Geographic Automata: From Paradigm to Software and Back to Paradigm

Itzhak Benenson[1], **Vlad Kharbash**[2]

[1]Department of Geography and Human Environment and
Environment Simulation Laboratory, Porter School of Environmental Studies
Tel Aviv University, Tel-Aviv, 69978, Israel
Tel: +972-3-6409178
FAX: +972-3-6406243
Email: bennya@post.tau.ac.il
URL: http://www.tau.ac.il/~bennya

[2]Environment Simulation Laboratory, Porter School of Environmental Studies,
Tel Aviv University, Tel-Aviv, 69978, Israel
Tel: +972-5-47522570
FAX: +972-3-6406243
Email: vlad@eslab.tau.ac.il
URL: http://eslab.tau.ac.il/Research

Abstract

Recently, an automata-based approach to portraying urban systems, based on the concept of Geographic Automata System (GAS) has been proposed. According to GAS paradigm, unitary urban objects are considered as automata of different types located in space, which are related to the other automata; ensembles of unitary automata are represented by relationships as well. The dynamics of GAS is defined by the transition rules that are applied to the automata and to the relationships. At the conceptual level, the majority, if not all, Cellular Automata and Multi-Agent urban models proposed until now can be reformulated in a GAS framework. The GAS-based view of urban systems has been recently implemented as an Object-Based Environment for Urban Simulation (OBEUS), which can be downloaded from http://eslab.tau.ac.il/OBEUS/OBEUS.htm. This paper presents the basics of the GAS paradigm and the user-friendly shareware version of the OBEUS. The goal of the GAS concept and of the OBEUS software is to become a common reference for high-resolution urban modeling.

1 Modern urban modeling requires common reference

Modern era of urban modeling began in 1960s with the intentionally simple and experimentally calibrated models of the land-use dynamics (Chapin and Weiss 1962; Lowry 1964; Chapin and Weiss 1968). Towards 1970s, the developments in the field bifurcated: some of the modelers proceeded with the pragmatic view of urban systems (Steinitz and Rogers 1970; Tobler 1970), while the others (Forrester 1969) followed paradigm of the complex system theory. “Pragmatic” models did not account for the model components and laws the developers could not verify. “Paradigmatic” models extended the view of physical and chemical systems (Prigogine 1967) to the urban ones, assumed that the city is governed by few “order” parameters (Haken 1983) and investigated the resulting low-dimensional system by means of limited number of specific scenarios. During 1970s, the paradigm of complex system theory took over, and urban models of 1970s and 80s were almost exclusively based on the view of the city as a set of regions represented by “stocks” of population groups, jobs, goods, and connected by “flows” of these materials. The impressive criticism of regional modeling (Lee 1973) as well as the early birds of the next twist of the spiral (Albin 1975; Tobler 1979) did not cause “collective” consequences.

It took urban modeling two decades to realize the basic shortage of the cybernetic view when applied to the cities – too many parameters and non-smooth dependences of the flows rates on the state of the system, make the models irreducible to low-dimensional ones; scenarios outcomes remain thus depending on a bulk of parameters, which could never be estimated. Following this disappointment, an interest to the urban modeling has been decreasing.

New twist began in 1990s, with Cellular Automata (CA) taking stage. Differently from regional models, the dimension of the urban CA is low – land units (cells) are characterized by few discrete states, and their state transition rules are simple. Being simple from geographic point of view, urban CA models are, however, far more complex than their mathematical counterparts, introduced in 1950s by John von Neumann and Stanislaw Ulam (von Neumann 1966). Indeed, urban CA have more than two states, their state transition rules are very specific, often irreversible, and, in addition, strongly depend on global parameters (Batty 1997; Clarke, Hoppen et al. 1997; Portugali 2000; White and Engelen 2000).

Recently, Multi-Agent System (MAS) model approach has taken the stage (Benenson, Omer et al. 2002; Gimblett 2002), just because humans are those who make decisions regarding the land use. Most recently, CA and MAS models were combined under the umbrella of Geographic Automata System (Benenson and Torrens 2004).

Compared to a regional modeling approach, CA, MAS or GAS are intuitive and easier for verification, but they have an evident disadvantage – namely, they do not propose any common formalism for model representation. Differential or difference equations - regional modeling tools - are understandable by all university graduates, and, at least theoretically, are sufficient for full representation of the model.

CA/MAS/GAS paradigm does not include specific form of model presentation, just because objects’ ‘automation rules’ are the synonym of an ‘algorithm’. The adequate representation of an algorithm is a computer program, and this enlightens the problem – if we depart from the very basic cases, it is very hard to understand the details of an algorithm by examining the code of a computer program. In addition, the cultural framework of the 2000s does not include any agreements regarding the style of the programming; say nothing about the verifications of the program outputs. The models, which include explicit presentation of the algorithm (Clarke, Hoppen et al. 1997) are thus ‘extraordinary’ and the fact that a model is published does not imply that other researcher can repeat the algorithms and obtain the same results (Berec 2002).

