Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers, Environment and Urban Systems
29 (2005) 425–453
Computers, Environment and Urban Systems
www.elsevier.com/locate/compenvurbsys

Let's talk objects: generic methodology for urban high-resolution simulation

Itzhak Benenson [a,b,*], Shai Aronovich [a], Saar Noam [a]

[a] *Revson Environment Simulation Laboratory of the Porter School of Environmental Studies, University Tel Aviv, Tel-Aviv, Ramat-Aviv 69978, Israel*
[b] *Department of Geography and Human Environment, University Tel Aviv, Tel-Aviv,Ramat-Aviv 69978, Israel*

Received 19 June 2003; received in revised form 25 November 2003; accepted 25 November 2003

"Much of the behavior of systems rests on relationships and interactions", Jay W. Forrester (1969). Urban Dynamics. Cambridge, MIT Press, p. 114.

Abstract

An object-based approach to portraying urban systems, based on the priority of spatial relationships, is proposed. Real-world entities are represented by means of unitary fixed and non-fixed objects. Fixed unitary objects are located directly, by coordinates, while non-fixed unitary objects are located by pointing to fixed objects. Self-organizing spatial ensembles of unitary urban objects are represented by means of emerging domain objects. Software objects represent unitary objects and domains as well as the relationships between them. The user is responsible for formulating rules of object creation, updating, and destruction, while the system automatically updates relationships following movements of non-fixed objects, domain self-organization, change, and destruction. A prototype software system for simulating housing dynamics is presented.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Urban modeling; Modeling software; GIS; Cellular automata; Multi-agent systems; Object-oriented programming; Entity-relationship data model

[*] Corresponding author. Present address: Revson Environment Simulation Laboratory of the Porter School of Environmental Studies, University Tel Aviv, Tel-Aviv, Ramat-Aviv 69978, Israel.
E-mail address: bennya@post.tau.ac.il (I. Benenson).

0198-9715/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compenvurbsys.2003.11.008

1. Urban modeling: from regions to cells and agents and further to objects

During the first two decades of their development (1960s–1980s), urban modeling and simulation were almost exclusively based on flat partitions of urban space into regions that served as aggregates for lands, population, jobs, and transportation (Allen & Sanglier, 1979). Regional models concentrated on the redistribution of population and resources; their dynamics were represented by non-linear differential or difference equations formulated in terms of region state variables—population size, jobs, etc. Although providing a basic outline of urban dynamics, reconsideration of these models raises fundamental reservations regarding the adequacy of the aggregate view of the city (Lee, 1973, 1994). Recent intensive development of Cellular Automata (CA) models of urban land use dynamics can be considered a reaction to this dissatisfaction (Torrens, 2000). The idea behind CA is to approach high-resolution partitioning of urban space as a regular grid of internally homogeneous cells, each uniquely characterized by its states, where the future state of a cell depends on its current state and the states of its neighbors (Phipps, 1989; Tobler, 1979). The analogy between cells and states on the one hand and land parcels and land uses on the other provides a salient incentive for CA land-use models (White & Engelen, 1997).

To overcome the weakness of aggregation, regions in regional models are characterized by as many properties as possible. They consequently demand weighty data support and lengthy periods of calibration (Benenson, 1999). In contrast, CA models employ few cell states only, base on simple local relationships, and demand a handful number of parameters for plausible imitations of urban reality. Recent trends strongly favor CA modeling for theoretical and applied simulations alike (Batty, 1997; Torrens & O'Sullivan, 2001; White & Engelen, 2000), accompanied by new approaches to parameter estimation and calibration (Clarke, Hoppen, & Gaydos, 1997).

Urban CA models have one evident and principal limitation: the *immobility of cells*. Housing and transport dynamics are ready examples of the necessity for *mobile* or, more generally, *non-fixed* entities in urban models. Non-fixed units are obliged to make decisions regarding their location (in the broadest sense) and relocation. In an urban milieu, all these entities—landowners, householders, firms, pedestrians and cars make location and relocation decisions autonomously or semi-autonomously. Hence, they are all considered as *agents* and their dynamics are consequently studied with Multi-Agent Systems (MAS) (Ferber, 1999). Geographic MAS agents are automata just as are CA cells; the specificity of *geographic* agents is their location. To simulate a geographic agent, its state should include location information whereas automation rules should include a description of the agent's relocation.

