 To 4
        For RobotNum = 0 To TotNumber
            AgeCount = 0
            Set t1 = db1.OpenTable(TypeNum & RobotNum)
                t1.MoveFirst
                t1.MoveNext
                Do Until t1.EOF
                    For i = 0 To 3
                        If t1("st_budgetstate" & i) < -500 Then
                            indicator = 1
                        End If
                    Next i
                    If indicator = 1 Then
                        t2.AddNew
                        t2("Age_" & TypeNum & RobotNum) = AgeCount
                        t2("All") = AgeCount
                        t2("LearnStrat") = t1("LearnStrategy")
                        t2.Update
                        AgeCount = 1
                        indicator = 0
                    Else
                        AgeCount = AgeCount + 1
                    End If
                    store = t1("LearnStrategy")
                    t1.MoveNext
                Loop
                t2.AddNew
                t2("Age_" & TypeNum & RobotNum) = AgeCount
                t2("All") = AgeCount
                t2("LearnStrat") = store
                t2.Update
            t1.Close
        Next RobotNum
    Next TypeNum
t2.Close
db2.Close
db1.Close
End Sub
```

Sub GetTSeries ()

```
Dim db1 As database
Dim db2 As database
Dim t1 As table
Dim t2 As table
Dim t3 As table
Dim AgeCount1 As Integer
Dim AgeCount2 As Integer
Dim indicator As Integer
Dim RecCount As Integer
Dim i As Integer
Dim TotNumber As Integer
Dim TypeNum As Integer

Set db1 = OpenDatabase("C:\blah.mdb")
Set db2 = OpenDatabase("C:\results.mdb")
Set t1 = db1.OpenTable("Typedescriptor")
        t1.MoveFirst
        TotNumber = t1("number") - 1
t1.Close
Set t2 = db2.OpenTable("A1")
For TypeNum = 1 To 4
        For RobotNum = 0 To TotNumber
                AgeCount1 = 1
                AgeCount2 = 1
                Set t1 = db1.OpenTable(TypeNum & RobotNum)
                t1.MoveFirst
                t1.MoveNext
                RecCount = 1
                        Do Until t1.EOF
                                For i = 0 To 3
                                        If t1("st_budgetstate" & i) < -500 Then
                                                indicator = 1
                                        End If
                                Next i
                                Select Case RecCount
                                Case 1
                                        Set t2 = db2.OpenTable("A1")
                                        Case 250
                                                Set t3 = db2.OpenTable("A2")
                                                AgeCount2 = 0
                                        Case 500
                                                t2.AddNew
                                                t2("500") = AgeCount1
                                                t2.Update
                                                t2.Close
                                                AgeCount1 = 0
                                                Set t2 = db2.OpenTable("A3")
                                        Case 750
                                                t3.AddNew
                                                t3("750") = AgeCount2
                                                t3.Update
                                                t3.Close
                                                AgeCount2 = 0
                                                Set t3 = db2.OpenTable("A4")
                                        Case 1000
                                                t2.AddNew
                                                t2("1000") = AgeCount1
                                                t2.Update
                                                t2.Close
                                                AgeCount1 = 0
                                                Set t2 = db2.OpenTable("A5")
                                        Case 1250
                                                t3.AddNew
                                                t3("1250") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A6")
        Case 1500
                t2.AddNew
                t2("1500") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
                Set t2 = db2.OpenTable("A7")
        Case 1750
                t3.AddNew
                t3("1750") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A8")
        Case 2000
                t2.AddNew
                t2("2000") = AgeCount1
                t2.Update
                AgeCount1 = 0
                t2.Close
                Set t2 = db2.OpenTable("A9")
        Case 2250
                t3.AddNew
                t3("2250") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A10")
        Case 2500
                t2.AddNew
                t2("2500") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
                Set t2 = db2.OpenTable("A11")
        Case 2750
                t3.AddNew
                t3("2750") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A12")
        Case 3000
                t2.AddNew
                t2("3000") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
                Set t2 = db2.OpenTable("A13")
        Case 3250
                t3.AddNew
                t3("3250") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A14")
        Case 3500
                t2.AddNew
                t2("3500") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
                Set t2 = db2.OpenTable("A15")
        Case 3750
                t3.AddNew
                t3("3750") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A16")
        Case 4000
                t2.AddNew
                t2("4000") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
                Set t2 = db2.OpenTable("A17")
        Case 4250
                t3.AddNew
                t3("4250") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
                Set t3 = db2.OpenTable("A18")
        Case 4500
                t2.AddNew
                t2("4500") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
                Set t2 = db2.OpenTable("A19")
        Case 4750
                t3.AddNew
                t3("4750") = AgeCount2
                t3.Update
                t3.Close
                AgeCount2 = 0
        Case 5000
                t2.AddNew
                t2("5000") = AgeCount1
                t2.Update
                t2.Close
                AgeCount1 = 0
        End Select

        If indicator = 1 Then
                If RecCount < 500 Then
                        t2.AddNew
        t2("500") = AgeCount1
        t2.Update
End If
If (RecCount > 250) And (RecCount < 750) Then
        t3.AddNew
        t3("750") = AgeCount2
        t3.Update
End If
If (RecCount > 500) And (RecCount < 1000) Then
        t2.AddNew
        t2("1000") = AgeCount1
        t2.Update
End If
If (RecCount > 750) And (RecCount < 1250) Then
        t3.AddNew
        t3("1250") = AgeCount2
        t3.Update
End If
If (RecCount > 1000) And (RecCount < 1500) Then
        t2.AddNew
        t2("1500") = AgeCount1
        t2.Update
End If
If (RecCount > 1250) And (RecCount < 1750) Then
        t3.AddNew
        t3("1750") = AgeCount2
        t3.Update
End If
If (RecCount > 1500) And (RecCount < 2000) Then
        t2.AddNew
        t2("2000") = AgeCount1
        t2.Update
End If
If (RecCount > 1750) And (RecCount < 2250) Then
        t3.AddNew
        t3("2250") = AgeCount2
        t3.Update
End If
If (RecCount > 2000) And (RecCount < 2500) Then
        t2.AddNew
        t2("2500") = AgeCount1
        t2.Update
End If
If (RecCount > 2250) And (RecCount < 2750) Then
        t3.AddNew
        t3("2750") = AgeCount2
        t3.Update
End If
If (RecCount > 2500) And (RecCount < 3000) Then
        t2.AddNew
        t2("3000") = AgeCount1
        t2.Update
End If
If (RecCount > 2750) And (RecCount < 3250) Then
        t3.AddNew
        t3("3250") = AgeCount2
```

```
                                        t3.Update
                              End If
                              If (RecCount > 3000) And (RecCount < 3500) Then
                                        t2.AddNew
                                        t2("3500") = AgeCount1
                                        t2.Update
                              End If
                              If (RecCount > 3250) And (RecCount < 3750) Then
                                        t3.AddNew
                                        t3("3750") = AgeCount2
                                        t3.Update
                              End If
                              If (RecCount > 3500) And (RecCount < 4000) Then
                                        t2.AddNew
                                        t2("4000") = AgeCount1
                                        t2.Update
                              End If
                              If (RecCount > 3750) And (RecCount < 4250) Then
                                        t3.AddNew
                                        t3("4250") = AgeCount2
                                        t3.Update
                              End If
                              If (RecCount > 4000) And (RecCount < 4500) Then
                                        t2.AddNew
                                        t2("4500") = AgeCount1
                                        t2.Update
                              End If
                              If (RecCount > 4250) And (RecCount < 4750) Then
                                        t3.AddNew
                                        t3("4750") = AgeCount2
                                        t3.Update
                              End If
                              If (RecCount > 4500) And (RecCount < 5000) Then
                                        t2.AddNew
                                        t2("5000") = AgeCount1
                                        t2.Update
                              End If
                              AgeCount1 = 0
                              AgeCount2 = 0
                              indicator = 0
                    Else
                              AgeCount1 = AgeCount1 + 1
                              AgeCount2 = AgeCount2 + 1
                    End If
                    t1.MoveNext
                    RecCount = RecCount + 1
          Loop
     t1.Close
   Next RobotNum
 Next TypeNum
db2.Close
db1.Close
End Sub
```

Sub LogAge ()

```
Dim NewTd As New TableDef, NewTd1 As New TableDef, NewTd2 As New TableDef
Dim NewTd3 As New TableDef, NewTd4 As New TableDef, NewTd5 As New TableDef
Dim NewTd6 As New TableDef, NewTd7 As New TableDef, NewTd8 As New TableDef
Dim NewTd9 As New TableDef, NewTd10 As New TableDef, NewTd11 As New TableDef
Dim NewTd12 As New TableDef, NewTd13 As New TableDef, NewTd14 As New TableDef
Dim NewTd15 As New TableDef, NewTd16 As New TableDef, NewTd17 As New TableDef
Dim NewTd18 As New TableDef, NewTd19 As New TableDef

Dim f1 As New Field, f2 As New Field, f3 As New Field
Dim f4 As New Field, f5 As New Field, f6 As New Field
Dim f7 As New Field, f8 As New Field, f9 As New Field
Dim f10 As New Field, f11 As New Field, f12 As New Field
Dim f13 As New Field, f14 As New Field, f15 As New Field
Dim f16 As New Field, f17 As New Field, f18 As New Field
Dim f19 As New Field, f20 As New Field, f21 As New Field
Dim f22 As New Field, f23 As New Field, f24 As New Field
Dim f25 As New Field, f26 As New Field, f27 As New Field
Dim f28 As New Field, f29 As New Field, f30 As New Field
Dim f31 As New Field, f32 As New Field, f33 As New Field
Dim f34 As New Field, f35 As New Field, f36 As New Field
Dim f37 As New Field, f38 As New Field, f39 As New Field
Dim f40 As New Field, f41 As New Field, f42 As New Field
Dim f43 As New Field, f44 As New Field, f45 As New Field
Dim f46 As New Field, f47 As New Field, f48 As New Field
Dim f49 As New Field, f50 As New Field, f51 As New Field
Dim f52 As New Field, f53 As New Field, f54 As New Field
Dim f55 As New Field, f56 As New Field, f57 As New Field
Dim f58 As New Field, f59 As New Field, f60 As New Field
Dim f61 As New Field, f62 As New Field, f63 As New Field
Dim f64 As New Field, f65 As New Field, f66 As New Field
Dim f67 As New Field, f68 As New Field, f69 As New Field
Dim f70 As New Field, f71 As New Field, f72 As New Field
Dim f73 As New Field, f74 As New Field, f75 As New Field
Dim f76 As New Field, f77 As New Field, f78 As New Field
Dim f79 As New Field, f80 As New Field, f81 As New Field
Dim f82 As New Field, f83 As New Field, f84 As New Field
Dim f85 As New Field

Set db = CreateDatabase("C:\results.mdb", DB_LANG_GENERAL)
NewTd.Name = "Age"

f1.Name = "Age_10"
f1.Type = DB_INTEGER
NewTd.Fields.Append f1

f2.Name = "Age_11"
f2.Type = DB_INTEGER
NewTd.Fields.Append f2

f3.Name = "Age_12"
f3.Type = DB_INTEGER
NewTd.Fields.Append f3

f4.Name = "Age_13"
f4.Type = DB_INTEGER
NewTd.Fields.Append f4

f5.Name = "Age_14"
f5.Type = DB_INTEGER
```

```
NewTd.Fields.Append f5

f6.Name = "Age_15"
f6.Type = DB_INTEGER
NewTd.Fields.Append f6

f7.Name = "Age_16"
f7.Type = DB_INTEGER
NewTd.Fields.Append f7

f8.Name = "Age_17"
f8.Type = DB_INTEGER
NewTd.Fields.Append f8

f9.Name = "Age_18"
f9.Type = DB_INTEGER
NewTd.Fields.Append f9

f10.Name = "Age_19"
f10.Type = DB_INTEGER
NewTd.Fields.Append f10

f11.Name = "Age_110"
f11.Type = DB_INTEGER
NewTd.Fields.Append f11

f12.Name = "Age_111"
f12.Type = DB_INTEGER
NewTd.Fields.Append f12

f13.Name = "Age_112"
f13.Type = DB_INTEGER
NewTd.Fields.Append f13

f14.Name = "Age_113"
f14.Type = DB_INTEGER
NewTd.Fields.Append f14

f15.Name = "Age_114"
f15.Type = DB_INTEGER
NewTd.Fields.Append f15

f16.Name = "Age_115"
f16.Type = DB_INTEGER
NewTd.Fields.Append f16

f17.Name = "Age_20"
f17.Type = DB_INTEGER
NewTd.Fields.Append f17

f18.Name = "Age_21"
f18.Type = DB_INTEGER
NewTd.Fields.Append f18

f19.Name = "Age_22"
f19.Type = DB_INTEGER
NewTd.Fields.Append f19

f20.Name = "Age_23"
f20.Type = DB_INTEGER
NewTd.Fields.Append f20

f21.Name = "Age_24"
```

```
f21.Type = DB_INTEGER
NewTd.Fields.Append f21

f22.Name = "Age_25"
f22.Type = DB_INTEGER
NewTd.Fields.Append f22

f23.Name = "Age_26"
f23.Type = DB_INTEGER
NewTd.Fields.Append f23

f24.Name = "Age_27"
f24.Type = DB_INTEGER
NewTd.Fields.Append f24

f25.Name = "Age_28"
f25.Type = DB_INTEGER
NewTd.Fields.Append f25

f26.Name = "Age_29"
f26.Type = DB_INTEGER
NewTd.Fields.Append f26

f27.Name = "Age_210"
f27.Type = DB_INTEGER
NewTd.Fields.Append f27

f28.Name = "Age_211"
f28.Type = DB_INTEGER
NewTd.Fields.Append f28

f29.Name = "Age_212"
f29.Type = DB_INTEGER
NewTd.Fields.Append f29

f30.Name = "Age_213"
f30.Type = DB_INTEGER
NewTd.Fields.Append f30

f31.Name = "Age_214"
f31.Type = DB_INTEGER
NewTd.Fields.Append f31

f32.Name = "Age_215"
f32.Type = DB_INTEGER
NewTd.Fields.Append f32

f33.Name = "Age_30"
f33.Type = DB_INTEGER
NewTd.Fields.Append f33

f34.Name = "Age_31"
f34.Type = DB_INTEGER
NewTd.Fields.Append f34

f35.Name = "Age_32"
f35.Type = DB_INTEGER
NewTd.Fields.Append f35

f36.Name = "Age_33"
f36.Type = DB_INTEGER
NewTd.Fields.Append f36

f37.Name = "Age_34"
```

```
f37.Type = DB_INTEGER
NewTd.Fields.Append f37

f38.Name = "Age_35"
f38.Type = DB_INTEGER
NewTd.Fields.Append f38

f39.Name = "Age_36"
f39.Type = DB_INTEGER
NewTd.Fields.Append f39

f40.Name = "Age_37"
f40.Type = DB_INTEGER
NewTd.Fields.Append f40

f41.Name = "Age_38"
f41.Type = DB_INTEGER
NewTd.Fields.Append f41

f42.Name = "Age_39"
f42.Type = DB_INTEGER
NewTd.Fields.Append f42

f43.Name = "Age_310"
f43.Type = DB_INTEGER
NewTd.Fields.Append f43

f44.Name = "Age_311"
f44.Type = DB_INTEGER
NewTd.Fields.Append f44

f45.Name = "Age_312"
f45.Type = DB_INTEGER
NewTd.Fields.Append f45

f46.Name = "Age_313"
f46.Type = DB_INTEGER
NewTd.Fields.Append f46

f47.Name = "Age_314"
f47.Type = DB_INTEGER
NewTd.Fields.Append f47

f48.Name = "Age_315"
f48.Type = DB_INTEGER
NewTd.Fields.Append f48

f49.Name = "Age_40"
f49.Type = DB_INTEGER
NewTd.Fields.Append f49

f50.Name = "Age_41"
f50.Type = DB_INTEGER
NewTd.Fields.Append f50

f51.Name = "Age_42"
f51.Type = DB_INTEGER
NewTd.Fields.Append f51

f52.Name = "Age_43"
f52.Type = DB_INTEGER
NewTd.Fields.Append f52

f53.Name = "Age_44"
```

```
f53.Type = DB_INTEGER
NewTd.Fields.Append f53

f54.Name = "Age_45"
f54.Type = DB_INTEGER
NewTd.Fields.Append f54

f55.Name = "Age_46"
f55.Type = DB_INTEGER
NewTd.Fields.Append f55

f56.Name = "Age_47"
f56.Type = DB_INTEGER
NewTd.Fields.Append f56

f57.Name = "Age_48"
f57.Type = DB_INTEGER
NewTd.Fields.Append f57

f58.Name = "Age_49"
f58.Type = DB_INTEGER
NewTd.Fields.Append f58

f59.Name = "Age_410"
f59.Type = DB_INTEGER
NewTd.Fields.Append f59

f60.Name = "Age_411"
f60.Type = DB_INTEGER
NewTd.Fields.Append f60

f61.Name = "Age_412"
f61.Type = DB_INTEGER
NewTd.Fields.Append f61

f62.Name = "Age_413"
f62.Type = DB_INTEGER
NewTd.Fields.Append f62

f63.Name = "Age_414"
f63.Type = DB_INTEGER
NewTd.Fields.Append f63

f64.Name = "Age_415"
f64.Type = DB_INTEGER
NewTd.Fields.Append f64

f65.Name = "All"
f65.Type = DB_INTEGER
NewTd.Fields.Append f65

f77.Name = "LearnStrat"
f77.Type = DB_INTEGER
NewTd.Fields.Append f77

db.TableDefs.Append NewTd
NewTd1.Name = "A1"
f66.Name = "500"
f66.Type = DB_INTEGER
NewTd1.Fields.Append f66
db.TableDefs.Append NewTd1

NewTd2.Name = "A2"
```

```
f67.Name = "750"
f67.Type = DB_INTEGER
NewTd2.Fields.Append f67
db.TableDefs.Append NewTd2

NewTd3.Name = "A3"
f68.Name = "1000"
f68.Type = DB_INTEGER
NewTd3.Fields.Append f68
db.TableDefs.Append NewTd3

NewTd4.Name = "A4"
f69.Name = "1250"
f69.Type = DB_INTEGER
NewTd4.Fields.Append f69
db.TableDefs.Append NewTd4

NewTd5.Name = "A5"
f70.Name = "1500"
f70.Type = DB_INTEGER
NewTd5.Fields.Append f70
db.TableDefs.Append NewTd5