The ‘SimCity’ situation described above is intolerable and most of urban modelers share the view that something should come instead of a system of equations. We want to be able to vary the rules of cell state transitions and agents’ behavior on the one hand, while keep the model fully understandable and repeatable on the other. Two main views on how to maintain this balance can be recognized. The modeling environments originating from the CA tend to rigid parametric representation of the rules and provide the user’s interface for changing model parameters (Waddell 2001; Engelen, White et al. 2002). The environments originating from MAS (Tobias and Hofmann 2004) do not limit the user to the predefined parameterization, but arise anew the aforementioned problem of understanding model algorithm. Compromise environments, providing sufficient freedom in formulating model rules, while forcing researchers keep them understandable are necessary. In what follows, we build this compromise environment as an extension of relational database, based on recently introduced Geographic Automata Systems (GAS) (Benenson and Torrens 2004; Benenson, Aronovich et al. 2005).

2 Short introduction to Geographic Automata System (GAS)

Conceptually, GAS combines CA and MAS models within the object-based framework and directly represents real-world animate and inanimate objects by means of discrete, spatially located *Geographic Automata* (GA) (Benenson and Torrens 2004; Benenson, Aronovich et al. 2005; Torrens and Benenson 2005). GA represent spatially fixed and mobile entities, and Geographic Automata System (GAS) is nothing but a temporal database of urban automata. The recent GIS boom provides strong empirical support for a GAS approach - layers of urban GIS are nothing but collections of urban objects of the same kind.

GAS paradigm has there basic elements: automata objects, relationships between them, and the rules of automata behavior. Formally, it can be expressed as

$$G \sim \langle K, S, R, T \rangle \qquad (1)$$

$$T: (S_t, R_t) \rightarrow (S_{t+1}, R_{t+1}) \qquad (2)$$

where K denotes the types of GAS automata, S - automata states, R - relationships between automata (of the same or different types), and T - rules of objects and relationships changes.

The view of the world as relational database links between objects and relationships, and database tables, and between transition rules and insert/update/delete operators. The GAS model is thus updated following the changes of state and location the unitary GA undergo.

The operational core of the GAS concept lays in separation between fixed and non-fixed automata, and two ways of object location: direct, by means of coordinates, and indirect, by relating to other objects. It is natural to locate basic infrastructure elements (land parcels, buildings, roads, parks) directly, because in majority of the models they are fixed and do not change their location after being established. However, the other infrastructure element (apartments) and the non-fixed objects (householders, cars, pedestrians, property owners) are usually located indirectly, by means of 1:1, 1:N and M:N relationships, which relate between the apartment and the building, car and its position on the road link (linear referencing), owner and the property.

To make the idea of GAS operational one should interpret it as software, and this paper presents the latest progress in developing Object-Based Environment for Urban Simulation (OBEUS) that directly implements the GAS view. We assume below that the reader is familiar with the backgrounds of Relational DBMS and Entity-Relational data Model (ERM) (Howe 1983) and the Object-Oriented (OO) programming paradigm (Booch 1994).

3 From Geographic Automata System to a software

Implementing GAS view, we have to translate the components of Equations 1 and 2 into software objects and methods.

3.1 Automata of a given type $k \in K \rightarrow$ Instances of *population* class

Urban objects always deal with information found on levels above the individual level, and OBEUS population is a container for these meta-data.

3.2 Individual automata of type k $\rightarrow$ Class of *entities* of a type k

At an abstract level, OBEUS considers unitary automata, and distinguishes between fixed and non-fixed ones (Benenson and Torrens 2004). Fixed automata — buildings, parks, road links, traffic lights, etc. — do not change location after being established, while non-fixed entities may do that. Fixed objects are located directly, in a GIS manner, by means of a coordinate list. Non-fixed urban objects are located by pointing to one or several fixed ones (Benenson and Torrens 2004). As mentioned above, geo-referencing by pointing is actually a relationship. The priority of the relationships is the core of OBEUS.

3.3 Relationships between automata $\rightarrow$ Class of *relationships*

OBEUS considers relationships between entities explicitly, in a relational database sense - as software objects, which can have their own properties. To illustrate the use of relationships, the decision to sell or develop a lot might depend on the neighborhood's development potential (i.e. lot-lot relationships should be retrieved) and the relationship between the landowners and the lots (landowner-lot relationship).

To ensure consistency when applying transition rules T, we impose essential limitations on the semantics of the relationships in OBEUS. First, we do not permit direct relationships between non-fixed objects; second, we assume a leader-follower pattern of relationship between non-fixed and fixed objects (Noble 2000). According to the leader-follower pattern, non-fixed objects have exclusive update rights, while the related fixed objects can only query the relationship's attributes. The relationships between non-fixed objects are defined in OBEUS in a transitive way (see the list of methods generated by OBEUS below)

The distinction between objects and relationships offers immediate advantages in applications. Generally, it ensures that the algorithm of updating will result from *intentional* thought. Specifically, it provides an identical framework for abstract CA and explicit GIS-based land-use models, all of which are based on neighborhood relationships. The only difference between CA models and GIS-based irregular partitions of a plane is the variation in degree of neighborhood relationship from parcel to parcel (Flache and Hegselmann 2001; Benenson, Omer et al. 2002), the OBEUS table of neighborhood relationship is the same in both cases.