Urban systems contain fixed and non-fixed entities. The fusion of CA and MAS approaches is therefore a natural further step in urban high-resolution modeling. Recent urban models of this kind have usually been based on cellular automata, populated by agents migrating between cells (Portugali, 2000; Portugali, Benenson, & Omer, 1994, 1997). This view is evidently constrained: their basis, a regular grid of cells, does not allow for direct representation of infrastructure entities and agents

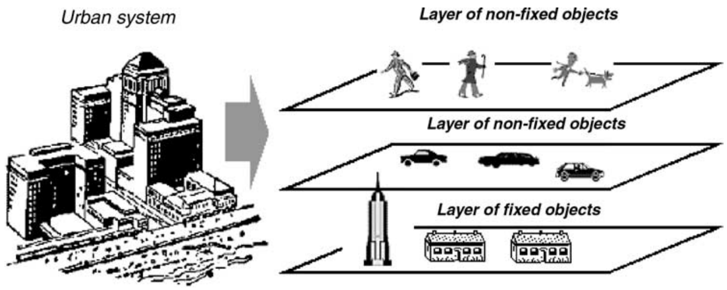


Fig. 1. Urban system as a set of layers of objects.

that are larger than a single cell and demand aggregation of those infrastructure entities and agents that are smaller in size. Regular partition of space engenders many other fundamental problems regarding representation of the network elements, the definition of spatial relations, the relationship between data on spatial entities of different size and so forth.

Regular partition of space, as a conceptual basis should therefore be substituted by another approach. In this paper, we propose such an alternative: the formal representation of urban systems by means of discrete *objects* that directly represent real-world urban entities. Objects can be of different natures, spatial extension and hierarchical rank; furthermore, they can represent non-fixed in addition to fixed entities and ensembles of entities. Spatial relationships between objects would thereby determine the structure of urban space. Although the object approach to the representation of urban reality has been mentioned in the literature (Erickson & Lloyd-Jones, 1997; Galton, 2001; Semboloni, 2000), the idea has remained inchoate to date.

The recent GIS boom provides strong empirical support for an object-based approach: The layers of urban GIS are nothing but collections of urban objects of the same kind (Fig. 1). Spatial relationships between objects can be estimated within GIS on the basis of adjacency, overlay, visibility and accessibility, among other variables. Geographic databases provide the foundations for the object-based approach; to take the next step and model the city we have to 'animate' the geographic features, that is, formulate the rules by which urban objects are created, relocated, changed, and destroyed.

We consider the object approach as fundamental for *every* software system intended for high-resolution simulation of urban dynamics. What follows is an attempt to establish an object-based paradigm for formalizing urban reality on the one hand, and implementing it as a software environment for urban simulations on the other. [1]

[1] The paper by Itzhak Benenson and Paul Torrens "Geographic automata systems and urban simulation," submitted to the *International Journal of Geographic Information Systems*, June 2003, proposes automata-based formalization of this paradigm.

The simulation is a field in which each concept should be 'down to earth', that is, enable for implementation at a level appropriate for non-expert users who wish to simulate specific systems for which they lack sufficient proficiency. What might be the specificity of the software required for such an object-based paradigm? We claim that this specificity rests on *object location and spatial relationships*. The priority of this information represents the core of the Object-Based Environment for Urban Simulation (OBEUS), which we develop in this milieu.

We consider the object approach as fundamental for *every* software system aimed at high-resolution simulation of urban dynamics. To illustrate OBEUS implementation, we employ one model of urban housing dynamics (Benenson, Omer, & Hatna, 2002). Other models have been accepted for publication (Portugali, 2004) or are in preparation.

In what follows, we use the concept 'urban object' when talking about the real-world city. 'Software object' is used when referring to the software environment. We assume below that the reader is familiar with the backgrounds of Relational DBMS and Entity-Relational data model (Howe, 1983) and the Object-Oriented programming paradigm (Booch, 1994).

2. Object-based representation of urban dynamics—the idea

We consider urban infrastructure and social objects as *spatially located* discrete entities, characterized by several properties, which can be directly interpreted as software objects. Computer representations of real-world urban entities comprise only one of the three basic components necessary for formalization of urban processes; the second represents self-organizing ensembles of entities and the third relationships between entities. All three components can be interpreted as software objects.

2.1. Unitary objects

Urban objects can be *fixed* or *non-fixed*. Fixed objects—buildings, parks, road links, traffic lights, etc.—do not change location after being established, while non-fixed objects may do that. In the majority of situations, non-fixed objects—householders, firms, pedestrians, vehicles—can be called *mobile* (whereas fixed objects can be called *immobile*) although we prefer a more general notion to include objects such as landlords or service companies, entities having several possessions or departments located at different places.