NewTd6.Name = "A6"
f71.Name = "1750"
f71.Type = DB_INTEGER
NewTd6.Fields.Append f71
db.TableDefs.Append NewTd6

NewTd7.Name = "A7"
f72.Name = "2000"
f72.Type = DB_INTEGER
NewTd7.Fields.Append f72
db.TableDefs.Append NewTd7

NewTd8.Name = "A8"
f73.Name = "2250"
f73.Type = DB_INTEGER
NewTd8.Fields.Append f73
db.TableDefs.Append NewTd8

NewTd9.Name = "A9"
f74.Name = "2500"
f74.Type = DB_INTEGER
NewTd9.Fields.Append f74
db.TableDefs.Append NewTd9

NewTd10.Name = "A10"
f75.Name = "2750"
f75.Type = DB_INTEGER
NewTd10.Fields.Append f75
db.TableDefs.Append NewTd10

NewTd11.Name = "A11"
f76.Name = "3000"
f76.Type = DB_INTEGER
NewTd11.Fields.Append f76
db.TableDefs.Append NewTd11

NewTd12.Name = "A12"
f78.Name = "3250"
```

```
f78.Type = DB_INTEGER
NewTd12.Fields.Append f78
db.TableDefs.Append NewTd12

NewTd13.Name = "A13"
f79.Name = "3500"
f79.Type = DB_INTEGER
NewTd13.Fields.Append f79
db.TableDefs.Append NewTd13

NewTd14.Name = "A14"
f80.Name = "3750"
f80.Type = DB_INTEGER
NewTd14.Fields.Append f80
db.TableDefs.Append NewTd14

NewTd15.Name = "A15"
f81.Name = "4000"
f81.Type = DB_INTEGER
NewTd15.Fields.Append f81
db.TableDefs.Append NewTd15

NewTd16.Name = "A16"
f82.Name = "4250"
f82.Type = DB_INTEGER
NewTd16.Fields.Append f82
db.TableDefs.Append NewTd16

NewTd17.Name = "A17"
f83.Name = "4500"
f83.Type = DB_INTEGER
NewTd17.Fields.Append f83
db.TableDefs.Append NewTd17

NewTd18.Name = "A18"
f84.Name = "4750"
f84.Type = DB_INTEGER
NewTd18.Fields.Append f84
db.TableDefs.Append NewTd18

NewTd19.Name = "A19"
f85.Name = "5000"
f85.Type = DB_INTEGER
NewTd19.Fields.Append f85
db.TableDefs.Append NewTd19
db.Close
End Sub
```

Sub ReadResults ()

```
Dim i As Integer
Dim count As Integer

Set db = OpenDatabase("C:\results.mdb", True)
Set t = db.OpenTable("Age")
        For i = 0 To 9
                Value(i) = 0
        Next i
        t.MoveFirst
        Occurrence = 0
        Do Until t.EOF
                Occurrence = Occurrence + 1
                t.MoveNext
        Loop
        ReDim AgeArray(Occurrence, 1)
        count = 0
        t.MoveFirst
        Do Until t.EOF
                AgeArray(count, 0) = t("All")
                Classify
                t.MoveNext
                count = count + 1
        Loop
t.Close
db.Close
End Sub
```

Sub ReadWeighRes ()

```
Dim i As Integer
Dim count As Integer

Set db = OpenDatabase("C:\results.mdb", True)
Set t = db.OpenTable("Age")
        For i = 0 To 9
                Value(i) = 0
        Next i
        t.MoveFirst
        count = 0
        t.MoveFirst
        Do Until t.EOF
                WeighClass
                t.MoveNext
                count = count + 1
        Loop
t.Close
db.Close
distribu.Picture1.Cls
For i = 0 To 9
        distribu.Picture1.Line (240 + (i * 840), 6600)-(500 + (i * 840), 6600 - Value(i) / 20),
                QBColor(7), BF
        distribu.Label1(i).Caption = Scalefac * i & " - " & Scalefac * (i + 1) - 1
        distribu.Label7(i).Caption = Value(i)
Next i
distribu.Label1(9).Caption = "> " & Scalefac * 9
End Sub
```

Sub WeighClass ()

```
If t("All") < Scalefac Then
        Value(0) = Value(0) + t("All")
ElseIf ((t("All") >= Scalefac) And (t("All") < 2 * Scalefac)) Then
        Value(1) = Value(1) + t("All")
ElseIf ((t("All") >= 2 * Scalefac) And (t("All") < 3 * Scalefac)) Then
        Value(2) = Value(2) + t("All")
ElseIf ((t("All") >= 3 * Scalefac) And (t("All") < 4 * Scalefac)) Then
        Value(3) = Value(3) + t("All")
```

```
ElseIf ((t("All") >= 4 * Scalefac) And (t("All") < 5 * Scalefac)) Then
        Value(4) = Value(4) + t("All")
ElseIf ((t("All") >= 5 * Scalefac) And (t("All") < 6 * Scalefac)) Then
        Value(5) = Value(5) + t("All")
ElseIf ((t("All") >= 6 * Scalefac) And (t("All") < 7 * Scalefac)) Then
        Value(6) = Value(6) + t("All")
ElseIf ((t("All") >= 7 * Scalefac) And (t("All") < 8 * Scalefac)) Then
        Value(7) = Value(7) + t("All")
ElseIf ((t("All") >= 8 * Scalefac) And (t("All") < 9 * Scalefac)) Then
        Value(8) = Value(8) + t("All")
ElseIf ((t("All") >= 9 * Scalefac)) Then
        Value(9) = Value(9) + t("All")
End If
End Sub
```

Binary.bas

```
Option Explicit

Dim i As Integer
Dim j As Integer
Global output1 As String
Global output2 As String
Global output3 As String
Global output4 As String
Dim actornum As Integer
Dim actorname As String
Global ipt0 As Double
Global ipt1 As Double
Global ipt2 As Double
Global ipt3 As Double
Global ipt4 As Double
Global ipt5 As Double
Global ipt6 As Double
Global ipt7 As Double
Global test1 As Double
Global test2 As Double
Global code0 As Double
Global code1 As Double
Global code2 As Double
Global code3 As Double
Global code4 As Double
Global code5 As Double
Global code6 As Double
Global code7 As Double
```

Sub bittest ()

```
For i = 0 To 15
        If (test1 And 2 ^ i) = 2 ^ i Then
                io_edit.Option1(i).Value = True
        Else io_edit.Option2(i).Value = True
        End If
Next i
For j = 0 To 15
        If (test2 And 2 ^ j) = 2 ^ j Then
                io_edit.Option3(j).Value = True
        Else io_edit.Option4(j).Value = True
        End If
Next j
End Sub
```

Sub retrieve ()

```
Dim db As database
Dim t1 As table
Dim t2 As table

Set db = OpenDatabase("c:\blah.mdb")
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", io_edit.Text3.Text
        output1 = t1("st_act_name1")            'activity names for display in combo box
        output2 = t1("st_act_name2")
        output3 = t1("st_act_name3")
        output4 = t1("st_act_name4")
        actornum = t1("number") 'number of actors
        actorname = t1("name")'name of actor which is edited
t1.Close
Set t2 = db.OpenTable(actorname & 0)
        t2.MoveFirst
        ipt0 = t2("st_io_code0")
        ipt1 = t2("st_io_code1")
        ipt2 = t2("st_io_code2")
        ipt3 = t2("st_io_code3")
        ipt4 = t2("st_io_code4")
        ipt5 = t2("st_io_code5")
        ipt6 = t2("st_io_code6")
        ipt7 = t2("st_io_code7")
t2.Close
db.Close

io_edit.Combo1.Clear 'removes previous activities
io_edit.Combo1.AddItem output1                  'adds new activities to combo
io_edit.Combo1.AddItem output2
io_edit.Combo1.AddItem output3
io_edit.Combo1.AddItem output4
End Sub
```

Sub Save ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", io_edit.Text3.Text
        actornum = t1("number") 'number of actors
        actorname = t1("name")'name of actor which is edited
t1.Close
For i = 0 To actornum - 1
        Set t2 = db.OpenTable(actorname & i)
```

```
            t2.MoveFirst
            t2.Edit
            t2("st_io_code0") = ipt0
            t2("st_io_code1") = ipt1
            t2("st_io_code2") = ipt2
            t2("st_io_code3") = ipt3
            t2("st_io_code4") = ipt4
            t2("st_io_code5") = ipt5
            t2("st_io_code6") = ipt6
            t2("st_io_code7") = ipt7
            t2.Update
        t2.Close
Next i
db.Close
End Sub
```

Cogmap.bas

```
Option Explicit

Global CMarray(4, 32)
Dim parameters(4)
Dim CMarea As Double
Dim CMweigharea As Double
Dim IOMatrix(4, 32) As Integer
Dim CMM(4, 9, 6) As Double
Dim Utility(5, 4) As Single  'Utility,Reward,Xpos,Ypos*Alternatives
Global ActCount As Integer
Dim Alt As Integer
Global Const gridfactor = 10
```

Sub CalcParameters (Robot() As Actor)

```
' This Routine calculates the input parameters for the cognitive Map
' parameters(0)=distance, parameters(1)=payoff of activity in question, parameters(2)=remaining time
        for activity, parameters(3)=unused

Dim Alternative As Integer
Dim CellCount As Integer
Static position(3, 4) As Single
Static Distance(4)
Static reward(4)

For Alternative = 0 To 3' loop over alternatives in  CMknow
' calculate distance to alternatives
        Distance(Alternative) = Sqr((Robot(RobotCount).CMknow(ActCount, Alternative, 0) -
                Robot(RobotCount).XPos) ^ 2 + (Robot(RobotCount).CMknow(ActCount, Alternative, 1)
        - Robot(RobotCount).Ypos) ^ 2)
' get time specific payoff of that activity in that grid cell
        CellCount = 0
        Do Until ((Robot(RobotCount).CMknow(ActCount, Alternative, 0) =
        ThisWorld.XPos(CellCount))) And ((Robot(RobotCount).CMknow(ActCount,
        Alternative, 1) = ThisWorld.Ypos(CellCount)))
        CellCount = CellCount + 1
        Loop
        Select Case ActCount            'calculates value for shopping and socialising whilst taking the
                                        reward for recreation and work
        Case 0 Or 1
```

```
                reward(Alternative) = ThisWorld.Act(ActCount, TimeofDay, CellCount)
        Case Else
                reward(Alternative) = ThisWorld.Act(ActCount, TimeofDay, CellCount) /
                        ThisWorld.Price(ActCount, CellCount)
        End Select
        position(0, Alternative) = ThisWorld.XPos(CellCount)
        position(1, Alternative) = ThisWorld.Ypos(CellCount)
        position(2, Alternative) = CellCount
Next Alternative
For Alt = 0 To 3
        parameters(0) = reward(Alt)
        parameters(1) = Distance(Alt)
        parameters(2) = Robot(RobotCount).RemTime(ActCount)
        parameters(3) = Robot(RobotCount).CMknow(ActCount, Alt, 2)

        CMFuzzify

        Utility(1, Alt) = reward(Alt)
        Utility(2, Alt) = position(0, Alt)
        Utility(3, Alt) = position(1, Alt)
        Utility(4, Alt) = position(2, Alt)
Next Alt
                                        'Now look for max utility within utility(4)
                                        'finaldecisionVector(Actcount)=max(utility(4)
Select Case ((Utility(0, 0) >= Utility(0, 1)) And (Utility(0, 0) >= Utility(0, 2)) And (Utility(0, 0) >=
        Utility(0, 3)))
Case True
        Robot(RobotCount).FinalDecisionvector(0, ActCount) = Utility(0, 0)
        Robot(RobotCount).FinalDecisionvector(1, ActCount) = Utility(1, 0)
        Robot(RobotCount).FinalDecisionvector(2, ActCount) = Utility(2, 0)
        Robot(RobotCount).FinalDecisionvector(3, ActCount) = Utility(3, 0)
        Robot(RobotCount).FinalDecisionvector(4, ActCount) = Utility(4, 0)
        Robot(RobotCount).CurrentActivity(2) = 0
End Select
Select Case ((Utility(0, 1) > Utility(0, 0)) And (Utility(0, 1) >= Utility(0, 2)) And (Utility(0, 1) >=
        Utility(0, 3)))
Case True
        Robot(RobotCount).FinalDecisionvector(0, ActCount) = Utility(0, 1)
        Robot(RobotCount).FinalDecisionvector(1, ActCount) = Utility(1, 1)
        Robot(RobotCount).FinalDecisionvector(2, ActCount) = Utility(2, 1)
        Robot(RobotCount).FinalDecisionvector(3, ActCount) = Utility(3, 1)
        Robot(RobotCount).FinalDecisionvector(4, ActCount) = Utility(4, 1)
        Robot(RobotCount).CurrentActivity(2) = 1
End Select
Select Case ((Utility(0, 2) > Utility(0, 0)) And (Utility(0, 2) > Utility(0, 1)) And (Utility(0, 2) >=
        Utility(0, 3)))
Case True
        Robot(RobotCount).FinalDecisionvector(0, ActCount) = Utility(0, 2)
        Robot(RobotCount).FinalDecisionvector(1, ActCount) = Utility(1, 2)
        Robot(RobotCount).FinalDecisionvector(2, ActCount) = Utility(2, 2)
        Robot(RobotCount).FinalDecisionvector(3, ActCount) = Utility(3, 2)
        Robot(RobotCount).FinalDecisionvector(4, ActCount) = Utility(4, 2)
        Robot(RobotCount).CurrentActivity(2) = 2
End Select
Select Case ((Utility(0, 3) > Utility(0, 0)) And (Utility(0, 3) > Utility(0, 1)) And (Utility(0, 3) >
        Utility(0, 2)))
Case True
```

```
        Robot(RobotCount).FinalDecisionvector(0, ActCount) = Utility(0, 3)
        Robot(RobotCount).FinalDecisionvector(1, ActCount) = Utility(1, 3)
        Robot(RobotCount).FinalDecisionvector(2, ActCount) = Utility(2, 3)
        Robot(RobotCount).FinalDecisionvector(3, ActCount) = Utility(3, 3)
        Robot(RobotCount).FinalDecisionvector(4, ActCount) = Utility(4, 3)
        Robot(RobotCount).CurrentActivity(2) = 3
End Select
End Sub
```

Sub CMdefuzzify ()

```
Dim l As Integer

Utility(0, Alt) = 0
CMarea = 0
CMweigharea = 0
For l = 0 To 3                          'loop over rules
        CMM(l, 9, 5) = 0
        CMM(l, 9, 3) = 0
        CMM(l, 9, 5) = CMM(l, 9, 4) * CMM(l, 9, 2)
        CMM(l, 9, 3) = CMM(l, 9, 1) * CMM(l, 9, 4) * CMM(l, 9, 2)
        CMarea = CMarea + CMM(l, 9, 5)
        CMweigharea = CMweigharea + CMM(l, 9, 3)
Next l
If CMarea = 0 Then
        Utility(0, Alt) = 0
Else
        Utility(0, Alt) = CMweigharea / CMarea
End If
End Sub
```

Sub CMFuzzify ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
                                'this is supposed to fuzzify the Input value for each set

For i = 0 To 3              'loop over number of rules
        For l = 0 To 3      'parameters(0)=distance, par(1)=payoff, par(2)=remaining time
                For j = (2 * l) To (2 * l + 1)
                        Select Case j
                        Case 2 * l
                                CMM(i, j, 4) = 1 / (1 + (Exp((CMM(i, j, 2) * (parameters(l) - CMM(i, j,
                                        1))))))
                        Case 2 * l + 1
                                CMM(i, j, 4) = 1 / (1 + (Exp(-(CMM(i, j, 2) * (parameters(l) - CMM(i, j,
                                        1))))))
                        End Select
                Next j
        Next l
                                        'this is supposed to find the minimum of all active sets

        If CMM(i, 0, 5) = 1 Then        'checks whether first set is active or not
                CMM(i, 9, 4) = CMM(i, 0, 4)
        Else
                CMM(i, 9, 4) = 0
        End If
        For k = 1 To 8                  ' loop over number of sets
                If CMM(i, k, 5) = 1 And CMM(i, (k - 1), 6) = 0 Then
                        CMM(i, 9, 4) = CMM(i, k, 4)
                ElseIf CMM(i, k, 5) = 1 And CMM(i, (k - 1), 6) <> 0 And CMM(i, k, 4) < CMM(i, 9, 4)
                Then
                        CMM(i, 9, 4) = CMM(i, k, 4)
                End If
                CMM(i, k, 6) = CMM(i, (k - 1), 6) + CMM(i, k, 5)
        Next k
Next i
CMdefuzzify
End Sub
```

Sub DecodeRules (Robot() As Actor)

```
Dim k As Integer
Dim l As Integer
Dim test1 As Double
Dim test2 As Double
                                        'loop over all robots running
For ActCount = 0 To 3                   'loop over activities
        test1 = 0
        test2 = 0
        If ActCount = 0 Then
                test1 = Robot(RobotCount).CmIOCode(0)
                test2 = Robot(RobotCount).CmIOCode(1)
        ElseIf ActCount = 1 Then
                test1 = Robot(RobotCount).CmIOCode(2)
                test2 = Robot(RobotCount).CmIOCode(3)
        ElseIf ActCount = 2 Then
                test1 = Robot(RobotCount).CmIOCode(4)
                test2 = Robot(RobotCount).CmIOCode(5)
        ElseIf ActCount = 3 Then
                test1 = Robot(RobotCount).CmIOCode(6)
                test2 = Robot(RobotCount).CmIOCode(7)
        End If
        For k = 0 To 15
                If (test1 And 2 ^ k) = 2 ^ k Then
                        IOMatrix(ActCount, k) = 1
                Else IOMatrix(ActCount, k) = 0
                End If
        Next k
        For l = 0 To 15
                If (test2 And 2 ^ l) = 2 ^ l Then
                        IOMatrix(ActCount, l + 16) = 1
                        Else IOMatrix(ActCount, l + 16) = 0
                End If
        Next l
Next ActCount
End Sub
```

Sub FeedValues (Robot() As Actor)

```
                                        'passes the input parameters on to fuzzy cognitive map
Dim i As Integer
Dim j As Integer
Dim n As Integer