3.4 Transition rules T $\rightarrow$ Automata behavior rules

Geographic automata ‘behave’, and, in terms of OBEUS, their behavior results in insert/delete/update operations applied to objects and relationships. The behavioral rules are represented in OBEUS by class *methods*. We separate between assessment rules, aimed at estimating the state of automaton environment and automation rules that employ the changes of objects and relationships.

3.5 Collective phenomena in OBEUS

GAS and OBEUS frameworks do not contain any specific reference to *collective* phenomena. The ensembles of unitary automata (we call them *patterns*) are represented in OBEUS by means of the relationship table, which relates between pattern and unitary automata that compose it. Let us note that this is M:N relationship, just because the same unitary entity can belong to several patterns. The OBEUS patterns are evidently limited to ‘foreseeable’ ones, which can be recognized by transition rules T that are defined *a priory*. This structure is yet sufficient to recognize the typical examples of spatial self-organization, as areas populated predominantly by individuals of certain socio-economic groups (poor, rich, immigrants) (Portugali, Benenson et al. 1997), or deltatrons (Clarke 1997; Candau, Rasmussen et al. 2000); we are not aware of urban collective phenomena that cannot be presented by relationships.

4 Transition rules and the problem of time-management

The structure of the GAS, given by Equation 1, is implemented in OBEUS within the relational database framework. Let us turn now to the description of the GAS dynamics, which is given by Equation 2. Intuitively, application of the transition rules T makes objects “born,” “die,” or alter their properties, as well as force creation, destruction, or alteration the properties of objects’ relationships. The tools for objects management are provided by every OOP environment; tools for managing relationships are provided by OBEUS itself (see next section). To apply these tools we have to establish our view of time management in general. Namely, just as Equation 1 demands discrimination between objects in space, the implementation of Equation 2 demands discrimination between object-related events in time. The former is usually easy, while the latter is not easy at all.

Conceptually, the model view of the time of the objects, ensembles, and system as a whole, is crucial for understanding system dynamics. The famous patterns of the Game of Life are determined by the simultaneous (also synchronous or parallel) application of transition rules to all cells. Gliders, etc., do not appear when the *asynchronous* updating scheme is employed (Schonfisch and de Roos 1999). Generally, different arrangements of related events entail qualitatively different model outcomes. The continuous time and differential equations marginalize this problem and provide the basis for comparison between discrete techniques of computational solutions. In discrete systems, the time is a part of the system, and the way it is managed determines the model outcomes, while it is usually very hard if ever possible to estimate the consequences of a certain mode of time management without thorough investigation (Zeigler, Praehofer et al. 2000; Berec 2002; Worboys 2005)

In practice, the recognition of the model synchronization mode is critical for analyzing simulation program. The sequence of model events is defined by the sequence of the program operators, and two fully debugged programs can nonetheless produce different outcomes, just because two lines of code are reordered. In the same time, each version can be correct and represent different views of the time flow in the system.

Following interpretation of Equation 1 by means of populations and unitary objects, we define in OBEUS two time lines. The population time is referenced by *iterations*, while the time of objects by *ticks*. We limit OBEUS view of the time-flow to the simplest and “unavoidable” synchronous and asynchronous modes (which, however, seem sufficient for all urban models we are aware of):

Synchronous: In this mode, the transition rules are applied to all objects of a given population simultaneously. The calling order of the objects of a certain population has no influence on the model outcome.

Asynchronous: In this mode, objects change in turn, with each observing the urban reality as left by the previous object. In asynchronous mode, the modeler has to define the call order of the objects: currently they can be either random sequence or sequence in order of some characteristic.

The user of OBEUS has to choose one of above modes, and it will be employed when the objects' properties are updated. Transition rules are always employed a*synchronously*: no matter which mode is chosen, the information regarding the changes of relationships is immediately available to the other objects.

The OBEUS models are developed via *User Interface*

5 OBEUS User Interface

OBEUS is a .NET environment built with the C# programming language. It employs the .NET libraries of method for managing GIS layers and graphs, and its performance is sufficient for managing ~5*10^5 objects.

To define GAS model in OBEUS, one has to define

- Populations, Objects and Relationships
- Transition rules
- Model time flow
-

5.1 Model Tree – an interface for defining populations, objects, relations and transition rules

Through the Model Tree user defines populations of objects, objects, relationships, and the properties of these components (Figure 1).