2.2. Relationships

To portray an urban system, we have to know *relationships* between urban objects, which define urban dynamics in addition to the objects themselves. To illustrate, the decision to sell or develop a lot depends on (a) the properties of the landowner and the land estate in addition to (b) the neighborhood's development potential. The latter demands information about which lots are proximate to the respective lot, and the nature of the influence of each neighbor on the lot considered. A decision to open a new boutique, for example, depends on (a) an entrepreneur's ability and the rent at the potential site as well as (b) the location's attractiveness for potential customers, a factor that depends in turn on accessibility (e.g. the time it takes to travel to the site) and other factors that impinge on the properties of the location–customer relationship. The above examples emphasize specificity of the urban system. Like any other geographic systems they necessarily rest on the *spatial relations* maintained between objects.

Relationships, whether spatial or not, have their own properties; the total dominance of the relational database approach, which follows the Entity-Relationship Model (ERM), is the best proof of the efficiency benefits achieved by the distinction between objects and relationships and the view of relationships as self-existing software objects (Howe, 1983).

2.3. Domains

The contemporary view of the city as a complex system assumes that ensembles of objects that partly inherit properties from their unitary components while partly displaying properties of their own can self-organize over time (Portugali, 2000). From the perspective of the object-based approach, we also consider self-organizing spatial ensembles of unitary urban objects as objects and refer to them as *domains*. Typically, domains represent neighborhoods populated by individuals belonging to distinct socioeconomic groups or characterized by specific land uses. The million-dollar question asked by complex system theory with respect to self-organization is to what extent the properties of ensembles are determined by the properties of the assembling objects. Whatever the answer, the simulation framework should provide tools for recognizing ensemble emergence and for relating between them and unitary objects.

3. From an object-based paradigm to simulation software

As distinct from physics or chemistry, urban as well as ecological or other high-level systems have not given rise to universally corroborated laws. Instead, geographic theory proposes *concepts*, such as the concept of regional modeling or the concepts of cellular automata and multi-agent systems that we sketched above. To apply the concept to multi-faceted urban reality, the researcher can utilize some of the existing formalizations of the phenomena or construct her own.

Simulation software then has to clearly determine its own position between the extremes of abstract concepts and rigid predetermined formalizations. Ideally, an environment for urban simulation should be as 'open' as concepts of urban dynamics are. However, the greater the freedom endowed the user, the less we benefit from the specialized simulation software. General-purpose programming languages implemented within user-friendly development environments—such as Visual Basic, Java, Delphi, C++ or C#—are standard features of modern education, always at hand. The environments based on predetermined formalization of the laws of system dynamics promise significant reduction of development time but demand significant time for learning; they are also bound to specific kinds of data. Their use is always accompanied by the fear of discovering that something 'does not fit,' too late in the process.

The simulation practitioners are aware of this trouble; hence, the recent trend is towards environments that support specific concepts of the human or environmental system in question. For example, software environments for agent-based simulations provide frameworks for formalizing agent systems that are enriched with standard tools for the presentation and analysis of simulation results. Two fundamental features of this environment are: (1) its tendency to be as open as possible for different formalizations of agents' behavior provided that the user accepts an agent-based view of the system, and (2) the user's full responsibility for the formalization's correctness. Regarding urban systems, the most popular agent-based simulation environments are SWARM and SWARM-based systems. Recent developments, either superstructures based on SWARM—MAML (Gulyas, Kozsik, & Corliss, 1999), EVO (Krumpus, 2001)—or constructed from scratch—such as RePast (RePast, 2003)—emphasize this approach as they develop additional frameworks for formulating rules of agent behavior.

Our view is obviously close to this concept-oriented software. Our aim is to construct an environment that directly reflects the above object-based approach to the representation of urban reality. We argue that urban geographers share a common perspective; if we limit ourselves to *urban systems* and *urban processes*, we will be able to specify the universal object-oriented paradigm in a way that can significantly assist the "object-talking" student in formalizing and studying the dynamics of specific urban phenomena.

3.1. Software for high-resolution modeling of urban dynamics

Several approaches to high-resolution representation of urban systems have recently been proposed. They either focus on CA exclusively (Besussi, Cecchini, & Rinaldi, 1998; Couclelis, 1997; Wagner, 1997), or consider spatial relationships as a marginal issue (Ginot, Le Page, & Souissi, 2002). Some of these approaches approach the OB paradigm we propose. 'Free Agents in City Space' (Portugali, 2000; Portugali et al., 1997) and the FEARLUS software system (Polhill, Gotts, & Law, 2001) differentiate between mobile and immobile objects but because both are based on cellular space, they defer the idea of objects as representations of real-world entities and do not consider relationships explicitly. Erickson and Lloyd-Jones (1997) as well as Semboloni (2000) consider non-uniform partitions of urban space but account for infrastructure units only. Batty, Xie, and Sun (1999) use ensembles limited to neighborhoods of different size whereas others (Benenson, 1999; Portugali et al., 1997) restrict themselves to areas populated by agents possessing similar properties. As far as we could ascertain the idea of direct software interpretation of real-world objects, ensembles, and the relationships between them is absent from the journal literature (Ginot et al., 2002; Gruer, Hilaire, Koukam, & Cetnarowicz, 2002; Lorek & Sonnenschein, 1999; Noth, Borning, & Waddell, 2003) and Internet sources (Anonymous, 2002).