For i = 0 To 3
        For j = 0 To 7
                CMM(i, j, 3) = 1                'height of input sets
                CMM(i, j, 2) = Robot(RobotCount).CMFuzSigma(j)      'sigma of input sets
                CMM(i, j, 1) = Robot(RobotCount).CMFuzCentre(j)     'centre positon of input sets
                CMM(0, j, 5) = IOMatrix(i, j)
                CMM(1, j, 5) = IOMatrix(i, j + 8)
                CMM(2, j, 5) = IOMatrix(i, j + 16)
                CMM(3, j, 5) = IOMatrix(i, j + 24)
        Next j
        CMM(i, 9, 2) = Robot(RobotCount).CMDefuzSigma(i)        'comes from database and is stored
                                                                with robot's data
        CMM(i, 9, 1) = Robot(RobotCount).CMDefuzCentre(i) 'see above
        CMM(i, 9, 4) = 0
Next i
CMM(0, 0, 6) = CMM(0, 0, 5)                                     'copies On/Off values to next row
CMM(1, 0, 6) = CMM(1, 0, 5)
CMM(2, 0, 6) = CMM(2, 0, 5)
CMM(3, 0, 6) = CMM(3, 0, 5)
End Sub
```

Dataent.bas

```
Option Explicit
Dim TypeNum As Integer
Dim i As Integer
Dim Numcells As Integer
Dim Numactors As Integer

Global Const DB_BOOLEAN = 1
Global Const DB_BYTE = 2
Global Const DB_INTEGER = 3
Global Const DB_LONG = 4
Global Const DB_CURRENCY = 5
Global Const DB_SINGLE = 6
Global Const DB_DOUBLE = 7
Global Const DB_DATE = 8
Global Const DB_TEXT = 10
Global Const DB_LONGBINARY = 11
Global Const DB_MEMO = 12
Global Const DB_LANG_GENERAL = ";LANGID=0x0809;CP=1252;COUNTRY=0"
```

Sub tableadd ()

```
Dim db As database
Dim t As Table

Numactors = NameAct.Text2.Text
For TypeNum = 1 To 4
        For i = 0 To Numactors - 1
                Set db = OpenDatabase("c:\blah.mdb", True)
                        Dim NewTd As New TableDef
                        Dim f1 As New Field, f84 As New Field, f85 As New Field
                        Dim f2 As New Field, f86 As New Field, f87 As New Field
                        Dim f3 As New Field, f88 As New Field, f89 As New Field
                        Dim f4 As New Field, f90 As New Field, f91 As New Field
Dim f5 As New Field, f92 As New Field, f93 As New Field
Dim f6 As New Field, f94 As New Field, f95 As New Field
Dim f7 As New Field, f8 As New Field, f10 As New Field
Dim f12 As New Field, f14 As New Field, f16 As New Field
Dim f18 As New Field, f19 As New Field, f20 As New Field
Dim f21 As New Field, f22 As New Field, f23 As New Field
Dim f24 As New Field, f25 As New Field, f26 As New Field
Dim f27 As New Field, f28 As New Field, f29 As New Field
Dim f30 As New Field, f31 As New Field, f32 As New Field
Dim f33 As New Field, f34 As New Field, f35 As New Field
Dim f36 As New Field, f37 As New Field, f38 As New Field
Dim f39 As New Field, f40 As New Field, f41 As New Field
Dim f42 As New Field, f43 As New Field, f44 As New Field
Dim f45 As New Field, f46 As New Field, f48 As New Field
Dim f49 As New Field, f50 As New Field, f51 As New Field
Dim f52 As New Field, f53 As New Field, f55 As New Field
Dim f56 As New Field, f57 As New Field, f58 As New Field
Dim f59 As New Field, f60 As New Field, f61 As New Field
Dim f62 As New Field, f63 As New Field, f64 As New Field
Dim f65 As New Field, f66 As New Field, f67 As New Field
Dim f69 As New Field, f70 As New Field, f83 As New Field
Dim f71 As New Field, f72 As New Field, f73 As New Field
Dim f74 As New Field, f75 As New Field, f76 As New Field
Dim f77 As New Field, f78 As New Field, f79 As New Field
Dim f80 As New Field, f81 As New Field, f82 As New Field

NewTd.Name = TypeNum & i
f33.Name = "st_io_code0"
f33.Type = 7
NewTd.Fields.Append f33

f1.Name = "st_io_Code1"
f1.Type = 7
NewTd.Fields.Append f1

f30.Name = "st_io_code2"
f30.Type = 7
NewTd.Fields.Append f30

f31.Name = "st_io_code3"
f31.Type = 7
NewTd.Fields.Append f31

f32.Name = "st_io_code4"
f32.Type = 7
NewTd.Fields.Append f32

f34.Name = "st_io_code5"
f34.Type = 7
NewTd.Fields.Append f34

f35.Name = "st_io_code6"
f35.Type = 7
NewTd.Fields.Append f35

f36.Name = "st_io_code7"
f36.Type = 7
NewTd.Fields.Append f36

f2.Name = "st_fuz_centre_pos0"
```

```
f2.Type = 4
NewTd.Fields.Append f2

f4.Name = "st_fuz_centre_pos2"
f4.Type = 4
NewTd.Fields.Append f4

f6.Name = "st_fuz_centre_pos4"
f6.Type = 4
NewTd.Fields.Append f6

f8.Name = "st_fuz_centre_pos6"
f8.Type = 4
NewTd.Fields.Append f8

f10.Name = "st_fuz_sigma0"
f10.Type = 6
NewTd.Fields.Append f10

f12.Name = "st_fuz_sigma2"
f12.Type = 6
NewTd.Fields.Append f12

f14.Name = "st_fuz_sigma4"
f14.Type = 6
NewTd.Fields.Append f14

f16.Name = "st_fuz_sigma6"
f16.Type = 6
NewTd.Fields.Append f16

f18.Name = "st_defuz_centre0"
f18.Type = 3
NewTd.Fields.Append f18

f19.Name = "st_defuz_centre1"
f19.Type = 3
NewTd.Fields.Append f19

f20.Name = "st_defuz_centre2"
f20.Type = 3
NewTd.Fields.Append f20

f21.Name = "st_defuz_centre3"
f21.Type = 3
NewTd.Fields.Append f21

f22.Name = "st_defuz_sigma0"
f22.Type = 3
NewTd.Fields.Append f22

f23.Name = "st_defuz_sigma1"
f23.Type = 3
NewTd.Fields.Append f23

f24.Name = "st_defuz_sigma2"
f24.Type = 3
NewTd.Fields.Append f24

f25.Name = "st_defuz_sigma3"
f25.Type = 3
NewTd.Fields.Append f25

f26.Name = "st_budgetstate0"
```

```
f26.Type = 6
NewTd.Fields.Append f26

f27.Name = "st_budgetstate1"
f27.Type = 6
NewTd.Fields.Append f27

f28.Name = "st_budgetstate2"
f28.Type = 6
NewTd.Fields.Append f28

f29.Name = "st_budgetstate3"
f29.Type = 6
NewTd.Fields.Append f29

f37.Name = "st_acttime0"
f37.Type = 6
NewTd.Fields.Append f37

f38.Name = "st_acttime1"
f38.Type = 6
NewTd.Fields.Append f38

f39.Name = "st_acttime2"
f39.Type = 6
NewTd.Fields.Append f39

f40.Name = "st_acttime3"
f40.Type = 6
NewTd.Fields.Append f40

f41.Name = "st_success00"
f41.Type = DB_INTEGER
NewTd.Fields.Append f41

f42.Name = "st_success01"
f42.Type = DB_INTEGER
NewTd.Fields.Append f42

f43.Name = "st_success02"
f43.Type = DB_INTEGER
NewTd.Fields.Append f43

f44.Name = "st_success03"
f44.Type = DB_INTEGER
NewTd.Fields.Append f44

f84.Name = "st_success10"
f84.Type = DB_INTEGER
NewTd.Fields.Append f84

f85.Name = "st_success11"
f85.Type = DB_INTEGER
NewTd.Fields.Append f85

f86.Name = "st_success12"
f86.Type = DB_INTEGER
NewTd.Fields.Append f86

f87.Name = "st_success13"
f87.Type = DB_INTEGER
NewTd.Fields.Append f87

f88.Name = "st_success20"
```

```
f88.Type = DB_INTEGER
NewTd.Fields.Append f88

f89.Name = "st_success21"
f89.Type = DB_INTEGER
NewTd.Fields.Append f89

f95.Name = "st_success22"
f95.Type = DB_INTEGER
NewTd.Fields.Append f95

f90.Name = "st_success23"
f90.Type = DB_INTEGER
NewTd.Fields.Append f90

f91.Name = "st_success30"
f91.Type = DB_INTEGER
NewTd.Fields.Append f91

f92.Name = "st_success31"
f92.Type = DB_INTEGER
NewTd.Fields.Append f92

f93.Name = "st_success32"
f93.Type = DB_INTEGER
NewTd.Fields.Append f93

f94.Name = "st_success33"
f94.Type = DB_INTEGER
NewTd.Fields.Append f94

f45.Name = "CMIOCode0"
f45.Type = 7
NewTd.Fields.Append f45

f46.Name = "CMIOCode1"
f46.Type = 7
NewTd.Fields.Append f46

f53.Name = "CM_fuz_centre_pos0"
f53.Type = 6
NewTd.Fields.Append f53

f55.Name = "CM_fuz_centre_pos2"
f55.Type = 6
NewTd.Fields.Append f55

f57.Name = "CM_fuz_centre_pos4"
f57.Type = 6
NewTd.Fields.Append f57

f59.Name = "CM_fuz_centre_pos6"
f59.Type = 6
NewTd.Fields.Append f59

f61.Name = "CM_fuz_sigma0"
f61.Type = 6
NewTd.Fields.Append f61

f63.Name = "CM_fuz_sigma2"
f63.Type = 6
NewTd.Fields.Append f63

f65.Name = "CM_fuz_sigma4"
```

```
f65.Type = 6
NewTd.Fields.Append f65

f67.Name = "CM_fuz_sigma6"
f67.Type = 6
NewTd.Fields.Append f67

f69.Name = "CM_defuz_centre0"
f69.Type = 3
NewTd.Fields.Append f69

f70.Name = "CM_defuz_centre1"
f70.Type = 3
NewTd.Fields.Append f70

f71.Name = "CM_defuz_centre2"
f71.Type = 3
NewTd.Fields.Append f71

f72.Name = "CM_defuz_centre3"
f72.Type = 3
NewTd.Fields.Append f72

f73.Name = "CM_defuz_sigma0"
f73.Type = 3
NewTd.Fields.Append f73

f74.Name = "CM_defuz_sigma1"
f74.Type = 3
NewTd.Fields.Append f74

f75.Name = "CM_defuz_sigma2"
f75.Type = 3
NewTd.Fields.Append f75

f76.Name = "CM_defuz_sigma3"
f76.Type = 3
NewTd.Fields.Append f76

f77.Name = "CM_know0"
f77.Type = DB_TEXT
f77.Size = 32
NewTd.Fields.Append f77

f78.Name = "CM_know1"
f78.Type = DB_TEXT
f78.Size = 32
NewTd.Fields.Append f78

f79.Name = "CM_know2"
f79.Type = DB_TEXT
f79.Size = 32
NewTd.Fields.Append f79

f80.Name = "CM_know3"
f80.Type = DB_TEXT
f80.Size = 32
NewTd.Fields.Append f80

f81.Name = "XHome"
f81.Type = DB_INTEGER
NewTd.Fields.Append f81
```

```
        f82.Name = "YHome"
        f82.Type = DB_INTEGER
        NewTd.Fields.Append f82

        f3.Name = "learn_io_code0"
        f3.Type = 7
        NewTd.Fields.Append f3

        f5.Name = "learn_io_code1"
        f5.Type = 7
        NewTd.Fields.Append f5

        f7.Name = "LearnStrategy"
        f7.Type = DB_INTEGER
        NewTd.Fields.Append f7

        f48.Name = "learn_fuz_centre_pos2"
        f48.Type = 6
        NewTd.Fields.Append f48

        f49.Name = "learn_fuz_centre_pos4"
        f49.Type = 6
        NewTd.Fields.Append f49

        f50.Name = "learn_fuz_centre_pos6"
        f50.Type = 6
        NewTd.Fields.Append f50

        f51.Name = "learn_fuz_centre_pos8"
        f51.Type = 6
        NewTd.Fields.Append f51

        f52.Name = "learn_fuz_centre_pos10"
        f52.Type = 6
        NewTd.Fields.Append f52

        f56.Name = "learn_fuz_sigma2"
        f56.Type = 6
        NewTd.Fields.Append f56

        f58.Name = "learn_fuz_sigma4"
        f58.Type = 6
        NewTd.Fields.Append f58

        f60.Name = "learn_fuz_sigma6"
        f60.Type = 6
        NewTd.Fields.Append f60

        f62.Name = "learn_fuz_sigma8"
        f62.Type = 6
        NewTd.Fields.Append f62

        f64.Name = "learn_fuz_sigma10"
        f64.Type = 6
        NewTd.Fields.Append f64

        f83.Name = "Activities"
        f83.Type = DB_TEXT
        f83.Size = 72
        NewTd.Fields.Append f83
        db.TableDefs.Append NewTd
db.Close
Set db = OpenDatabase("c:\blah.mdb")  'writes default intial values to first record in table
Set t = db.OpenTable(TypeNum & i)
        t.AddNew

        t("st_io_code0") = 258
        t("st_io_code1") = 0
        t("st_io_code2") = 1032
        t("st_io_code3") = 0
        t("st_io_code4") = 4128
        t("st_io_code5") = 0
        t("st_io_code6") = 16512
        t("st_io_code7") = 0

        t("st_fuz_centre_pos0") = 5
        t("st_fuz_centre_pos2") = 30
        t("st_fuz_centre_pos4") = 15
        t("st_fuz_centre_pos6") = 5

        t("st_fuz_sigma0") = 1
        t("st_fuz_sigma2") = 1
        t("st_fuz_sigma4") = 1
        t("st_fuz_sigma6") = 1

        t("st_defuz_centre0") = 0
        t("st_defuz_centre1") = 100
        t("st_defuz_centre2") = 0
        t("st_defuz_centre3") = 100

        t("st_defuz_sigma0") = 15
        t("st_defuz_sigma1") = 15
        t("st_defuz_sigma2") = 15
        t("st_defuz_sigma3") = 15

        t("st_budgetstate0") = 0
        t("st_budgetstate1") = 0
        t("st_budgetstate2") = 0
        t("st_budgetstate3") = 0

        t("st_acttime0") = 0
        t("st_acttime1") = 0
        t("st_acttime2") = 0
        t("st_acttime3") = 0

        t("st_success00") = 0
        t("st_success01") = 0
        t("st_success02") = 0
        t("st_success03") = 0
        t("st_success10") = 0
        t("st_success11") = 0
        t("st_success12") = 0
        t("st_success13") = 0
        t("st_success20") = 0
        t("st_success21") = 0
        t("st_success22") = 0
        t("st_success23") = 0
        t("st_success30") = 0
        t("st_success31") = 0
        t("st_success32") = 0
        t("st_success33") = 0

        t("CMIOCode0") = 18822
```

```
                t("CMIOCode1") = 4128

                t("CM_fuz_centre_pos0") = .5
                t("CM_fuz_centre_pos2") = .5
                t("CM_fuz_centre_pos4") = 10
                t("CM_fuz_centre_pos6") = 0

                t("CM_fuz_sigma0") = 15
                t("CM_fuz_sigma2") = 1
                t("CM_fuz_sigma4") = .5
                t("CM_fuz_sigma6") = 1

                t("CM_defuz_centre0") = 100
                t("CM_defuz_centre1") = 0
                t("CM_defuz_centre2") = 100
                t("CM_defuz_centre3") = 0

                t("CM_defuz_sigma0") = 15
                t("CM_defuz_sigma1") = 15
                t("CM_defuz_sigma2") = 15
                t("CM_defuz_sigma3") = 15

                t("CM_know0") = "00000000"
                t("CM_know1") = "00000000"
                t("CM_know2") = "00000000"
                t("CM_know3") = "00000000"

                t("XHome") = 0
                t("YHome") = 0

                t("learn_io_code0") = 151578
                t("learn_io_code1") = 0
                t("LearnStrategy") = 0

                t("learn_fuz_centre_pos2") = 0
                t("learn_fuz_centre_pos4") = 0
                t("learn_fuz_centre_pos6") = 0
                t("learn_fuz_centre_pos8") = 0
                t("learn_fuz_centre_pos10") = 0

                t("learn_fuz_sigma2") = .3
                t("learn_fuz_sigma4") = .3
                t("learn_fuz_sigma6") = .3
                t("learn_fuz_sigma8") = .3
                t("learn_fuz_sigma10") = .3

                t("Activities") = ""
                t.Update
            t.Close
            db.Close
        Next i
Next TypeNum
End Sub
```

Sub wldadd ()

```
Dim db As database
Dim w As Table
Dim t2 As Table
Dim j As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t2 = db.OpenTable("world")
        t2.MoveLast
        Numcells = t2("count")
t2.Close
db.Close

For j = 0 To Numcells - 1
        Set db = OpenDatabase("c:\blah.mdb", True)
                Dim NewTd2 As New TableDef
                Dim w1 As New Field, w2 As New Field, w3 As New Field
                Dim w4 As New Field, w5 As New Field, w6 As New Field
                Dim w7 As New Field, w8 As New Field, w9 As New Field
                Dim w10 As New Field, w11 As New Field, w12 As New Field
                Dim w13 As New Field, w14 As New Field, w15 As New Field
                Dim w16 As New Field, w17 As New Field, w18 As New Field
                Dim w19 As New Field, w20 As New Field, w21 As New Field
                Dim w22 As New Field, w23 As New Field, w24 As New Field
                Dim w25 As New Field, w26 As New Field, w27 As New Field
                Dim w28 As New Field, w29 As New Field, w30 As New Field
                Dim w31 As New Field, w32 As New Field, w33 As New Field
                Dim w34 As New Field, w35 As New Field, w36 As New Field
                Dim w37 As New Field, w38 As New Field, w39 As New Field
                Dim w40 As New Field, w41 As New Field, w42 As New Field
                Dim w43 As New Field, w44 As New Field, w45 As New Field
                Dim w46 As New Field, w47 As New Field, w48 As New Field
                Dim w49 As New Field, w50 As New Field, w51 As New Field
                Dim w52 As New Field, w53 As New Field, w54 As New Field
                Dim w55 As New Field, w56 As New Field, w57 As New Field
                Dim w58 As New Field, w59 As New Field, w60 As New Field
                Dim w61 As New Field, w62 As New Field, w63 As New Field
                Dim w64 As New Field, w65 As New Field, w66 As New Field
                Dim w67 As New Field, w68 As New Field, w69 As New Field
                Dim w70 As New Field, w71 As New Field, w72 As New Field
                Dim w73 As New Field, w74 As New Field, w75 As New Field
                Dim w76 As New Field, w77 As New Field, w78 As New Field
                Dim w79 As New Field, w80 As New Field, w81 As New Field
                Dim w82 As New Field, w83 As New Field, w84 As New Field
                Dim w85 As New Field, w86 As New Field, w87 As New Field
                Dim w88 As New Field, w89 As New Field, w90 As New Field
                Dim w91 As New Field, w92 As New Field, w93 As New Field
                Dim w94 As New Field, w95 As New Field, w96 As New Field
                Dim w97 As New Field, w98 As New Field, w99 As New Field
                Dim w100 As New Field, w101 As New Field, w102 As New Field
                Dim w103 As New Field, w104 As New Field