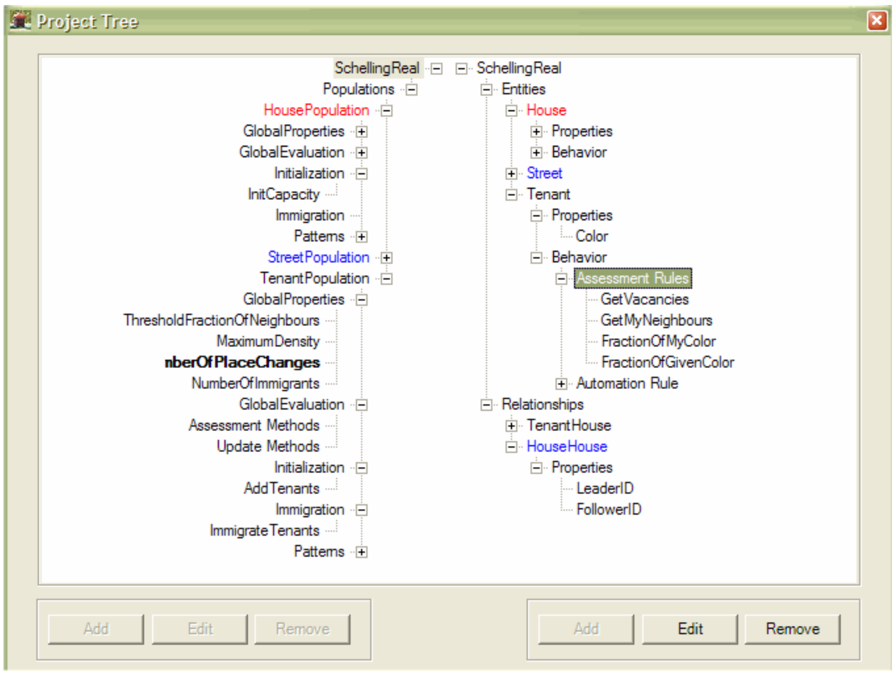


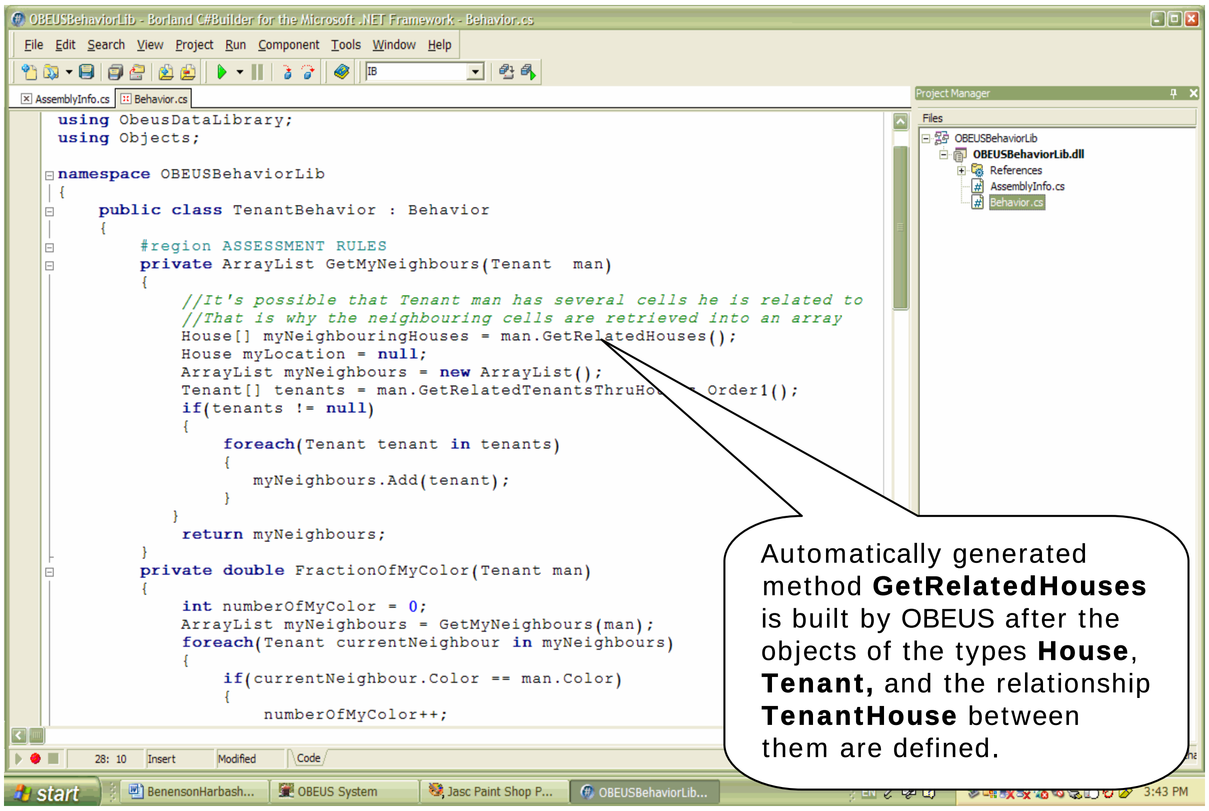
Figure 1: Model Tree for the generalized Schelling model. Properties marked: *bold* - presented by a chart; *red* – stored during the model run; *blue* – stored during the model run.

The objects in OBEUS are spatially located; fixed objects are located directly and presented to the user as GIS layers, non-fixed objects are located by relating to the fixed ones (i.e. by means of the relationship table). To mention some details of the OBEUS – GIS connection:

- Objects can be exported from the ArcInfo shapes, MapInfo tables, or constructed anew. In the latter case, objects' spatial arrangement is limited to regular grids of polygons and points, or random grid of points.
- In case the objects are exported from the GIS, the relationships between them, if ever, should be also exported. In case objects are defined anew within OBEUS, the relationships are built depending on distance or based on von Neumann or Moore neighborhood relationships in case of regular grid.

5.2 Transition rules – C# compiler enriched with automatically generated methods

Transition rules are the core of the model, and our design decision is avoiding any limitation in formulating these rules. To formulate them, OBEUS enables Borland C# compiler and the set of automatically generated methods, most of which regard management of relationships. Using Borland C# as an engine, the user has to define *assessment rules* (providing estimates of the state of object and its environment) and *behavioral rules* (those describing the transformations themselves) (Figure 2).



```
using ObeusDataLibrary;
using Objects;

namespace OBEUSBehaviorLib
{
    public class TenantBehavior : Behavior
    {
        #region ASSESSMENT RULES
        private ArrayList GetMyNeighbours(Tenant  man)
        {
            //It's possible that Tenant man has several cells he is related to
            //That is why the neighbouring cells are retrieved into an array
            House[] myNeighbouringHouses = man.GetRelatedHouses();
            House myLocation = null;
            ArrayList myNeighbours = new ArrayList();
            Tenant[] tenants = man.GetRelatedTenantsThruHo[illegible] Order1();
            if(tenants != null)
            {
                foreach(Tenant tenant in tenants)
                {
                   myNeighbours.Add(tenant);
                }
           }
            return myNeighbours;
        }
        private double FractionOfMyColor(Tenant man)
        {
            int numberOfMyColor = 0;
            ArrayList myNeighbours = GetMyNeighbours(man);
            foreach(Tenant currentNeighbour in myNeighbours)
            {
                if(currentNeighbour.Color == man.Color)
                {
                    numberOfMyColor++;
```

Automatically generated method **GetRelatedHouses** is built by OBEUS after the objects of the types **House**, **Tenant,** and the relationship **TenantHouse** between them are defined.

Figure 2: Behavioral component of the generalized Schelling model

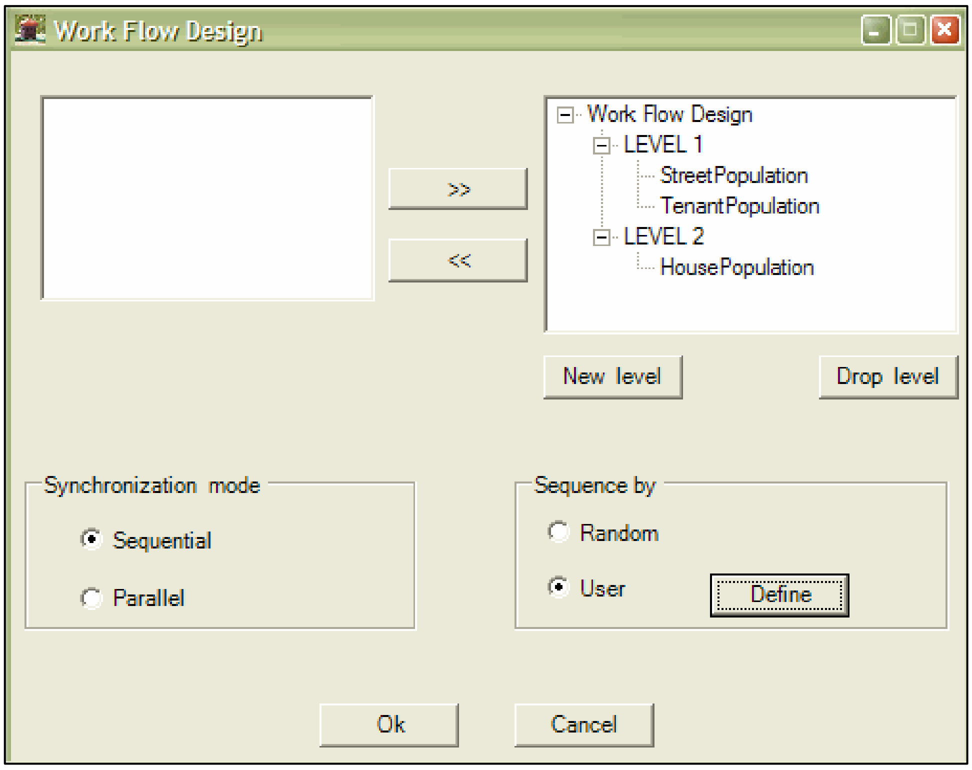
Implementation of GAS theory demands instantaneous access from the software objects representing model entities to those representing relationships, and *vice versa*. OBEUS resolves these demands by automatic construction of the access methods (that is a part of the T-rules given by Equation 1), with every new entity and/or relationship (that is S or R) defined. The idea can be illustrated with the Schelling model, which needs at least Tenant and House entities and TenantHouse relationship classes. The automatically constructed methods of the Tenant class are such as *Tenant.GetRelatedHouses()*, *Tenant.RemoveRelationship(House house)*, *Tenant.SetRelationship(House house)*, etc. These methods represent three of four groups of OBEUS methods, each corresponding to basic group of the database operators: **Get** methods (correspond to Select), **Add** methods (Create),