                NewTd2.Name = "world" & j

                w3.Name = "cap_0"
                w3.Type = DB_INTEGER
                NewTd2.Fields.Append w3

                w1.Name = "price_0"
                w1.Type = DB_SINGLE
                NewTd2.Fields.Append w1

                w4.Name = "payoff0_0"
                w4.Type = DB_SINGLE
                NewTd2.Fields.Append w4
```

```
w5.Name = "payoff0_1"
w5.Type = DB_SINGLE
NewTd2.Fields.Append w5

w6.Name = "payoff0_2"
w6.Type = DB_SINGLE
NewTd2.Fields.Append w6

w7.Name = "payoff0_3"
w7.Type = DB_SINGLE
NewTd2.Fields.Append w7

w8.Name = "payoff0_4"
w8.Type = DB_SINGLE
NewTd2.Fields.Append w8

w9.Name = "payoff0_5"
w9.Type = DB_SINGLE
NewTd2.Fields.Append w9

w10.Name = "payoff0_6"
w10.Type = DB_SINGLE
NewTd2.Fields.Append w10

w11.Name = "payoff0_7"
w11.Type = DB_SINGLE
NewTd2.Fields.Append w11

w12.Name = "payoff0_8"
w12.Type = DB_SINGLE
NewTd2.Fields.Append w12

w13.Name = "payoff0_9"
w13.Type = DB_SINGLE
NewTd2.Fields.Append w13

w14.Name = "payoff0_10"
w14.Type = DB_SINGLE
NewTd2.Fields.Append w14

w15.Name = "payoff0_11"
w15.Type = DB_SINGLE
NewTd2.Fields.Append w15

w16.Name = "payoff0_12"
w16.Type = DB_SINGLE
NewTd2.Fields.Append w16

w17.Name = "payoff0_13"
w17.Type = DB_SINGLE
NewTd2.Fields.Append w17

w18.Name = "payoff0_14"
w18.Type = DB_SINGLE
NewTd2.Fields.Append w18

w19.Name = "payoff0_15"
w19.Type = DB_SINGLE
NewTd2.Fields.Append w19

w20.Name = "payoff0_16"
w20.Type = DB_SINGLE
NewTd2.Fields.Append w20
```

```
w21.Name = "payoff0_17"
w21.Type = DB_SINGLE
NewTd2.Fields.Append w21

w22.Name = "payoff0_18"
w22.Type = DB_SINGLE
NewTd2.Fields.Append w22

w23.Name = "payoff0_19"
w23.Type = DB_SINGLE
NewTd2.Fields.Append w23

w24.Name = "payoff0_20"
w24.Type = DB_SINGLE
NewTd2.Fields.Append w24

w25.Name = "payoff0_21"
w25.Type = DB_SINGLE
NewTd2.Fields.Append w25

w26.Name = "payoff0_22"
w26.Type = DB_SINGLE
NewTd2.Fields.Append w26

w27.Name = "payoff0_23"
w27.Type = DB_SINGLE
NewTd2.Fields.Append w27

w28.Name = "cap_1"
w28.Type = DB_INTEGER
NewTd2.Fields.Append w28

w2.Name = "price_1"
w2.Type = DB_SINGLE
NewTd2.Fields.Append w2

w29.Name = "payoff1_0"
w29.Type = DB_SINGLE
NewTd2.Fields.Append w29

w30.Name = "payoff1_1"
w30.Type = DB_SINGLE
NewTd2.Fields.Append w30

w31.Name = "payoff1_2"
w31.Type = DB_SINGLE
NewTd2.Fields.Append w31

w32.Name = "payoff1_3"
w32.Type = DB_SINGLE
NewTd2.Fields.Append w32

w33.Name = "payoff1_4"
w33.Type = DB_SINGLE
NewTd2.Fields.Append w33

w34.Name = "payoff1_5"
w34.Type = DB_SINGLE
NewTd2.Fields.Append w34

w35.Name = "payoff1_6"
w35.Type = DB_SINGLE
NewTd2.Fields.Append w35
```

```
w36.Name = "payoff1_7"
w36.Type = DB_SINGLE
NewTd2.Fields.Append w36

w37.Name = "payoff1_8"
w37.Type = DB_SINGLE
NewTd2.Fields.Append w37

w38.Name = "payoff1_9"
w38.Type = DB_SINGLE
NewTd2.Fields.Append w38

w39.Name = "payoff1_10"
w39.Type = DB_SINGLE
NewTd2.Fields.Append w39

w40.Name = "payoff1_11"
w40.Type = DB_SINGLE
NewTd2.Fields.Append w40

w41.Name = "payoff1_12"
w41.Type = DB_SINGLE
NewTd2.Fields.Append w41

w42.Name = "payoff1_13"
w42.Type = DB_SINGLE
NewTd2.Fields.Append w42

w43.Name = "payoff1_14"
w43.Type = DB_SINGLE
NewTd2.Fields.Append w43

w44.Name = "payoff1_15"
w44.Type = DB_SINGLE
NewTd2.Fields.Append w44

w45.Name = "payoff1_16"
w45.Type = DB_SINGLE
NewTd2.Fields.Append w45

w46.Name = "payoff1_17"
w46.Type = DB_SINGLE
NewTd2.Fields.Append w46

w47.Name = "payoff1_18"
w47.Type = DB_SINGLE
NewTd2.Fields.Append w47

w48.Name = "payoff1_19"
w48.Type = DB_SINGLE
NewTd2.Fields.Append w48

w49.Name = "payoff1_20"
w49.Type = DB_SINGLE
NewTd2.Fields.Append w49

w50.Name = "payoff1_21"
w50.Type = DB_SINGLE
NewTd2.Fields.Append w50

w51.Name = "payoff1_22"
w51.Type = DB_SINGLE
NewTd2.Fields.Append w51
```

```
w52.Name = "payoff1_23"
w52.Type = DB_SINGLE
NewTd2.Fields.Append w52

w53.Name = "cap_2"
w53.Type = DB_INTEGER
NewTd2.Fields.Append w53

w103.Name = "price_2"
w103.Type = DB_SINGLE
NewTd2.Fields.Append w103

w54.Name = "payoff2_0"
w54.Type = DB_SINGLE
NewTd2.Fields.Append w54

w55.Name = "payoff2_1"
w55.Type = DB_SINGLE
NewTd2.Fields.Append w55

w56.Name = "payoff2_2"
w56.Type = DB_SINGLE
NewTd2.Fields.Append w56

w57.Name = "payoff2_3"
w57.Type = DB_SINGLE
NewTd2.Fields.Append w57

w58.Name = "payoff2_4"
w58.Type = DB_SINGLE
NewTd2.Fields.Append w58

w59.Name = "payoff2_5"
w59.Type = DB_SINGLE
NewTd2.Fields.Append w59

w60.Name = "payoff2_6"
w60.Type = DB_SINGLE
NewTd2.Fields.Append w60

w61.Name = "payoff2_7"
w61.Type = DB_SINGLE
NewTd2.Fields.Append w61

w62.Name = "payoff2_8"
w62.Type = DB_SINGLE
NewTd2.Fields.Append w62

w63.Name = "payoff2_9"
w63.Type = DB_SINGLE
NewTd2.Fields.Append w63

w64.Name = "payoff2_10"
w64.Type = DB_SINGLE
NewTd2.Fields.Append w64

w65.Name = "payoff2_11"
w65.Type = DB_SINGLE
NewTd2.Fields.Append w65

w66.Name = "payoff2_12"
w66.Type = DB_SINGLE
NewTd2.Fields.Append w66
```

```
w67.Name = "payoff2_13"
w67.Type = DB_SINGLE
NewTd2.Fields.Append w67

w68.Name = "payoff2_14"
w68.Type = DB_SINGLE
NewTd2.Fields.Append w68

w69.Name = "payoff2_15"
w69.Type = DB_SINGLE
NewTd2.Fields.Append w69

w70.Name = "payoff2_16"
w70.Type = DB_SINGLE
NewTd2.Fields.Append w70

w71.Name = "payoff2_17"
w71.Type = DB_SINGLE
NewTd2.Fields.Append w71

w72.Name = "payoff2_18"
w72.Type = DB_SINGLE
NewTd2.Fields.Append w72

w73.Name = "payoff2_19"
w73.Type = DB_SINGLE
NewTd2.Fields.Append w73

w74.Name = "payoff2_20"
w74.Type = DB_SINGLE
NewTd2.Fields.Append w74

w75.Name = "payoff2_21"
w75.Type = DB_SINGLE
NewTd2.Fields.Append w75

w76.Name = "payoff2_22"
w76.Type = DB_SINGLE
NewTd2.Fields.Append w76

w77.Name = "payoff2_23"
w77.Type = DB_SINGLE
NewTd2.Fields.Append w77

w78.Name = "cap_3"
w78.Type = DB_INTEGER
NewTd2.Fields.Append w78

w104.Name = "price_3"
w104.Type = DB_SINGLE
NewTd2.Fields.Append w104

w79.Name = "payoff3_0"
w79.Type = DB_SINGLE
NewTd2.Fields.Append w79

w80.Name = "payoff3_1"
w80.Type = DB_SINGLE
NewTd2.Fields.Append w80

w81.Name = "payoff3_2"
w81.Type = DB_SINGLE
NewTd2.Fields.Append w81
```

```
w82.Name = "payoff3_3"
w82.Type = DB_SINGLE
NewTd2.Fields.Append w82

w83.Name = "payoff3_4"
w83.Type = DB_SINGLE
NewTd2.Fields.Append w83

w84.Name = "payoff3_5"
w84.Type = DB_SINGLE
NewTd2.Fields.Append w84

w85.Name = "payoff3_6"
w85.Type = DB_SINGLE
NewTd2.Fields.Append w85

w86.Name = "payoff3_7"
w86.Type = DB_SINGLE
NewTd2.Fields.Append w86

w87.Name = "payoff3_8"
w87.Type = DB_SINGLE
NewTd2.Fields.Append w87

w88.Name = "payoff3_9"
w88.Type = DB_SINGLE
NewTd2.Fields.Append w88

w89.Name = "payoff3_10"
w89.Type = DB_SINGLE
NewTd2.Fields.Append w89

w90.Name = "payoff3_11"
w90.Type = DB_SINGLE
NewTd2.Fields.Append w90

w91.Name = "payoff3_12"
w91.Type = DB_SINGLE
NewTd2.Fields.Append w91

w92.Name = "payoff3_13"
w92.Type = DB_SINGLE
NewTd2.Fields.Append w92

w93.Name = "payoff3_14"
w93.Type = DB_SINGLE
NewTd2.Fields.Append w93

w94.Name = "payoff3_15"
w94.Type = DB_SINGLE
NewTd2.Fields.Append w94

w95.Name = "payoff3_16"
w95.Type = DB_SINGLE
NewTd2.Fields.Append w95

w96.Name = "payoff3_17"
w96.Type = DB_SINGLE
NewTd2.Fields.Append w96

w97.Name = "payoff3_18"
w97.Type = DB_SINGLE
NewTd2.Fields.Append w97
```

```
w98.Name = "payoff3_19"
w98.Type = DB_SINGLE
NewTd2.Fields.Append w98

w99.Name = "payoff3_20"
w99.Type = DB_SINGLE
NewTd2.Fields.Append w99

w100.Name = "payoff3_21"
w100.Type = DB_SINGLE
NewTd2.Fields.Append w100

w101.Name = "payoff3_22"
w101.Type = DB_SINGLE
NewTd2.Fields.Append w101

w102.Name = "payoff3_23"
w102.Type = DB_SINGLE
NewTd2.Fields.Append w102

db.TableDefs.Append NewTd2
Set w = db.OpenTable("world" & j)
    w.AddNew
    w("cap_0") = 10
    w("cap_1") = 10
    w("cap_2") = 10
    w("cap_3") = 10

    w("price_0") = 1
    w("price_1") = 1
    w("price_2") = 2
    w("price_3") = 2

    w("payoff0_0") = 2
    w("payoff0_1") = 2
    w("payoff0_2") = 2
    w("payoff0_3") = 2
    w("payoff0_4") = 2
    w("payoff0_5") = 2
    w("payoff0_6") = 2
    w("payoff0_7") = .5
    w("payoff0_8") = .5
    w("payoff0_9") = .5
    w("payoff0_10") = .5
    w("payoff0_11") = .5
    w("payoff0_12") = .5
    w("payoff0_13") = .5
    w("payoff0_14") = .5
    w("payoff0_15") = .5
    w("payoff0_16") = .5
    w("payoff0_17") = .5
    w("payoff0_18") = .5
    w("payoff0_19") = 2
    w("payoff0_20") = 2
    w("payoff0_21") = 2
    w("payoff0_22") = 2
    w("payoff0_23") = 2

    w("payoff1_0") = 0
    w("payoff1_1") = 0
```

```
w("payoff1_2") = 0
w("payoff1_3") = 0
w("payoff1_4") = 0
w("payoff1_5") = 0
w("payoff1_6") = 0
w("payoff1_7") = 1
w("payoff1_8") = 1
w("payoff1_9") = 1
w("payoff1_10") = 1
w("payoff1_11") = 1
w("payoff1_12") = 1
w("payoff1_13") = 1
w("payoff1_14") = 1
w("payoff1_15") = 1
w("payoff1_16") = 1
w("payoff1_17") = 1
w("payoff1_18") = 1
w("payoff1_19") = 1
w("payoff1_20") = 1
w("payoff1_21") = 1
w("payoff1_22") = 1
w("payoff1_23") = 1

w("payoff2_0") = 0
w("payoff2_1") = 0
w("payoff2_2") = 0
w("payoff2_3") = 0
w("payoff2_4") = 0
w("payoff2_5") = 0
w("payoff2_6") = 0
w("payoff2_7") = 6
w("payoff2_8") = 6
w("payoff2_9") = 6
w("payoff2_10") = 6
w("payoff2_11") = 6
w("payoff2_12") = 6
w("payoff2_13") = 6
w("payoff2_14") = 6
w("payoff2_15") = 6
w("payoff2_16") = 6
w("payoff2_17") = 6
w("payoff2_18") = 6
w("payoff2_19") = 6
w("payoff2_20") = 6
w("payoff2_21") = 0
w("payoff2_22") = 0
w("payoff2_23") = 0

w("payoff3_0") = 0
w("payoff3_1") = 0
w("payoff3_2") = 0
w("payoff3_3") = 0
w("payoff3_4") = 0
w("payoff3_5") = 0
w("payoff3_6") = 0
w("payoff3_7") = 0
w("payoff3_8") = 0
```

```
                    w("payoff3_9") = 0
                    w("payoff3_10") = 0
                    w("payoff3_11") = 0
                    w("payoff3_12") = 0
                    w("payoff3_13") = 0
                    w("payoff3_14") = 2
                    w("payoff3_15") = 2
                    w("payoff3_16") = 2
                    w("payoff3_17") = 2
                    w("payoff3_18") = 2
                    w("payoff3_19") = 2
                    w("payoff3_20") = 2
                    w("payoff3_21") = 2
                    w("payoff3_22") = 2
                    w("payoff3_23") = 2

                    w.Update
            w.Close
        db.Close
Next j
NameAct.Show
End Sub
```

Dbstart.bas

```
Option Explicit
```

Sub addrecords ()

```
Dim db As Database, t As Table, t2 As Table

Set db = OpenDatabase("c:\blah.mdb", True)
Set t = db.OpenTable("typedescriptor")
        t.AddNew
        t("type") = 1
        t("number") = 1
        t("name") = 1
        t("st_bud_name1") = "sleep"
        t("st_bud_name2") = "money"
        t("st_bud_name3") = "goods"
        t("st_bud_name4") = "social"
        t("st_act_name1") = "recreation"
        t("st_act_name2") = "work"
        t("st_act_name3") = "shopping"
        t("st_act_name4") = "socialise"
        t.Update

        t.AddNew
        t("type") = 2
        t("number") = 1
        t("name") = 2
        t("st_bud_name1") = "sleep"
        t("st_bud_name2") = "money"
        t("st_bud_name3") = "goods"
        t("st_bud_name4") = "social"
        t("st_act_name1") = "recreation"
        t("st_act_name2") = "work"
        t("st_act_name3") = "shopping"
        t("st_act_name4") = "socialise"
        t.Update

        t.AddNew
        t("type") = 3
        t("number") = 1
        t("name") = 3
        t("st_bud_name1") = "sleep"
        t("st_bud_name2") = "money"
        t("st_bud_name3") = "goods"
        t("st_bud_name4") = "social"
        t("st_act_name1") = "recreation"
        t("st_act_name2") = "work"
        t("st_act_name3") = "shopping"
        t("st_act_name4") = "socialise"
        t.Update

        t.AddNew
        t("type") = 4
        t("number") = 1
        t("name") = 4
        t("st_bud_name1") = "sleep"
        t("st_bud_name2") = "money"
        t("st_bud_name3") = "goods"
        t("st_bud_name4") = "social"
        t("st_act_name1") = "recreation"
        t("st_act_name2") = "work"
        t("st_act_name3") = "shopping"
        t("st_act_name4") = "socialise"
        t.Update
t.Close
Set t2 = db.OpenTable("world")
        t2.AddNew
        t2("count") = 1
        t2("x_pos") = 0
        t2("y_pos") = 0
        t2.Update

        t2.AddNew
        t2("count") = 2
        t2("x_pos") = 10
        t2("y_pos") = 0
        t2.Update

        t2.AddNew
        t2("count") = 3
        t2("x_pos") = 20
        t2("y_pos") = 0
        t2.Update

        t2.AddNew
        t2("count") = 4
        t2("x_pos") = 0
        t2("y_pos") = 10
        t2.Update

        t2.AddNew
        t2("count") = 5
        t2("x_pos") = 10
        t2("y_pos") = 10
        t2.Update

        t2.AddNew
        t2("count") = 6
        t2("x_pos") = 20
        t2("y_pos") = 10
        t2.Update

        t2.AddNew
        t2("count") = 7
        t2("x_pos") = 0
        t2("y_pos") = 20
        t2.Update

        t2.AddNew
        t2("count") = 8
        t2("x_pos") = 10
        t2("y_pos") = 20
        t2.Update

        t2.AddNew
        t2("count") = 9
        t2("x_pos") = 20
        t2("y_pos") = 20
        t2.Update
t2.Close
db.Close
wldadd
End Sub
```

Sub blah ()

```
Dim db As Database
Dim NewTd As New TableDef
Dim WorldTd As New TableDef
Dim I1 As New Index, I2 As New Index

Dim f1 As New Field, f2 As New Field, f3 As New Field
Dim f4 As New Field, f5 As New Field, f6 As New Field
Dim f7 As New Field', f8 As New Field, f9 As New Field
Dim f12 As New Field
Dim f13 As New Field, f14 As New Field, f15 As New Field
Dim w0 As New Field, w1 As New Field, w2 As New Field

Set db = CreateDatabase("C:\BLAH.MDB", DB_LANG_GENERAL)
NewTd.Name = "typedescriptor"
WorldTd.Name = "world"

f1.Name = "type"
f1.Type = DB_INTEGER
NewTd.Fields.Append f1

f2.Name = "number"
f2.Type = DB_INTEGER
NewTd.Fields.Append f2

f3.Name = "name"
f3.Type = DB_TEXT
f3.Size = 10
NewTd.Fields.Append f3
```

```
f4.Name = "st_bud_name1"
f4.Type = DB_TEXT
f4.Size = 10
NewTd.Fields.Append f4

f5.Name = "st_bud_name2"
f5.Type = DB_TEXT
f5.Size = 10
NewTd.Fields.Append f5

f6.Name = "st_bud_name3"
f6.Type = DB_TEXT
f6.Size = 10
NewTd.Fields.Append f6

f7.Name = "st_bud_name4"
f7.Type = DB_TEXT
f7.Size = 10
NewTd.Fields.Append f7

f12.Name = "st_act_name1"
f12.Type = DB_TEXT
f12.Size = 10
NewTd.Fields.Append f12

f13.Name = "st_act_name2"
f13.Type = DB_TEXT
f13.Size = 10
NewTd.Fields.Append f13

f14.Name = "st_act_name3"
f14.Type = DB_TEXT
f14.Size = 10
NewTd.Fields.Append f14

f15.Name = "st_act_name4"
f15.Type = DB_TEXT
f15.Size = 10
NewTd.Fields.Append f15

I1.Name = "Type_Index"
I1.Fields = "type"
I1.Primary = True
NewTd.Indexes.Append I1

I2.Name = "Number_Index"
I2.Fields = "Number"
I2.Primary = False
NewTd.Indexes.Append I2

db.TableDefs.Append NewTd

w0.Name = "count"
w0.Type = DB_INTEGER
WorldTd.Fields.Append w0

w1.Name = "x_pos"
w1.Type = DB_SINGLE
WorldTd.Fields.Append w1

w2.Name = "y_pos"
w2.Type = DB_SINGLE
```

```
WorldTd.Fields.Append w2

db.TableDefs.Append WorldTd
db.Close
addrecords
End Sub
```

Display.bas

```
Option Explicit
Global TotNumber As Integer
Global ResArray()
Global CapArray()
Global PriArray()
Global TotalWealth()
Global BudArray()  As Single
Global BudMin As Single
Global BudMax As Single
Global PriMin As Single
Global PriMax As Single
Global CapMin As Single
Global CapMax As Single
Global WealthMin As Single
Global WealthMax As Single
Dim DisplayDemand(3, 8) As Single                    'activities and cells
Dim L_Number As Integer
Dim A_number As Integer
```

Sub CalcDem ()

```
Dim i As Integer
Dim j As Integer

For j = 0 To 3          'loop over activities, cells have to be stated explicitely
        If CapArray(BarChart.HScroll1.Value, 0, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 0, 0, j) = 0 Then
                DisplayDemand(j, 0) = 0
        ElseIf CapArray(BarChart.HScroll1.Value, 0, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 0, 0, j) <> 0 Then
                DisplayDemand(j, 0) = 9.99
        Else
                DisplayDemand(j, 0) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 0,
                0, j) / CapArray(BarChart.HScroll1.Value, 0, j)
        End If

        If CapArray(BarChart.HScroll1.Value, 1, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 1, 0, j) = 0 Then
                DisplayDemand(j, 1) = 0
        ElseIf CapArray(BarChart.HScroll1.Value, 1, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 1, 0, j) <> 0 Then
                DisplayDemand(j, 1) = 9.99
        Else
                DisplayDemand(j, 1) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 1,
                0, j) / CapArray(BarChart.HScroll1.Value, 1, j)
        End If

        If CapArray(BarChart.HScroll1.Value, 2, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 2, 0, j) = 0 Then
        DisplayDemand(j, 2) = 0
ElseIf CapArray(BarChart.HScroll1.Value, 2, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 2, 0, j) <> 0 Then
        DisplayDemand(j, 2) = 9.99
Else
        DisplayDemand(j, 2) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 2,
        0, j) / CapArray(BarChart.HScroll1.Value, 2, j)
End If

If CapArray(BarChart.HScroll1.Value, 3, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 0, 1, j) = 0 Then
        DisplayDemand(j, 3) = 0
ElseIf CapArray(BarChart.HScroll1.Value, 3, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 0, 1, j) <> 0 Then
        DisplayDemand(j, 3) = 9.99
Else
        DisplayDemand(j, 3) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 0,
        1, j) / CapArray(BarChart.HScroll1.Value, 3, j)
End If

If CapArray(BarChart.HScroll1.Value, 4, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 1, 1, j) = 0 Then
        DisplayDemand(j, 4) = 0
ElseIf CapArray(BarChart.HScroll1.Value, 4, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 1, 1, j) <> 0 Then
        DisplayDemand(j, 4) = 9.99
Else
        DisplayDemand(j, 4) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 1,
        1, j) / CapArray(BarChart.HScroll1.Value, 4, j)
End If

If CapArray(BarChart.HScroll1.Value, 5, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 2, 1, j) = 0 Then
        DisplayDemand(j, 5) = 0
ElseIf CapArray(BarChart.HScroll1.Value, 5, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 2, 1, j) <> 0 Then
        DisplayDemand(j, 5) = 9.99
Else
        DisplayDemand(j, 5) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 2,
        1, j) / CapArray(BarChart.HScroll1.Value, 5, j)
End If

If CapArray(BarChart.HScroll1.Value, 6, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 0, 2, j) = 0 Then
        DisplayDemand(j, 6) = 0
ElseIf CapArray(BarChart.HScroll1.Value, 6, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 0, 2, j) <> 0 Then
        DisplayDemand(j, 6) = 9.99
Else
        DisplayDemand(j, 6) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 0,
        2, j) / CapArray(BarChart.HScroll1.Value, 6, j)
End If

If CapArray(BarChart.HScroll1.Value, 7, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 1, 2, j) = 0 Then
        DisplayDemand(j, 7) = 0
ElseIf CapArray(BarChart.HScroll1.Value, 7, j) = 0 And ResArray(BarChart.HScroll1.Value,
BarChart.HScroll2.Value, 1, 2, j) <> 0 Then
```

```
            DisplayDemand(j, 7) = 9.99
        Else
            DisplayDemand(j, 7) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 1,
            2, j) / CapArray(BarChart.HScroll1.Value, 7, j)
        End If

        If CapArray(BarChart.HScroll1.Value, 8, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 2, 2, j) = 0 Then
            DisplayDemand(j, 8) = 0
        ElseIf CapArray(BarChart.HScroll1.Value, 8, j) = 0 And ResArray(BarChart.HScroll1.Value,
        BarChart.HScroll2.Value, 2, 2, j) <> 0 Then
            DisplayDemand(j, 8) = 9.99
        Else
            DisplayDemand(j, 8) = ResArray(BarChart.HScroll1.Value, BarChart.HScroll2.Value, 2,
            2, j) / CapArray(BarChart.HScroll1.Value, 8, j)
        End If
Next j
DrawCharts
End Sub
```

Sub displayactivity ()

```
Dim i As Integer
Dim Message As String
Dim j As Integer
Dim n As Variant
Dim m As Variant

Message = DisplayRes.Text1(0).Text
DisplayRes.Grid1.HighLight = False
For i = 1 To 3                          'clears grid display
    For j = 1 To 3
        DisplayRes.Grid1.Row = (i)
        DisplayRes.Grid1.Col = (j)
        DisplayRes.Grid1.Text = ""
    Next j
Next i
m = Mid(Message, 3 * DisplayRes.HScroll1.Value, 1)
n = Mid(Message, 3 * DisplayRes.HScroll1.Value - 1, 1)

DisplayRes.Grid1.Row = m + 1
DisplayRes.Grid1.Col = n + 1
DisplayRes.Grid1.SelStartRow = DisplayRes.Grid1.Row
DisplayRes.Grid1.SelEndRow = DisplayRes.Grid1.Row

DisplayRes.Grid1.SelStartCol = DisplayRes.Grid1.Col
DisplayRes.Grid1.SelEndCol = DisplayRes.Grid1.Col
DisplayRes.Grid1.HighLight = True
DisplayRes.Grid1.Text = Mid(Message, 3 * DisplayRes.HScroll1.Value - 2, 1)
Select Case DisplayRes.Grid1.Text
Case "A"
    DisplayRes.Grid1.BackColor = &HFFFF00
Case "B"
    DisplayRes.Grid1.BackColor = &HC0&
Case "C"
    DisplayRes.Grid1.BackColor = &HFFFF&
Case "D"
    DisplayRes.Grid1.BackColor = &HFF00FF
```

```
End Select
End Sub
```

Sub DrawBudgets ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim Y_factor As Single
Dim X_factor As Single

BudGraph.Picture1.Cls
BudGraph.Picture1.Line (700, 200)-(700, 3400)  'y-axis
For i = 0 To 9
        BudGraph.Label7(i).Caption = Int((L_Number - 1) / 10 * (10 - i))
Next i
For j = 0 To 5
        BudGraph.Label3(j).Caption = Left(BudMin + (((5 - j) / 5) * (BudMax - BudMin)), 6)
Next j
Y_factor = 3200 / (BudMax - BudMin)
X_factor = 10400 / (L_Number)
BudGraph.Picture1.Line (700, 3400 - Y_factor * Abs(BudMin))-(11200, 3400 - Y_factor *
Abs(BudMin))
For l = 0 To 3
        BudGraph.Picture1.CurrentX = 700
        BudGraph.Picture1.CurrentY = 3400 - (BudArray(0, BudGraph.HScroll1.Value,
        BudGraph.HScroll2.Value, l) * Y_factor) - Y_factor * Abs(BudMin)
        For k = 1 To L_Number - 1
                BudGraph.Picture1.Line -((X_factor * k) + 700, 3400 - BudArray(k,
                BudGraph.HScroll1.Value, BudGraph.HScroll2.Value, l) * Y_factor - Y_factor *
                Abs(BudMin)), QBColor(2 * l)
        Next k
Next l
End Sub
```

Sub DrawCap ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim Y_factor As Single
Dim X_factor As Single

BudGraph.Picture1.Cls
BudGraph.Picture1.Line (700, 200)-(700, 3400)  'y-axis
For i = 0 To 9
        BudGraph.Label7(i).Caption = Int((L_Number - 1) / 10 * (10 - i))
Next i
For j = 0 To 5
        BudGraph.Label3(j).Caption = Left(CapMin + (((5 - j) / 5) * (CapMax - CapMin)), 6)
Next j
Y_factor = 3200 / (CapMax - CapMin)
X_factor = 10400 / (L_Number)
BudGraph.Picture1.Line (700, 3400 - Y_factor * Abs(CapMin))-(11200, 3400 - Y_factor *
Abs(CapMin))
For l = 0 To 3
```

```
        BudGraph.Picture1.CurrentX = 700
        BudGraph.Picture1.CurrentY = 3400 - (CapArray(0, BudGraph.HScroll3.Value, l) * Y_factor) -
        Y_factor * Abs(CapMin)
        For k = 1 To L_Number - 1
                BudGraph.Picture1.Line -((X_factor * k) + 700, 3400 - CapArray(k,
                BudGraph.HScroll3.Value, l) * Y_factor - Y_factor * Abs(CapMin)), QBColor(2 * l)
        Next k
Next l
End Sub
```

Sub DrawCharts ()

```
'calculate relative demand
'move to first pic box
'draw coordinate system
'draw four barcharts (activities)

Dim i As Integer

For i = 0 To 8
        BarChart.Picture1(i).Cls
        BarChart.Picture1(i).Line (200, 200)-(200, 1800)                'coordinate system
        BarChart.Picture1(i).Line (200, 1800)-(2500, 1800)
        BarChart.Picture1(i).Line (300, 1800)-(775, 1800 - DisplayDemand(0, i) * 500), QBColor(0), BF
        BarChart.Picture1(i).Line (875, 1800)-(1350, 1800 - DisplayDemand(1, i) * 500), QBColor(2),
        BF
        BarChart.Picture1(i).Line (1450, 1800)-(1925, 1800 - DisplayDemand(2, i) * 500), QBColor(4),
        BF
        BarChart.Picture1(i).Line (2025, 1800)-(2500, 1800 - DisplayDemand(3, i) * 500), QBColor(6),
        BF
        BarChart.Label5(i).Caption = Left(DisplayDemand(0, i) * 100, 3) & "%"
        BarChart.Label6(i).Caption = Left(DisplayDemand(1, i) * 100, 3) & "%"
        BarChart.Label7(i).Caption = Left(DisplayDemand(2, i) * 100, 3) & "%"
        BarChart.Label8(i).Caption = Left(DisplayDemand(3, i) * 100, 3) & "%"
Next i
End Sub
```

Sub DrawPri ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim Y_factor As Single
Dim X_factor As Single

BudGraph.Picture1.Cls
BudGraph.Picture1.Line (700, 200)-(700, 3400)  'y-axis
For i = 0 To 9
        BudGraph.Label7(i).Caption = Int((L_Number - 1) / 10 * (10 - i))
Next i
For j = 0 To 5
        BudGraph.Label3(j).Caption = Left(PriMin + (((5 - j) / 5) * (PriMax - PriMin)), 6)
Next j
Y_factor = 3200 / (PriMax - PriMin)
X_factor = 10400 / (L_Number)
BudGraph.Picture1.Line (700, 3400 - Y_factor * Abs(PriMin))-(11200, 3400 - Y_factor * Abs(PriMin))
For l = 0 To 3
```

```
        BudGraph.Picture1.CurrentX = 700
        BudGraph.Picture1.CurrentY = 3400 - (PriArray(0, BudGraph.HScroll3.Value, l) * Y_factor) -
        Y_factor * Abs(PriMin)
        For k = 1 To L_Number - 1
                BudGraph.Picture1.Line -((X_factor * k) + 700, 3400 - PriArray(k,
                BudGraph.HScroll3.Value, l) * Y_factor - Y_factor * Abs(PriMin)), QBColor(2 * l)
        Next k
Next l
End Sub
```

Sub DrawRelWealth ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim Y_factor As Single
Dim X_factor As Single
Dim RelMin As Single
Dim RelMax As Single

RelMin = -.5
RelMax = .5
BudGraph.Picture1.Cls
BudGraph.Picture1.Line (700, 200)-(700, 3400)  'y-axis
For i = 0 To 9
        BudGraph.Label7(i).Caption = Int((L_Number - 1) / 10 * (10 - i))   ' labels on x-axis
Next i
For j = 0 To 5
        BudGraph.Label3(j).Caption = Left(RelMin + (((5 - j) / 5) * (RelMax - RelMin)), 6)
Next j
Y_factor = 3200 / (RelMax - RelMin)
X_factor = 10400 / (L_Number)

BudGraph.Picture1.Line (700, 3400 - Y_factor * Abs(RelMin))-(11200, 3400 - Y_factor *  Abs(RelMin))
        'draw x-axis
BudGraph.Picture1.Line (700, 3400 - Y_factor * TotalWealth(0, 0) / (4 * TotNumber))-(11200, 3400 -
        Y_factor * TotalWealth(0, 0) / (4 * TotNumber))'draw theoretical share of agent
For l = 0 To 3
        BudGraph.Picture1.CurrentX = 700
        BudGraph.Picture1.CurrentY = 3400 - ((BudArray(0, BudGraph.HScroll1.Value,
                BudGraph.HScroll2.Value, l) * Y_factor) - Y_factor * Abs(RelMin)) / (TotalWealth(0, l))
        For k = 1 To L_Number - 1
                BudGraph.Picture1.Line -((X_factor * k) + 700, 3400 - (BudArray(k,
                BudGraph.HScroll1.Value, BudGraph.HScroll2.Value, l) * Y_factor /
                TotalWealth(k, l)) - Y_factor * Abs(RelMin)), QBColor(2 * l)
        Next k
Next l
End Sub
```

Sub DrawWealth ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim Y_factor As Single
Dim X_factor As Single
```

```
BudGraph.Picture1.Cls
BudGraph.Picture1.Line (700, 200)-(700, 3400)  'y-axis
For i = 0 To 9
        BudGraph.Label7(i).Caption = Int((L_Number - 1) / 10 * (10 - i))        'labels on x-axis
Next i
For j = 0 To 5
        BudGraph.Label3(j).Caption = Left(WealthMin + (((5 - j) / 5) * (WealthMax - WealthMin)), 6)
Next j
Y_factor = 3200 / (WealthMax - WealthMin)
X_factor = 10400 / (L_Number)
BudGraph.Picture1.Line (700, 3400 - Y_factor * Abs(WealthMin))-(11200, 3400 - Y_factor *
        Abs(WealthMin))             'draw x-axis
For l = 0 To 4
        BudGraph.Picture1.CurrentX = 700
        BudGraph.Picture1.CurrentY = 3400 - ((TotalWealth(0, l) * Y_factor) - Y_factor *
        Abs(WealthMin))
        For k = 1 To L_Number - 1
                BudGraph.Picture1.Line -((X_factor * k) + 700, 3400 - (TotalWealth(k, l) * Y_factor) -
                        Y_factor * Abs(WealthMin)), QBColor(2 * l)
        Next k
Next l
End Sub
```