Remove methods (Delete), and **Set** methods (Update).

The methods are generated just after the automata S and relationships R are defined via the Model Tree, and immediately become available to model developer at right-click prompt just as the other methods of C# environment. The list of currently implemented automatically generated methods can be found in the OBEUS manual – http://eslab.tau.ac.il/OBEUS/OBEUSManual.pdf.

5.3 Model time flow

The dialog box for establishing synchronization mode is the second basic element of the OBEUS interface (Figure 3). It implements the aforementioned synchronous and asynchronous modes of objects' updating:



Synchronization mode: [✓ Sequential] Parallel

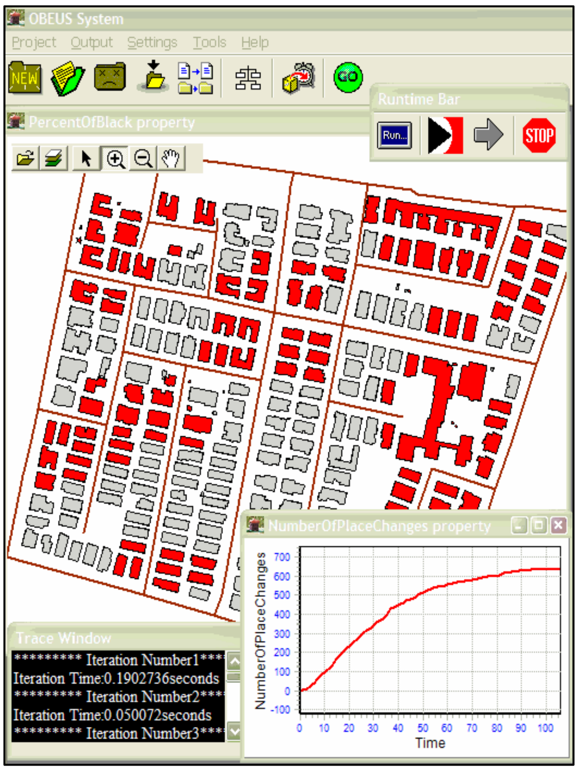
Sequence by: Random [✓ User]

Figure 3: Dialog box for establishing synchronization mode (an example of the generalized Schelling model)

5.4 OBEUS Output

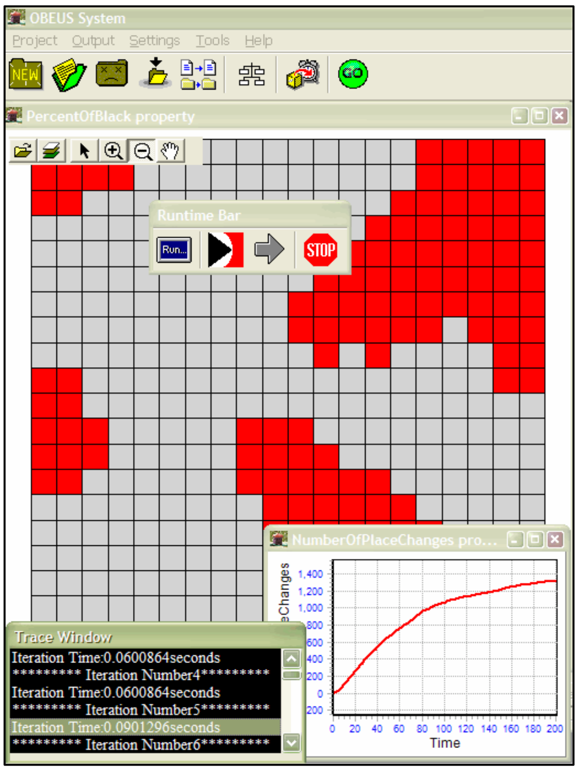
OBEUS output exploits its GIS/Database background. The attributes and features requested by the modeler are presented on the screen (Figure 4) and stored in the database for further processing.

To verify GAS concept and its implementation in OBEUS, we rebuild various urban high-resolution models published during last decades. In the lecture, we also compare OBEUS to RePast and NetLogo, acknowledged as most helpful in recent reviews (Tobias and Hofmann 2004).



```
********** Iteration Number1****
Iteration Time:0.1902736seconds
********** Iteration Number2****
Iteration Time:0.050072seconds
********** Iteration Number3****
```

a



```
Iteration Time:0.0600864seconds
********** Iteration Number4**********
Iteration Time:0.0600864seconds
********** Iteration Number5**********
Iteration Time:0.0901296seconds
********** Iteration Number6**********
```

b

Figure 4: Generalized Schelling model in (a) real-world and (b) abstract versions: standard windows of OBEUS output include maps, charts, and trace windows.

6 Back from the software to the paradigm

The concepts presented in this paper as well as the initial version of the OBEUS software were proposed in 2001 (Benenson 2001) and developed during 2002-2003 (Benenson and Torrens 2004; Benenson, Aronovich et al. 2005; Torrens and Benenson 2005). It took us another year to develop the user-friendly version of the software; the only component, which is not yet included, is the one dealing with patters, and we plan to complete it very soon. We still have to enhance many of the OBEUS components; however, our current focus is on the operational testing of OBEUS ability to implement *every* object-based model. To test this claim, we simply implement the models we know, beginning with those discussed in (Benenson and Torrens 2004), one by one. Our initial conclusions from these experiments are as follows:

- The GAS/OBEUS framework is handy and efficient for representing urban mobile and immobile objects and relationships, including cases when the latter are used for locating and for indicating aggregates.

- Each model we test demands a clear formulation of the details of behavioral rules; the use of a low-level programming language (C#) for representing the rules of automata behavior seems, thus, inevitable.

- The majority of the urban models we are aware of belong to one of very few classes: models of land-use dynamics, residential dynamics, car traffic, pedestrian traffic. Within the model class, the shareable set of objects, relationships and automatically generated methods can be specified *a priory*.

For example, car traffic models demand accounting for road lanes and the “adjacency” relationship between them, the, “OnLeft/OnRight” Boolean property of this relationship and automatic generation of methods that retrieve the car in front and the car behind of a given one. In terms of GAS, for a given class of models, each of S, R, and T becomes a pair - (S_M, S_U), (R_M, R_U) and (T_M, T_U), where the index M denotes the shareable and the index U the user-defined part of a component. Specifying model classes and developing their shareable components will be our next step in OBEUS design.

- The constraints of the OBEUS time-flow do not limit the developer when *weak reactive agents* (Wooldridge and Jennings 1995) are employed. The use of *strong agents* (Maes 1995b; Maes 1995a; Wooldridge and Jennings 1995), who negotiate and make coalitions, often demands to “break the time-flow,” that is to interrupt looping over the objects and to change the update order depending on the results of objects’ behavior. In this case, the OBEUS time-flow representation is insufficient.

The latter observation entails a series of basic questions, all regarding “human properties” of the GAS automata: do we really need strong agency in urban models? What are the characteristics of strong agents’ behavior? How should we specify these characteristics in order to be able to operationally compare between the “strength of agents” in two models? We do not have answers to these questions and hope to discuss them at GeoComputation 2005.

7 Concluding note

OBEUS software: **http://eslab.tau.ac.il/OBEUS/OBEUS.htm**
OBEUS User’s Manual: **http://eslab.tau.ac.il/OBEUS/OBEUSManual.pdf**

8 References

Albin, P. S. (1975). The Analysis of Complex Socioeconomic Systems. Lexington, MS, Lexington Books.

Batty, M. (1997). "Cellular automata and urban form: A primer." Journal of the American Planning Association **63**(2): 266-274.

Benenson, I. (2001). OBEUS: Object-Based Environment for Urban Simulation. 6th International Conference on GeoComputation, University of Queensland, Brisbane, Australia, GeoComputation CD-ROM, available also at http://www.geocomputation.org/2001/papers/benenson.pdf.

Benenson, I., S. Aronovich and S. Noam (2005). "Let's Talk Objects: Generic Methodology for Urban High-Resolution Simulation." Computers, Environment and Urban Systems **29**: 425–453.

Benenson, I., I. Omer and E. Hatna (2002). "Entity-based modeling of urban residential dynamics - the case of Yaffo, Tel-Aviv." Environment and Planning B. **29**: 491-512.

Benenson, I. and P. M. Torrens (2004). Geosimulation: automata-based modeling of urban phenomena. London, Wiley.

Berec, L. (2002). "Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis." Ecological Modelling **150**: 55-81.

Booch, G. (1994). Object-oriented analysis and design with applications. Menlo Park, Ca, Addison-Wesley.

Candau, J. T., S. Rasmussen and K. C. Clarke (2000). A coupled cellular automata model for land use/land cover change dynamics. 4th International Conference on Integrating GIS and Environmental Modeling (GIS/EM4): Problems, Prospects and Research Needs, Banff, Alberta, Canada.

Chapin, F. S. and S. F. Weiss (1962). Factors influencing land development. An urban studies research monograph. Chapel Hill, Institute for Research in Social Science, University of North Carolina**:** 68.