Sub FindCapPriMinMax ()

```
Dim i As Integer
Dim j As Integer

PriMin = PriArray(0, 1, 0)
PriMax = PriArray(0, 1, 0)
CapMin = CapArray(0, 1, 0)
CapMax = CapArray(0, 1, 0)
For i = 0 To L_Number
        For j = 0 To 3
                If PriArray(i, BudGraph.HScroll3.Value, j) > PriMax Then
                        PriMax = PriArray(i, BudGraph.HScroll3.Value, j)
                End If
                If PriArray(i, BudGraph.HScroll3.Value, j) < PriMin Then
                        PriMin = PriArray(i, BudGraph.HScroll3.Value, j)
                End If
                If CapArray(i, BudGraph.HScroll3.Value, j) > CapMax Then
                        CapMax = CapArray(i, BudGraph.HScroll3.Value, j)
                End If
                If CapArray(i, BudGraph.HScroll3.Value, j) < CapMin Then
                        CapMin = CapArray(i, BudGraph.HScroll3.Value, j)
                End If
        Next j
Next i
End Sub
```

Sub FindMinMax ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer

BudMin = BudArray(0, 1, 0, 0)
```

```
BudMax = BudArray(0, 1, 0, 0)
For k = 0 To L_Number
        For l = 0 To 3
                If BudArray(k, BudGraph.HScroll1.Value, BudGraph.HScroll2.Value, l) < BudMin Then
                        BudMin = BudArray(k, BudGraph.HScroll1.Value, BudGraph.HScroll2.Value, l)
                End If
                If BudArray(k, BudGraph.HScroll1.Value, BudGraph.HScroll2.Value, l) > BudMax Then
                        BudMax = BudArray(k, BudGraph.HScroll1.Value, BudGraph.HScroll2.Value, l)
                End If
        Next l
Next k
End Sub
```

Sub FindWealthMinMax ()

```
Dim i As Integer
Dim j As Integer

WealthMin = TotalWealth(0, 0)
WealthMax = TotalWealth(0, 0)
For i = 0 To L_Number
        For j = 0 To 4
                If TotalWealth(i, j) > WealthMax Then
                        WealthMax = TotalWealth(i, j)
                End If
                If TotalWealth(i, j) < WealthMin Then
                        WealthMin = TotalWealth(i, j)
                End If
        Next j
Next i
End Sub
```

Sub LoadActivity ()

```
Dim db As Database
Dim t1 As Table
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim s As Integer
Dim t As Integer
Dim u As Integer
Dim TotNumber As Integer
Dim n As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.MoveFirst
        TotNumber = t1("number")
t1.Close
n = 0
Set t1 = db.OpenTable("10")
        t1.MoveFirst
        Do Until t1.EOF
                t1.MoveNext
                n = n + 1
        Loop
```

```
t1.Close
ReDim ResArray(n, 23, 2, 2, 3)
        'number of iterations in db, time of day, xcoordinates, ycoordinates, 4 activities
ReDim CapArray(n, 8, 3)             'number of iterations, cellnumber,activities
ReDim PriArray(n, 8, 3)
BarChart.HScroll1.Max = n
For i = 0 To TotNumber - 1
        For l = 1 To 4
                Set t1 = db.OpenTable(l & i)
                        t1.MoveLast
                        For k = n To 1 Step -1
                                For j = 1 To 24
                                        Select Case Mid(t1("activities"), 3 * j - 2, 1) = "A"
                                        Case True
                                                ResArray(k, j - 1, Mid(t1("activities"), 3 * j - 1, 1),
                                                Mid(t1("activities"), 3 * j, 1), 0) = ResArray(k, j - 1,
                                                Mid(t1("activities"), 3 * j - 1, 1), Mid(t1("activities"), 3
                                        * j, 1), 0) + 1
                                        End Select
                                        Select Case Mid(t1("activities"), 3 * j - 2, 1) = "B"
                                        Case True
                                                ResArray(k, j - 1, Mid(t1("activities"), 3 * j - 1, 1),
                                                Mid(t1("activities"), 3 * j, 1), 1) = ResArray(k, j - 1,
                                                Mid(t1("activities"), 3 * j - 1, 1), Mid(t1("activities"), 3
                                        * j, 1), 1) + 1
                                        End Select
                                        Select Case Mid(t1("activities"), 3 * j - 2, 1) = "C"
                                        Case True
                                                ResArray(k, j - 1, Mid(t1("activities"), 3 * j - 1, 1),
                                                Mid(t1("activities"), 3 * j, 1), 2) = ResArray(k, j - 1,
                                                Mid(t1("activities"), 3 * j - 1, 1), Mid(t1("activities"), 3
                                        * j, 1), 2) + 1
                                        End Select
                                        Select Case Mid(t1("activities"), 3 * j - 2, 1) = "D"
                                        Case True
                                                ResArray(k, j - 1, Mid(t1("activities"), 3 * j - 1, 1),
                                                Mid(t1("activities"), 3 * j, 1), 3) = ResArray(k, j - 1,
                                                Mid(t1("activities"), 3 * j - 1, 1), Mid(t1("activities"), 3
                                        * j, 1), 3) + 1
                                        End Select
                                Next j
                                t1.MovePrevious
                        Next k
                t1.Close
        Next l
Next i
                                        'get capacity from world tables
For s = 0 To 8
        Set t1 = db.OpenTable("world" & s)
                t1.MoveFirst
                t1.MoveNext
                u = 1
                Do Until t1.EOF
                        For t = 0 To 3
                                CapArray(u, s, t) = t1("cap_" & t)  'iteration number, cellnumber, activity
                                PriArray(u, s, t) = t1("price_" & t)
                        Next t
                        u = u + 1
                        t1.MoveNext
                Loop
        t1.Close
Next s
End Sub
```

Sub LoadBudgets ()

```
Dim db As Database
Dim t1 As Table
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim n As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.MoveFirst
        TotNumber = t1("number")
t1.Close
A_number = TotNumber
n = 0
Set t1 = db.OpenTable("10")
        t1.MoveFirst
        Do Until t1.EOF  'get number of timesteps
                t1.MoveNext
                n = n + 1
        Loop
t1.Close
L_Number = n

ReDim BudArray(n, 4, TotNumber - 1, 3)          'timesteps, Type, tablenumber, budgets
ReDim TotalWealth(n, 4)
ReDim CapArray(n, 8, 3)                                 'number of iterations, cellnumber,activities
ReDim PriArray(n, 8, 3)
BudGraph.HScroll2.Max = TotNumber - 1
For i = 1 To 4                                  'loop over types
        For j = 0 To TotNumber - 1              'loop over agents of type
                Set t1 = db.OpenTable(i & j)
                        t1.MoveFirst
                        For k = 0 To n - 1              'loop over number of timesteps
                                For l = 0 To 3          'loop over budgets
                                        BudArray(k, i, j, l) = t1("st_budgetstate" & l)
                                Next l
                                t1.MoveNext
                        Next k
                t1.Close
        Next j
Next i
db.Close
End Sub
```

Sub LoadCapPri ()

```
Dim s As Integer
Dim t As Integer
Dim u As Integer
Dim db As Database
Dim t1 As Table

Set db = OpenDatabase("c:\blah.mdb", True)                'get capacity from world tables
For s = 0 To 8
    Set t1 = db.OpenTable("world" & s)
        t1.MoveFirst
        t1.MoveNext
        u = 1
        Do Until t1.EOF
            For t = 0 To 3
                CapArray(u, s, t) = t1("cap_" & t)  'iteration number, cellnumber, activity
                PriArray(u, s, t) = t1("price_" & t)
            Next t
            u = u + 1
            t1.MoveNext
        Loop
    t1.Close
Next s
db.Close
End Sub
```

Sub RelWealth ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim n As Integer

For k = 0 To 4
    For n = 0 To L_Number - 1
        TotalWealth(n, k) = 0                'reset totalwealth
    Next n
Next k
For k = 0 To 3                                          'loop over budgets
    For n = 0 To L_Number - 1                           'loop over length of simulation
            For j = 1 To 4                     'loop over types of agents
                For i = 0 To TotNumber - 1 'loop over all agents of one type
                    TotalWealth(n, k) = BudArray(n, j, i, k) + TotalWealth(n, k)
                Next i
            Next j
    Next n
Next k
For n = 0 To L_Number - 1
    TotalWealth(n, 4) = TotalWealth(n, 0) + TotalWealth(n, 1) + TotalWealth(n, 2) +
        TotalWealth(n, 3)
Next n
End Sub
```

Fuzzrule.bas

```
Option Explicit

Global IOarray(4, 32) As Integer
Global LearnIO(4, 48) As Integer
Global InputNum(4) As Single
Global LearnInput(6) As Single
Dim area As Single
Dim weigharea As Single
Dim Larea As Single
Dim Lweigharea As Single
Dim BRB(4, 9, 6) 'As Double
Dim LRB(4, 13, 6)
```

Sub DecodeLearnRules (Robot() As Actor)

```
Dim k As Integer
Dim l As Integer
Dim test1 As Double
Dim test2 As Double

'loop over all robots running

For ActCount = 0 To 3                    'loop over activities
        test1 = 0
        test2 = 0
        If ActCount = 0 Then
                test1 = Robot(RobotNum).LearnIOCode(0)
                test2 = Robot(RobotNum).LearnIOCode(1)
        ElseIf ActCount = 1 Then
                test1 = Robot(RobotNum).LearnIOCode(2)
                test2 = Robot(RobotNum).LearnIOCode(3)
        ElseIf ActCount = 2 Then
                test1 = Robot(RobotNum).LearnIOCode(4)
                test2 = Robot(RobotNum).LearnIOCode(5)
        ElseIf ActCount = 3 Then
                test1 = Robot(RobotNum).LearnIOCode(6)
                test2 = Robot(RobotNum).LearnIOCode(7)
        End If
        For k = 0 To 23
                If (test1 And 2 ^ k) = 2 ^ k Then
                        LearnIO(ActCount, k) = 1
                Else LearnIO(ActCount, k) = 0
                End If
        Next k
        For l = 0 To 23
                If (test2 And 2 ^ l) = 2 ^ l Then
                        LearnIO(ActCount, l + 24) = 1
                Else LearnIO(ActCount, l + 24) = 0
                End If
        Next l
Next ActCount
End Sub
```

Sub Defuzzify ()

```
Dim l As Integer

area = 0
weigharea = 0
For l = 0 To 3          'loop over rules
        BRB(l, 9, 5) = 0
        BRB(l, 9, 3) = 0
        BRB(l, 9, 5) = BRB(l, 9, 4) * BRB(l, 9, 2)
        BRB(l, 9, 3) = BRB(l, 9, 1) * BRB(l, 9, 4) * BRB(l, 9, 2)
        area = area + BRB(l, 9, 5)
        weigharea = weigharea + BRB(l, 9, 3)
Next l
If area = 0 Then
        importance(Actinum) = 0
Else importance(Actinum) = weigharea / area
End If
End Sub
```

Sub Fuzzify ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer

'this is supposed to fuzzify the Input value for each set

For i = 0 To 3                  'loop over number of rules
        For l = 0 To 3
                For j = (2 * l) To (2 * l + 1)
                        Select Case j
                        Case 2 * l
                                If BRB(i, j, 2) * (InputNum(l) - BRB(i, j, 1)) > 10000 Then
                                        BRB(i, j, 4) = 0
                                Else
                                        BRB(i, j, 4) = 1 / (1 + (Exp((BRB(i, j, 2) * (InputNum(l) - BRB(i, j,
                                        1))))))
                                End If
                        Case 2 * l + 1
                                If BRB(i, j, 2) * (InputNum(l) - BRB(i, j, 1)) < -10000 Then
                                        BRB(i, j, 4) = 1
                                Else
                                        BRB(i, j, 4) = 1 / (1 + (Exp(-(BRB(i, j, 2) * (InputNum(l) - BRB(i, j,
                                1))))))
                                End If
                        End Select
                Next j
        Next l

'this is supposed to find the minimum of all active sets

        If BRB(i, 0, 5) = 1 Then                'checks whether first set is active or not
                BRB(i, 9, 4) = BRB(i, 0, 4)
        Else BRB(i, 9, 4) = 0
        End If
        For k = 1 To 8                          'loop over number of sets
                If BRB(i, k, 5) = 1 And BRB(i, (k - 1), 6) = 0 Then
                        BRB(i, 9, 4) = BRB(i, k, 4)
                ElseIf BRB(i, k, 5) = 1 And BRB(i, (k - 1), 6) <> 0 And BRB(i, k, 4) < BRB(i, 9, 4) Then
                        BRB(i, 9, 4) = BRB(i, k, 4)
                End If
                BRB(i, k, 6) = BRB(i, (k - 1), 6) + BRB(i, k, 5)
        Next k
Next i
End Sub
```

Sub GetLearnValues (Robot() As Actor)

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim n As Integer

LearnInput(0) = Robot(RobotNum).ShortTermBudgets(Actinum)
If WriteNum < 10 Then
        LearnInput(1) = Robot(RobotNum).CentreMatrix(WriteNum, Actinum) -
                Robot(RobotNum).CentreMatrix(WriteNum + 290, Actinum)
Else LearnInput(1) = Robot(RobotNum).CentreMatrix(WriteNum, Actinum) -
        Robot(RobotNum).CentreMatrix(WriteNum - 10, Actinum)
End If
If WriteNum < 10 Then
        LearnInput(2) = Robot(RobotNum).CentreMatrix(WriteNum, MuteAlt) -
                Robot(RobotNum).CentreMatrix(WriteNum + 290, MuteAlt)
Else LearnInput(2) = Robot(RobotNum).CentreMatrix(WriteNum, MuteAlt) -
        Robot(RobotNum).CentreMatrix(WriteNum - 10, MuteAlt)
End If
LearnInput(3) = 0
LearnInput(4) = 0
LearnInput(5) = 0
For i = 0 To 3
        For j = 0 To 1
                LRB(i, j, 2) = Robot(RobotNum).ShortTermFuzSigma(j)          'sigma of input sets
                LRB(i, j, 1) = Robot(RobotNum).ShortTermFuzCentre(j)         'centre positon of input sets
        Next j
        For j = 2 To 11
                LRB(i, j, 2) = Robot(RobotNum).LearnSigma(j)
                LRB(i, j, 1) = Robot(RobotNum).LearnCentre(j)
        Next j
        For j = 0 To 11
                LRB(i, j, 3) = 1                         'height of input sets
                LRB(0, j, 5) = LearnIO(Actinum, j)
                LRB(1, j, 5) = LearnIO(Actinum, j + 12)
                LRB(2, j, 5) = LearnIO(Actinum, j + 24)
                LRB(3, j, 5) = LearnIO(Actinum, j + 36)
        Next j
        LRB(i, 13, 2) = Robot(RobotNum).ShortTermDefuzSigma(i)         'comes from database and
                                                                      is stored with robot's data
        LRB(i, 13, 1) = Robot(RobotNum).ShortTermDefuzCentre(i)
        LRB(i, 13, 4) = 0
Next i
LRB(0, 0, 6) = LRB(0, 0, 5)
LRB(1, 0, 6) = LRB(1, 0, 5)
LRB(2, 0, 6) = LRB(2, 0, 5)
LRB(3, 0, 6) = LRB(3, 0, 5)
End Sub
```

Sub GetValues (Robot() As Actor)

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim n As Integer

For n = 0 To 3
```

```
        InputNum(n) = Robot(RobotNum).ShortTermBudgets(n)       'this is to be the budget state of
                                                                robot in question!
Next n
For i = 0 To 3
        For j = 0 To 7
                BRB(i, j, 3) = 1                                'height of input sets
                BRB(i, j, 2) = Robot(RobotNum).ShortTermFuzSigma(j)     'sigma of input sets
                BRB(i, j, 1) = Robot(RobotNum).ShortTermFuzCentre(j)    'centre positon of input sets
                BRB(0, j, 5) = IOarray(Actinum, j)
                BRB(1, j, 5) = IOarray(Actinum, j + 8)
                BRB(2, j, 5) = IOarray(Actinum, j + 16)
                BRB(3, j, 5) = IOarray(Actinum, j + 24)
        Next j
        BRB(i, 9, 2) = Robot(RobotNum).ShortTermDefuzSigma(i)
        BRB(i, 9, 1) = Robot(RobotNum).ShortTermDefuzCentre(i)
        BRB(i, 9, 4) = 0
Next i
BRB(0, 0, 6) = BRB(0, 0, 5)
BRB(1, 0, 6) = BRB(1, 0, 5)
BRB(2, 0, 6) = BRB(2, 0, 5)
BRB(3, 0, 6) = BRB(3, 0, 5)
End Sub
```

Sub LearnDefuzzify ()

```
Dim l As Integer

Larea = 0
Lweigharea = 0
For l = 0 To 3                  'loop over rules
        LRB(l, 13, 5) = 0
        LRB(l, 13, 3) = 0
        LRB(l, 13, 5) = LRB(l, 13, 4) * LRB(l, 13, 2)
        LRB(l, 13, 3) = LRB(l, 13, 1) * LRB(l, 13, 4) * LRB(l, 13, 2)
        Larea = Larea + LRB(l, 13, 5)
        Lweigharea = Lweigharea + LRB(l, 13, 3)
Next l
If Larea = 0 Then
        LearnAlt(MuteAlt) = 0           'Correlation factor of other budgets with activitty in question
Else LearnAlt(MuteAlt) = Lweigharea / Larea
End If
End Sub
```

Sub LearnFuzzify ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer

'this is supposed to fuzzify the Input value for each set

For i = 0 To 3                  'loop over number of rules
        For l = 0 To 5
                For j = (2 * l) To (2 * l + 1)
                        Select Case j
                        Case 2 * l
                                If LRB(i, j, 2) * (LearnInput(l) - LRB(i, j, 1)) > 10000 Then
                        LRB(i, j, 4) = 0
                    Else
                        LRB(i, j, 4) = 1 / (1 + (Exp((LRB(i, j, 2) * (LearnInput(l) - LRB(i, j,
                            1))))))
                    End If
                Case 2 * l + 1
                    If LRB(i, j, 2) * (LearnInput(l) - LRB(i, j, 1)) < -10000 Then
                        LRB(i, j, 4) = 1
                    Else
                        LRB(i, j, 4) = 1 / (1 + (Exp(-(LRB(i, j, 2) * (LearnInput(l) - LRB(i, j,
                        1))))))
                    End If
                End Select
            Next j
        Next l
'this is supposed to find the minimum of all active sets
        If LRB(i, 0, 5) = 1 Then                'checks whether first set is active or not
            LRB(i, 13, 4) = LRB(i, 0, 4)
        Else LRB(i, 13, 4) = 0
        End If
        For k = 1 To 12                         'loop over number of sets
            If LRB(i, k, 5) = 1 And LRB(i, (k - 1), 6) = 0 Then
                LRB(i, 13, 4) = LRB(i, k, 4)
            ElseIf LRB(i, k, 5) = 1 And LRB(i, (k - 1), 6) <> 0 And LRB(i, k, 4) < LRB(i, 13, 4)
            Then
                LRB(i, 13, 4) = LRB(i, k, 4)
            End If
            LRB(i, k, 6) = LRB(i, (k - 1), 6) + LRB(i, k, 5)
        Next k
Next i
End Sub
```