Chapin, F. S. and S. F. Weiss (1968). "A Probabilistic Model for Residential Growth." Transportation Research **2**: 375-90.

Clarke, K. C. (1997). Land Transition Modeling With Deltatrons, http://www.geog.ucsb.edu/~kclarke/Papers/deltatron.html. **2002**.

Clarke, K. C., S. Hoppen and L. J. Gaydos (1997). "A self-modifying cellular automata model of historical urbanization in the San Francisco Bay area." Environment and Planning B: Planning and Design **24**(2): 247-261.

Engelen, G., R. White and I. Uljee (2002). The MURBANDY and MOLAND models for Dublin. Submitted to European Comisssion Joint Research Center, Ispra, Italy. Dublin, ERA-Maptec**:** 172.

Flache, A. and R. Hegselmann (2001). "Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics." Journal of Artificial Societies and Social Simulation vol. 4, no. 4 **4**(4): <http://www.soc.surrey.ac.uk/JASSS/4/4/6.html>.

Forrester, J. W. (1969). Urban Dynamics. Cambridge, MA, The MIT Press.

Gimblett, H. R. (2002). Integrating geographic information systems and agent-based technologies for modeling and simulating social and ecological phenomena. Integrating Geographic Information Systems and Agent-Based Modeling Techniques for Simulating Social and Ecological Processes. H. R. Gimblett. Oxford, Oxford University Press**:** 1-21.

Haken, H. (1983). Synergetics. An Introduction. Berlin, Springer.

Howe, D. R. (1983). Data analysis for data base design. London, Edward Arnold.

Lee, D. B. (1973). "A Requiem for Large Scale Modeling." Journal Of The American Institute Of Planners **39**(3): 163-178.

Lowry, I. S. (1964). A model of metropolis. Santa Monica, California, The RAND Corporation**:** 136 p.

Maes, P. (1995a). "Artificial Life Meets Entertainment: Life like Autonomous Agents." Communications of the ACM **38**(11): 108-114.

Maes, P. (1995b). Modeling Adaptive Autonomous Agents. Artificial Life, An Overview. C. G. Langton. Oxford, MIT Press**:** 135-162.

Noble, J. (2000). Chapter 6. Basic Relationship Patterns. Pattern Languages of Program Design 4. N. Harrison, B. Foote and H. Rohnert, Addison-Wesley**:** 73-89.

Portugali, J. (2000). Self-Organization and the City. Berlin, Springer.

Portugali, J., I. Benenson and I. Omer (1997). "Spatial cognitive dissonance and sociospatial emergence in a self-organizing city." Environment and Planning B-Planning & Design **24**(2): 263-285.

Prigogine, I. (1967). Introduction to thermodynamics of irreversible processes. NY, Interscience.

Schonfisch, B. and A. M. de Roos (1999). "Synchronous And Asynchronous Updating In Cellular Automata." Biosystems **51**: 123-143.

Steinitz, C. and P. Rogers (1970). A System Analysis Model of Urbanization and Change: An Experiment in Interdisciplinary Education. Cambridge, Massachusetts, MIT Press.

Tobias, R. and C. Hofmann (2004). "Evaluation of free Java-libraries for social-scientific agent based simulation." Journal of Artificial Societies and Social Simulation **7**(1): <http://jasss.soc.surrey.ac.uk/7/1/6.html>.

Tobler, W. (1970). "A computer movie simulating urban growth in the Detroit region." Economic Geography **46** (Issue Supplement: Proceedings. International Geographical

Union. Commission on Quantitative Methods (Jun.,1970)): 234-240.

Tobler, W. (1979). Cellular Geography. Philosophy in Geography. S. Gale and G. Ollson. Dordrecht, Kluwer**:** 379-386.

Torrens, P. M. and I. Benenson (2005). "Geographic Automata Systems." International Journal of Geographic Information Science **19**(4): 385-412.

von Neumann, J. (1966). Theory and organization of complicated automata (Part One) . Theory of Self-Reproducing Automata [by] John von Neumann. A. W. Burks. Urbana, University of Illinois Press**:** 29-87.

Waddell, P. A. (2001). Between politics and planning: UrbanSim as a decision-support system for metropolitan planning. Planning Support Systems: Integrating Geographic Information Systems, Models, and Visualization Tools. R. K. Brail and R. E. Klosterman. Redlands, CA, ESRI Press**:** 201-228.

White, R. and G. Engelen (2000). "High-resolution integrated modelling of the spatial dynamics of urban and regional systems." Computers, Environment and Urban Systems **24**(5): 383-400.

Wooldridge, M. J. and N. R. Jennings (1995). "Intelligent agents: theory and practice." Knowledge Engineering Review **10**(2): 115-152.

Worboys, M. (2005). "Event-oriented approaches to geographic phenomena." International Journal of Geographical Information Science **19**(1): 1-28.

Zeigler, B. P., H. Praehofer and T. G. Kim (2000). Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. New York, Academic Press.