Inistore.bas

```
Option Explicit

Global actorname As String
Global actornum As Integer
```

Sub cmdefretrieve ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", defuzenter.Text1.Text
        actorname = t1("name")                  'name of actor which is edited
t1.Close
Set t2 = db.OpenTable(actorname & 0)
        t2.MoveFirst
            For i = 0 To 3
                defuzenter.defcen(i) = t2("cm_defuz_centre" & i)
                defuzenter.defsig(i) = t2("cm_defuz_sigma" & i)
            Next i
t2.Close
db.Close
End Sub
```

Sub cmdefstore ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer
Dim j As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
    t1.Index = "type_index"
    t1.Seek "=", defuzenter.Text1.Text
    actornum = t1("number")               'number of actors
    actorname = t1("name")                'name of actor which is edited
t1.Close
For i = 0 To actornum - 1
    Set t2 = db.OpenTable(actorname & i)
        t2.MoveFirst
        t2.Edit
        For j = 0 To 3
            t2("cm_defuz_centre" & j) = defuzenter.defcen(j)
            t2("cm_defuz_sigma" & j) = defuzenter.defsig(j)
        Next j
        t2.Update
    t2.Close
Next i
db.Close
End Sub
```

Sub CMfuzretrieve ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
    t1.Index = "type_index"
    t1.Seek "=", fuzzyenter.Text1.Text
    actorname = t1("name")                'name of actor which is edited
t1.Close
Set t2 = db.OpenTable(actorname & 0)
    t2.MoveFirst
    For i = 0 To 3
        fuzzyenter.setinput(2 * i + 16) = t2("cm_fuz_centre_pos" & 2 * i)
        fuzzyenter.setinput(2 * i + 8) = t2("cm_fuz_sigma" & 2 * i)
        fuzzyenter.setinput(2 * i + 17) = t2("cm_fuz_centre_pos" & 2 * i)
        fuzzyenter.setinput(2 * i + 9) = t2("cm_fuz_sigma" & 2 * i)
    Next i
t2.Close
End Sub
```

Sub cmfuzstore ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer
Dim j As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", fuzzyenter.Text1.Text
        actornum = t1("number")                 'number of actors
        actorname = t1("name")                  'name of actor which is edited
t1.Close
For i = 0 To actornum - 1
        Set t2 = db.OpenTable(actorname & i)
                t2.MoveFirst
                t2.Edit
                For j = 0 To 3
                        t2("cm_fuz_centre_pos" & 2 * j) = fuzzyenter.setinput(2 * j + 16)
                        t2("cm_fuz_sigma" & 2 * j) = fuzzyenter.setinput(2 * j + 8)
                Next j
                t2.Update
        t2.Close
Next i
db.Close
End Sub
```

Sub defuzretrieve ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", defuzenter.Text1.Text
        actorname = t1("name")                  'name of actor which is edited
t1.Close
Set t2 = db.OpenTable(actorname & 0)
        t2.MoveFirst
        For i = 0 To 3
                defuzenter.defcen(i) = t2("st_defuz_centre" & i)
                defuzenter.defsig(i) = t2("st_defuz_sigma" & i)
        Next i
t2.Close
db.Close
End Sub
```

Sub defuzstore ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer
Dim j As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", defuzenter.Text1.Text
        actornum = t1("number")                         'number of actors
        actorname = t1("name")                          'name of actor which is edited
t1.Close
For i = 0 To actornum - 1
        Set t2 = db.OpenTable(actorname & i)
                t2.MoveFirst
                t2.Edit
                For j = 0 To 3
                        t2("st_defuz_centre" & j) = defuzenter.defcen(j)
                        t2("st_defuz_sigma" & j) = defuzenter.defsig(j)
                Next j
                t2.Update
        t2.Close
Next i
db.Close
End Sub
```

Sub fuzzretrieve ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer

Set db = OpenDatabase("C:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
        t1.Index = "type_index"
        t1.Seek "=", fuzzyenter.Text1.Text
        actorname = t1("name")              'name of actor which is edited
t1.Close
Set t2 = db.OpenTable(actorname & 0)
        t2.MoveFirst
        For i = 0 To 3
                fuzzyenter.setinput(2 * i + 16) = t2("st_fuz_centre_pos" & 2 * i)
                fuzzyenter.setinput(2 * i + 8) = t2("st_fuz_sigma" & 2 * i)
                fuzzyenter.setinput(2 * i + 17) = t2("st_fuz_centre_pos" & 2 * i)
                fuzzyenter.setinput(2 * i + 9) = t2("st_fuz_sigma" & 2 * i)
        Next i
t2.Close
db.Close
End Sub
```

Sub fuzzstore ()

```
Dim db As database
Dim t1 As table
Dim t2 As table
Dim i As Integer
Dim j As Integer

Set db = OpenDatabase("c:\blah.mdb", True)
Set t1 = db.OpenTable("typedescriptor")
```

```
        t1.Index = "type_index"
        t1.Seek "=", fuzzyenter.Text1.Text
        actornum = t1("number")           'number of actors
        actorname = t1("name")            'name of actor which is edited
t1.Close
For i = 0 To actornum - 1
        Set t2 = db.OpenTable(actorname & i)
                t2.MoveFirst
                t2.Edit
                For j = 0 To 3
                        t2("st_fuz_centre_pos" & 2 * j) = fuzzyenter.setinput(2 * j + 16)
                        t2("st_fuz_sigma" & 2 * j) = fuzzyenter.setinput(2 * j + 8)
                Next j
                t2.Update
        t2.Close
Next i
db.Close
End Sub
```

Robots.bas

```
Option Explicit

Dim indicator As Integer
```

Sub Activity (Robot() As Actor)

```
'this one does determine the theoretical time spent on each activity each day

Dim total As Single
Static acttime(4) As Single
Static Payoff(4) As Single
Static decay(4) As Single
Dim i As Integer
Dim j As Integer
Dim k As Integer

total = 0
For i = 0 To 3
        total = total + importance(i)
Next i
For j = 0 To 3
        If total = 0 And (Robot(RobotNum).ShortTermIOCode(2 * j) <> 0 Or
                Robot(RobotNum).ShortTermIOCode(2 * j + 1) <> 0) Then
                importance(j) = .00001
                total = total + importance(j)
        Else
        End If
        acttime(j) = importance(j) / total * 24
        Robot(RobotNum).ShortTermActivities(j) = acttime(j)
Next j
For k = 0 To 3
        decay(k) = dec.Text1(k).Text
        Robot(RobotNum).ShortTermBudgets(k) = Robot(RobotNum).ShortTermBudgets(k) - decay(k)
        Robot(RobotNum).RemTime(k) = acttime(k)
Next k
Robot(RobotNum).DailyActivities = ""                      'reset activity string
End Sub
```

Sub GetSomething (Robot() As Actor)

```
'Acitvity 0 =recreation/sleep, 1=work, 2=shopping, 3=socialising
'CurrentActivity(0)=activity
'CurrentActivity(1)=arrival number in cell
'i) all activities get a payoff if capacity is not exeeded
'ii) Shopping and socialising are successful if i) is met AND there is somebody present to work
'iii) shopping and socialising cost money

Select Case Robot(RobotCount).CurrentActivity(0) = 0                    'recreation
Case True
    If Robot(RobotCount).CurrentActivity(1) <= ThisWorld.Capacity(0,
        Robot(RobotCount).CurrentCell) Then
        If Robot(RobotCount).PreviousActivity(0) = 0 Then
            Robot(RobotCount).ShortTermBudgets(0) = Robot(RobotCount).
            ShortTermBudgets(0) + (ThisWorld.Act(0, TimeOfDay, Robot(RobotCount).
            CurrentCell) * Exp(-((Robot(RobotCount).PreviousActivity(1) - 3.5) ^ 2) / (18)))

            Robot(RobotCount).ShortTermBudgets(1) =
            Robot(RobotCount).ShortTermBudgets(1)
        Else
            Robot(RobotCount).ShortTermBudgets(0) = Robot(RobotCount).
            ShortTermBudgets(0) + (ThisWorld.Act(0, TimeOfDay, Robot(RobotCount).
            CurrentCell) * Exp(-((0 - 3.5) ^ 2) / (18)))

            Robot(RobotCount).ShortTermBudgets(1) =
            Robot(RobotCount).ShortTermBudgets(1)

            If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
            Robot(RobotCount).CurrentActivity(2), 2) < 100 Then
                Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
                CMknow(Robot(RobotCount).CurrentActivity(0), Robot(RobotCount).
                CurrentActivity(2), 2) + 1
            Else
            End If
        End If
    Else
        If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
            Robot(RobotCount).CurrentActivity(2), 2) > -100 Then
            Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
            Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
            CMknow(Robot(RobotCount).CurrentActivity(0), Robot(RobotCount).
            CurrentActivity(2), 2) - 1
        Else
        End If
    End If
    If Robot(RobotCount).PreviousActivity(0) <> 0 Then
        Robot(RobotCount).PreviousActivity(0) = 0
        Robot(RobotCount).PreviousActivity(1) = 1
    Else
        Robot(RobotCount).PreviousActivity(0) = 0
        Robot(RobotCount).PreviousActivity(1) = Robot(RobotCount).PreviousActivity(1) + 1
    End If
    If ThisWorld.Capacity(0, Robot(RobotCount).CurrentCell) = 0 Then
        Demand(0, TimeOfDay, Robot(RobotCount).CurrentCell) = .5
    Else
```

```
                Demand(0, TimeOfDay, Robot(RobotCount).CurrentCell) = ((-CellCapacityCount(0,
                Robot(RobotCount).CurrentCell, 0)) + ThisWorld.Capacity(0, Robot(RobotCount).
        CurrentCell)) / ThisWorld.Capacity(0, Robot(RobotCount).CurrentCell)
        End If
End Select

Select Case Robot(RobotCount).CurrentActivity(0) = 1                                'work
Case True
        If Robot(RobotCount).CurrentActivity(1) <= ThisWorld.Capacity(1,
                Robot(RobotCount).CurrentCell) Then
                If Robot(RobotCount).PreviousActivity(0) = 1 Then
                        Robot(RobotCount).ShortTermBudgets(1) = Robot(RobotCount).
                        ShortTermBudgets(1) + (ThisWorld.Act(1, TimeOfDay,
                        Robot(RobotCount).CurrentCell))

                        Robot(RobotCount).ShortTermBudgets(1) =
                        Robot(RobotCount).ShortTermBudgets(1)
                Else
                        Robot(RobotCount).ShortTermBudgets(1) = Robot(RobotCount).
                        ShortTermBudgets(1) + (ThisWorld.Act(1, TimeOfDay,
                        Robot(RobotCount).CurrentCell))

                        Robot(RobotCount).ShortTermBudgets(1) =
                        Robot(RobotCount).ShortTermBudgets(1)
                        If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) < 100 Then
                                Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
                                CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) + 1
                        Else
                        End If
                End If
        Else
                If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) > -100 Then
                        Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
                        CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) - 1
                Else
                End If
        End If
        If Robot(RobotCount).PreviousActivity(0) <> 1 Then
                Robot(RobotCount).PreviousActivity(0) = 1
                Robot(RobotCount).PreviousActivity(1) = 1
        Else
                Robot(RobotCount).PreviousActivity(0) = 1
                Robot(RobotCount).PreviousActivity(1) = Robot(RobotCount).PreviousActivity(1) + 1
        End If
        If ThisWorld.Capacity(1, Robot(RobotCount).CurrentCell) = 0 Then
                Demand(1, TimeOfDay, Robot(RobotCount).CurrentCell) = .5
        Else
                Demand(1, TimeOfDay, Robot(RobotCount).CurrentCell) = ((-CellCapacityCount(1,
                Robot(RobotCount).CurrentCell, 0)) + ThisWorld.Capacity(1, Robot(RobotCount).
        CurrentCell)) / ThisWorld.Capacity(1, Robot(RobotCount).CurrentCell)
        End If
End Select

Select Case Robot(RobotCount).CurrentActivity(0) = 2                    'shopping
Case True
        If Robot(RobotCount).CurrentActivity(1) <= ThisWorld.Capacity(2, Robot(RobotCount).
        CurrentCell) And (CellCapacityCount(1, Robot(RobotCount).CurrentCell, 0) <>
        ThisWorld.Capacity(1, Robot(RobotCount).CurrentCell)) Then
                If Robot(RobotCount).PreviousActivity(0) = 2 Then
                        Robot(RobotCount).ShortTermBudgets(2) = Robot(RobotCount).
                        ShortTermBudgets(2) + (ThisWorld.Act(2, TimeOfDay,
                        Robot(RobotCount).CurrentCell))

                        Robot(RobotCount).ShortTermBudgets(1) = Robot(RobotCount).
                        ShortTermBudgets(1) - ThisWorld.Price(2, Robot(RobotCount).CurrentCell)
                Else
                        Robot(RobotCount).ShortTermBudgets(2) = Robot(RobotCount).
                        ShortTermBudgets(2) + (ThisWorld.Act(2, TimeOfDay,
                        Robot(RobotCount).CurrentCell))

                        Robot(RobotCount).ShortTermBudgets(1) = Robot(RobotCount).
                        ShortTermBudgets(1) - ThisWorld.Price(2, Robot(RobotCount).CurrentCell)
                        If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) < 100 Then
                                Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
                                CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) + 1
                        Else
                        End If
                End If
        Else
                If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) > -100 Then
                        Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
                        CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) - 1
                Else
                End If
        End If
        If Robot(RobotCount).PreviousActivity(0) <> 2 Then
                Robot(RobotCount).PreviousActivity(0) = 2
                Robot(RobotCount).PreviousActivity(1) = 1
        Else
                Robot(RobotCount).PreviousActivity(0) = 2
                Robot(RobotCount).PreviousActivity(1) = Robot(RobotCount).PreviousActivity(1) + 1
        End If

        If ThisWorld.Capacity(2, Robot(RobotCount).CurrentCell) = 0 Then
                Demand(2, TimeOfDay, Robot(RobotCount).CurrentCell) = .5
        Else
                Demand(2, TimeOfDay, Robot(RobotCount).CurrentCell) = ((-CellCapacityCount(2,
                Robot(RobotCount).CurrentCell, 0)) + ThisWorld.Capacity(2, Robot(RobotCount).
        CurrentCell)) / ThisWorld.Capacity(2, Robot(RobotCount).CurrentCell)
        End If
End Select

Select Case Robot(RobotCount).CurrentActivity(0) = 3                    'socialising
```

```
Case True
        If Robot(RobotCount).CurrentActivity(1) <= ThisWorld.Capacity(3, Robot(RobotCount).
        CurrentCell) And (CellCapacityCount(0, Robot(RobotCount).CurrentCell, 0) <>
        ThisWorld.Capacity(1, Robot(RobotCount).CurrentCell)) Then
                If Robot(RobotCount).PreviousActivity(0) = 3 Then
                        Robot(RobotCount).ShortTermBudgets(3) = Robot(RobotCount).
                        ShortTermBudgets(3) + (ThisWorld.Act(3, TimeOfDay, Robot(RobotCount).
                        CurrentCell) * Exp(-((Robot(RobotCount).PreviousActivity(1) - 3.5) ^ 2) / (18)))

                        Robot(RobotCount).ShortTermBudgets(1) = Robot(RobotCount).
                        ShortTermBudgets(1) - ThisWorld.Price(3, Robot(RobotCount).CurrentCell)
                Else
                        Robot(RobotCount).ShortTermBudgets(3) = Robot(RobotCount).
                        ShortTermBudgets(3) + (ThisWorld.Act(3, TimeOfDay, Robot(RobotCount).
                        CurrentCell) * Exp(-((0 - 3.5) ^ 2) / (18)))

                        Robot(RobotCount).ShortTermBudgets(1) = Robot(RobotCount).
                        ShortTermBudgets(1) - ThisWorld.Price(3, Robot(RobotCount).CurrentCell)
                        If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) < 100 Then
                                Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) = Robot(RobotCount).
                                CMknow(Robot(RobotCount).CurrentActivity(0),
                                Robot(RobotCount).CurrentActivity(2), 2) + 1
                        Else
                        End If
                End If
        Else
                If Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) > -100 Then
                        Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) =
                        Robot(RobotCount).CMknow(Robot(RobotCount).CurrentActivity(0),
                        Robot(RobotCount).CurrentActivity(2), 2) - 1
                Else
                End If
        End If

        If Robot(RobotCount).PreviousActivity(0) <> 3 Then
                Robot(RobotCount).PreviousActivity(0) = 3
                Robot(RobotCount).PreviousActivity(1) = 1
        Else
                Robot(RobotCount).PreviousActivity(0) = 3
                Robot(RobotCount).PreviousActivity(1) = Robot(RobotCount).PreviousActivity(1) + 1
        End If
        If ThisWorld.Capacity(3, Robot(RobotCount).CurrentCell) = 0 Then
                Demand(3, TimeOfDay, Robot(RobotCount).CurrentCell) = .5
        Else
                Demand(3, TimeOfDay, Robot(RobotCount).CurrentCell) = ((-CellCapacityCount(3,
                Robot(RobotCount).CurrentCell, 0)) + ThisWorld.Capacity(3, Robot(RobotCount).
        CurrentCell)) / ThisWorld.Capacity(3, Robot(RobotCount).CurrentCell)
        End If
End Select

If Robot(RobotCount).PreviousActivity(1) > 30000 Then
        Robot(RobotCount).PreviousActivity(1) = 30000
End If
End Sub
```

Sub MoveAvg (Robot() As Actor)

```
'calculates the moving average of the last 300 periods for the central position of the fuzzy input sets

Dim i As Integer
Dim j As Integer
Dim k As Integer
Static CentrePos(4) As Single
Static dist(4) As Single

For i = 0 To 3
        Robot(RobotCount).CentreMatrix(WriteNum, i) = Robot(RobotCount).ShortTermBudgets(i)
        dist(i) = Robot(RobotCount).ShortTermFuzCentre(2 * i + 1) -
        Robot(RobotCount).ShortTermFuzCentre(2 * i)
Next i
For j = 0 To 3
        For k = 0 To 299
                CentrePos(j) = CentrePos(j) + Robot(RobotCount).CentreMatrix(k, j)
        Next k
        CentrePos(j) = CentrePos(j) / 300
        Robot(RobotCount).ShortTermFuzCentre(2 * j) = CentrePos(j) - (dist(j) / 2)
        Robot(RobotCount).ShortTermFuzCentre(2 * j + 1) = CentrePos(j) + (dist(j) / 2)
        If Robot(RobotCount).ShortTermFuzCentre(2 * j) < 0 Then
                Robot(RobotCount).ShortTermFuzCentre(2 * j) = 0
                Robot(RobotCount).ShortTermFuzCentre(2 * j + 1) = dist(j)
        End If
Next j
End Sub
```

Sub Mutate (Robot() As Actor)

```
Select Case LearnAlt(MuteAlt)
Case Is >= 50
        If ((Robot(RobotNum).ShortTermIOCode(2 * Actinum) And (2 ^ (2 * MuteAlt)) = (2 ^ (2 *
                MuteAlt))) And (Robot(RobotNum).ShortTermIOCode(2 * Actinum + 1) And (2 ^ (2 *
                MuteAlt + 1)) = (2 ^ (2 * MuteAlt + 1)))) Then
                If Rnd > .5 Then
                        Robot(RobotNum).ShortTermIOCode(2 * Actinum + 1) = Robot(RobotNum).
                        ShortTermIOCode(2 * Actinum + 1) Xor (2 ^ (2 * MuteAlt + 1))
                Else
                        Robot(RobotNum).ShortTermIOCode(2 * Actinum) = Robot(RobotNum).
                        ShortTermIOCode(2 * Actinum) Xor (2 ^ (2 * MuteAlt))
                End If
                Robot(RobotNum).MutationTag = 1
        Else
                If Rnd > AndOrProb Then
                        Robot(RobotNum).ShortTermIOCode(2 * Actinum + 1) = Robot(RobotNum).
                        ShortTermIOCode(2 * Actinum + 1) Xor (2 ^ (2 * MuteAlt) + 2 ^ (2 *
                        MuteAlt + 1))
                Else
                        Robot(RobotNum).ShortTermIOCode(2 * Actinum) = Robot(RobotNum).
                        ShortTermIOCode(2 * Actinum) Xor (2 ^ (2 * MuteAlt) + 2 ^ (2 * MuteAlt + 1))
                End If
                Robot(RobotNum).MutationTag = 1
        End If
Case Is < 20
        If ((Robot(RobotNum).ShortTermIOCode(2 * Actinum) And (2 ^ (2 * MuteAlt)) = (2 ^ (2 *
                MuteAlt)))) Then
                Robot(RobotNum).ShortTermIOCode(2 * Actinum) = Robot(RobotNum).
                ShortTermIOCode(2 * Actinum) Xor (2 ^ (2 * MuteAlt))
        ElseIf (Robot(RobotNum).ShortTermIOCode(2 * Actinum + 1) And (2 ^ (2 * MuteAlt + 1)) =
                (2^ (2 * MuteAlt + 1))) Then
                Robot(RobotNum).ShortTermIOCode(2 * Actinum) = Robot(RobotNum).
                ShortTermIOCode(2 * Actinum) Xor (2 ^ (2 * MuteAlt))
        End If
        Robot(RobotNum).MutationTag = 1
End Select
End Sub
```

Sub ResetAgent (Tobor() As Actor)

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim n As Integer
Dim accessnum As Integer
Dim oldstrat As Integer

indicator = 0
Tobor(RobotNum).MutationTag = 0
oldstrat = Tobor(RobotNum).LearnS
For n = 0 To 3
        If Tobor(RobotNum).ShortTermBudgets(n) < Resetvalue Then
                indicator = indicator + 1
        End If
Next n
accessnum = Int(CMASize * Rnd)
If indicator <> 0 Then
        For k = 0 To 3                                  'resets agents's budgets
                Tobor(RobotNum).ShortTermBudgets(k) = 0
        Next k
        Select Case Repfil < Rnd
        Case True
                Select Case RUNFORM.Check1(7).Value
                Case 0
                        If (oldstrat = 0 And Rnd <= .33) Then
                                Tobor(RobotNum).LearnS = 1              'selects new learning strategy
                        ElseIf Rnd <= .33 Then
                                Tobor(RobotNum).LearnS = 0
                        End If
                        If ((oldstrat = 0 Or oldstrat = 1) And (Rnd > .33 And Rnd <= .67)) Then
                                Tobor(RobotNum).LearnS = 2
                        ElseIf (Rnd > .33 And Rnd <= .67) Then
                                Tobor(RobotNum).LearnS = 1
                        End If
                        If (oldstrat = 3 And Rnd > .67) Then
                                Tobor(RobotNum).LearnS = 2
                        ElseIf Rnd > .67 Then
                                Tobor(RobotNum).LearnS = 3
                        End If
                Case Else
                        Tobor(RobotNum).LearnS = Int(4 * Rnd)
                End Select
        End Select
        Select Case RUNFORM.Check1(2).Value
        Case 1
                If ItNum > 500 Then
                        For i = 0 To 7                  'generates new set of rules
                                Tobor(RobotNum).ShortTermIOCode(i) =
                                ThisWorld.CommonRules(accessnum, i)
                        Next i
                End If
        End Select

        Select Case RUNFORM.Check1(5).Value
        Case 1
                If Rnd < CopyProb Then
                        For i = 0 To 7
                                Tobor(RobotNum).ShortTermIOCode(i) =
                                ThisWorld.CommonRules(accessnum, i)
                        Next i
                End If
        End Select
        Tobor(RobotNum).Age = 0
        Tobor(RobotNum).MutationTag = 1
End If                                                  'end of reset

Select Case RUNFORM.Check1(5).Value

'copies the agents rule set to common knowledge if its age is greater than 500 time steps

Case 1
        If Tobor(RobotNum).Age > 500 Then
                For i = 0 To 7
                        ThisWorld.CommonRules(accessnum, i) =
                        Tobor(RobotNum).ShortTermIOCode(i)
                Next i
        End If
End Select
End Sub
```

Sub ResetFDV (Robot() As Actor)

```
Dim i As Integer
Dim j As Integer

For i = 0 To 3                          'reset final decision vector
        For j = 0 To 4
                Robot(RobotCount).FinalDecisionvector(j, i) = 0
        Next j
Next i
End Sub
```

Sub RndChngAgent (Tobor() As Actor)

```
Dim k As Integer
Dim n As Integer

indicator = 0
For n = 0 To 3                          'checks whether to generate new rule parameters
        If Tobor(RobotNum).ShortTermBudgets(n) < Resetvalue * Lthres Then
                indicator = indicator + 1
        End If
        If indicator <> 0 Then
```

```
                Tobor(RobotNum).ShortTermIOCode(n) = Int(65536 * Rnd)
                Tobor(RobotNum).MutationTag = 1
        ElseIf indicator = 0 Then
                Tobor(RobotNum).MutationTag = 0
        End If
Next n
End Sub
```

Sub RndChngAgent1 (Tobor() As Actor)

```
Dim k As Integer
Dim n As Integer

indicator = 0
For n = 0 To 3                          'checks whether to generate new rule parameters
        If Tobor(RobotNum).ShortTermBudgets(n) < Resetvalue * Lthres Then
                indicator = indicator + 1
        End If
        If indicator <> 0 Then
                For k = 2 * n To 2 * n + 1
                        Tobor(RobotNum).ShortTermIOCode(k) = Int(65536 * Rnd)
                Next k
                Tobor(RobotNum).MutationTag = 1
        ElseIf indicator = 0 Then
                Tobor(RobotNum).MutationTag = 0
        End If
        indicator = 0
Next n
End Sub
```

Sub SelAct (Robot() As Actor)

```
'This routine decides on the current activities to be taken by each
'individual on each time of day

Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer

'decision on Activity

Select Case ((Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 0) >=
Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 1)) And (Robot(RandArray(1, TTC)
Mod TotalCount).FinalDecisionvector(0, 0) >= Robot(RandArray(1, TTC) Mod TotalCount).
FinalDecisionvector(0, 2)) And (Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 0)
>= Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 3)))
Case True
        Robot(RandArray(1, TTC) Mod TotalCount).XPos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(2, 0)
        Robot(RandArray(1, TTC) Mod TotalCount).Ypos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(3, 0)
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentCell = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(4, 0)

        Robot(RandArray(1, TTC) Mod TotalCount).DailyActivities = Robot(RandArray(1, TTC) Mod
        TotalCount).DailyActivities & "A" & (Robot(RandArray(1, TTC) Mod TotalCount).XPos /
        Gridfactor) & (Robot(RandArray(1, TTC) Mod TotalCount).Ypos / Gridfactor)
        Robot(RandArray(1, TTC) Mod TotalCount).RemTime(0) = Robot(RandArray(1, TTC) Mod
        TotalCount).RemTime(0) - 1
        CellCapacityCount(0, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 0), 0)
        = CellCapacityCount(0, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 0),
        0) - 1
        CellCapacityCount(0, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 0), 1)
        = CellCapacityCount(0, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 0),
        1) + 1
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(0) = 0
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(1) = CellCapacityCount(0,
        Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 0), 1)
End Select

Select Case ((Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 1) >
Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 0)) And (Robot(RandArray(1, TTC)
Mod TotalCount).FinalDecisionvector(0, 1) >= Robot(RandArray(1, TTC) Mod TotalCount).
FinalDecisionvector(0, 2)) And (Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 1)
>= Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 3)))
Case True
        Robot(RandArray(1, TTC) Mod TotalCount).XPos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(2, 1)
        Robot(RandArray(1, TTC) Mod TotalCount).Ypos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(3, 1)
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentCell = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(4, 1)
        Robot(RandArray(1, TTC) Mod TotalCount).DailyActivities = Robot(RandArray(1, TTC) Mod
        TotalCount).DailyActivities & "B" & (Robot(RandArray(1, TTC) Mod TotalCount).XPos /
        Gridfactor) & (Robot(RandArray(1, TTC) Mod TotalCount).Ypos / Gridfactor)

        Robot(RandArray(1, TTC) Mod TotalCount).RemTime(1) = Robot(RandArray(1, TTC) Mod
        TotalCount).RemTime(1) - 1
        CellCapacityCount(1, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 1), 0)
        = CellCapacityCount(1, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 1),
        0) - 1
        CellCapacityCount(1, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 1), 1)
        = CellCapacityCount(1, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 1),
        1) + 1
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(0) = 1
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(1) = CellCapacityCount(1,
        Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 1), 1)
End Select

Select Case ((Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 2) >
Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 0)) And (Robot(RandArray(1, TTC)
Mod TotalCount).FinalDecisionvector(0, 2) > Robot(RandArray(1, TTC) Mod TotalCount).
FinalDecisionvector(0, 1)) And (Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 2)
>= Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 3)))
Case True
        Robot(RandArray(1, TTC) Mod TotalCount).XPos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(2, 2)
        Robot(RandArray(1, TTC) Mod TotalCount).Ypos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(3, 2)
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentCell = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(4, 2)

        Robot(RandArray(1, TTC) Mod TotalCount).DailyActivities = Robot(RandArray(1, TTC) Mod
        TotalCount).DailyActivities & "C" & (Robot(RandArray(1, TTC) Mod TotalCount).XPos /
        Gridfactor) & (Robot(RandArray(1, TTC) Mod TotalCount).Ypos / Gridfactor)
        Robot(RandArray(1, TTC) Mod TotalCount).RemTime(2) = Robot(RandArray(1, TTC) Mod
        TotalCount).RemTime(2) - 1
        CellCapacityCount(2, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 2), 0)
        = CellCapacityCount(2, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 2),
        0) - 1
        CellCapacityCount(2, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 2), 1)
        = CellCapacityCount(2, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 2),
        1) + 1
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(0) = 2
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(1) = CellCapacityCount(2,
        Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 2), 1)
End Select

Select Case ((Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 3) >
Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 0)) And (Robot(RandArray(1, TTC)
Mod TotalCount).FinalDecisionvector(0, 3) > Robot(RandArray(1, TTC) Mod TotalCount).
FinalDecisionvector(0, 1)) And (Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 3)
> Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(0, 2)))
Case True
        Robot(RandArray(1, TTC) Mod TotalCount).XPos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(2, 3)
        Robot(RandArray(1, TTC) Mod TotalCount).Ypos = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(3, 3)
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentCell = Robot(RandArray(1, TTC) Mod
        TotalCount).FinalDecisionvector(4, 3)

        Robot(RandArray(1, TTC) Mod TotalCount).DailyActivities = Robot(RandArray(1, TTC) Mod
        TotalCount).DailyActivities & "D" & (Robot(RandArray(1, TTC) Mod TotalCount).XPos /
        Gridfactor) & (Robot(RandArray(1, TTC) Mod TotalCount).Ypos / Gridfactor)
        Robot(RandArray(1, TTC) Mod TotalCount).RemTime(3) = Robot(RandArray(1, TTC) Mod
        TotalCount).RemTime(3) - 1
        CellCapacityCount(3, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 3), 0)
        = CellCapacityCount(3, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 3),
        0) - 1
        CellCapacityCount(3, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 3), 1)
        = CellCapacityCount(3, Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 3),
        1) + 1
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(0) = 3
        Robot(RandArray(1, TTC) Mod TotalCount).CurrentActivity(1) = CellCapacityCount(3,
        Robot(RandArray(1, TTC) Mod TotalCount).FinalDecisionvector(4, 3), 1)
End Select
End Sub
```

World.bas

```
Option Explicit

Global Const CellNos = 9
Type World
        XPos(CellNos) As Single
        Ypos(CellNos) As Single
        Act(3, 23, CellNos) As Single
        Capacity(3, CellNos) As Integer
        Price(3, CellNos) As Single
        tablenumber As Integer
        MaxPrice(3, CellNos) As Single
        MaxCapacity(3, CellNos) As Integer
        Turnover(CellNos) As Single
        CommonRules(19, 7) As Long
End Type
Global ThisWorld As World
```

Sub ChangeCap ()

```
Dim CapSigma As Single
Dim CapNorm As Single
Static MediumDem(4, CellNos) As Single
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim m As Integer

CapSigma = CaChange.Text2.Text
CapNorm = CaChange.Text1.Text
For j = 0 To 3
        For i = 0 To CellNos - 1
                MediumDem(j, i) = 0
        Next i
Next j
For j = 0 To 3
        For i = 0 To CellNos - 1
                For k = 0 To 299
                        MediumDem(j, i) = MediumDem(j, i) + LtAvDemand(j, i, k)
                Next k
                MediumDem(j, i) = MediumDem(j, i) / 300
        Next i
Next j
For l = 3 To 0 Step -1
        For m = 0 To CellNos - 1
                Select Case l
                Case 3
                        ThisWorld.Capacity(l, m) = Int(1 / (1 + Exp((-1) * CapSigma * (MediumDem(l,
                        m) - CapNorm))) * ThisWorld.MaxCapacity(l, m))
                Case 2
                        ThisWorld.Capacity(l, m) = Int(1 / (1 + Exp((-1) * CapSigma * (MediumDem(l,
                        m) - CapNorm))) * ThisWorld.MaxCapacity(l, m))
                Case 1
                        ThisWorld.Capacity(l, m) = Int(Capfac * Sqr(ThisWorld.Capacity(2, m))) +
                Int(Capfac * Sqr(ThisWorld.Capacity(3, m)))
                Case 0
                        ThisWorld.Capacity(l, m) = Int(1 / (1 + Exp((-1) * CapSigma * (MediumDem(l,
                        m) - CapNorm))) * ThisWorld.MaxCapacity(l, m))
                End Select
        Next m
Next l
End Sub
```

Sub ChangePrice ()

```
Dim PriceSigma As Single
Dim PriceNorm As Single
Dim i As Integer
Dim j As Integer
```

```
PriceSigma = PrChange.Text2.Text
PriceNorm = PrChange.Text1.Text
For i = 0 To 3                              'loop over activities
        For j = 0 To CellNos - 1                        'loop over all cells
                ThisWorld.Price(i, j) = 1 / (1 + Exp((-1) * PriceSigma * (AvDemand(i, j) - PriceNorm)))
                * ThisWorld.MaxPrice(i, j) * BasePrice(i, j)
        Next j
Next i
End Sub
```

Sub GetDemand ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Static sum(4, CellNos) As Single

For k = 0 To CellNos - 1
        For j = 0 To 3
                sum(j, k) = 0
        Next j
Next k
For k = 0 To CellNos - 1
        For j = 0 To 3
                For i = 0 To 23
                        sum(j, k) = sum(j, k) + Demand(j, i, k)
                Next i
                AvDemand(j, k) = sum(j, k) / 24
                LtAvDemand(j, k, WriteNum) = sum(j, k) / 24
        Next j
Next k
End Sub
```

Sub GetTurnover ()

```
Dim i As Integer
Dim j As Integer
Dim k As Integer

For i = 0 To CellNos - 1

'This one is adding up the daily turnover of shopping and socialising in all cells

        If CellCapacityCount(2, i, 1) < ThisWorld.Capacity(2, i) Then
                ThisWorld.Turnover(i) = ThisWorld.Turnover(i) + CellCapacityCount(2, i, 1) *
                ThisWorld.Price(2, i)
        Else
                ThisWorld.Turnover(i) = ThisWorld.Turnover(i) + ThisWorld.Capacity(2, i) *
                ThisWorld.Price(2, i)
        End If
        If CellCapacityCount(3, i, 1) < ThisWorld.Capacity(3, i) Then
                ThisWorld.Turnover(i) = ThisWorld.Turnover(i) + CellCapacityCount(3, i, 1) *
                ThisWorld.Price(3, i)
        Else
                ThisWorld.Turnover(i) = ThisWorld.Turnover(i) + ThisWorld.Capacity(3, i) *
                ThisWorld.Price(3, i)
        End If
Next i
For j = 0 To CellNos - 1
```

```
        For k = 0 To 23
                If ThisWorld.Act(1, k, j) <> 0 Then
                        If ThisWorld.Turnover(j) / 24 < 1 Then
                                ThisWorld.Act(1, k, j) = 1
                        Else
                                ThisWorld.Act(1, k, j) = ThisWorld.Turnover(j) / 24
                                                'change payoff for work on the basis of daily turnover
                        End If
                End If
        Next k
        ThisWorld.Turnover(j) = 0               'reset turnover vector
Next j
End Sub
```