3.2. The problems to be addressed

A dynamic system aimed at integrating the ideas behind GIS, CA, MAS and self-organization within an OOP environment raises a number of technical and conceptual problems. The conceptual problems relate to the management of the two components—relationships and ensembles—that manifest system complexity. Let us mention just two basic issues:

Recognition and management of ensembles. The behavior of urban objects depends on the location, form, and state of domains; the system should capture the emergence and evolution of the latter. Recognition and updating of domains should accompany any change in the location and/or state of unitary objects, a task that might be computationally complex.

Synchronization of events. Two householders, ignorant of each other, try to occupy two apartments in the same house. Who will finally settle there? The management of events that occur simultaneously yet influence one another induces synchronization problems of the sort intensively discussed in CA, MAS, temporal GIS and other research dealing with discrete-time systems (Berec, 2002).

Within the framework of the object-based approach, the synchronization problem is directly translated into the problem of updating relationships that change in time and in space. The results, especially for many-to-many (M:N) relationships, depend on the updating sequence. For example, for landlords who can hold several possessions and do that in common (i.e., be in M:N relationship with them), the decision of one landlord to sell her share of a certain possession can influence the behavior of the others; whose decision has precedence, and how does the decision of one influence the behavior of the others? The urban modeler usually lacks any clear idea regarding these details, information that can initiate conflicts and introduce inconsistencies in the implementation of the model rules. The problem is not exclusive to M:N relationships. Householders who migrate between urban habitats provide a similar example. As mentioned, if a certain habitat is potentially suitable for several householders, which householder has the right to decide first as to whether to occupy the habitat? Should the others then be made aware that the habitat is no longer available and to excluded it from their search? The conceptual issues pertaining to synchronization thus entail basic technical problems in systems including non-fixed objects. *Any* relocation of urban non-fixed objects demands re-evaluation of their relationships; this might cause system performance to drop to an inappropriate level. Developments in CA, MAS, temporal GIS, and Object-Oriented Database Systems indicate that there is no general solution to the synchronization problem, but the existing methodology can provide reasonable solution in each specific case (Zeigler, Praehofer, & Kim, 2000).

What follows is a minimal formal framework aimed at implementing the object-based perspective of *urban systems*, which dictated the choice of limitations we imposed on the system. The only, and again 'down to earth,' argument in favor of the proposed implementation is its adequacy for the majority if not for all urban models of which we are aware.

4. OBEUS—minimal implementation of the object-based approach

OBEUS is comprised of two levels of classes—Universal and User. The abstract classes belonging to the Universal level are considered as necessary for *any* urban process. Classes at the User level inherit these abstract base classes and implement models of specific phenomena, housing dynamics in the cases illustrated. If the phenomena are sufficiently general, which is true of residential dynamics, then many of the objects at the User level can be reused in other models. Although we may hypothesize that this transferability might dissect the User level into two or more sub-levels, one containing more general and frequently reused classes of objects, and the other application-specific ones, the issue goes beyond the limits of this discussion. We therefore assume here that the objects found at the User level are constructed anew with every application.

4.1. Basic abstract OBEUS objects—unitary, relationships, populations

The minimal implementation of the proposed object-based approach starts with classes of *unitary* urban objects of different types. This generalizes the GIS layer approach (Fig. 1) and resembles representations of human agents as superimposed on the infrastructure CA (Polhill et al., 2001; Portugali, 2000; Portugali et al., 1997). OBEUS classes of unitary objects represent both actual spatially 'located' urban objects whose location and shape are important, and 'non-located' objects for which only their non-spatial properties are considered significant. Examples of unitary located objects are land parcels, buildings, street segments, firms, public institutions, and cars. The location of landlords or the municipality is usually of minor if any value for representation of their behavior, hence, they can serve as examples of unitary non-located objects.

Another kind of OBEUS abstract class is *population*. Population objects contain data regarding all unitary objects of a specific type and aim at managing *spatial ensemble*. Typical examples of ensembles are suburbs populated by members of a particular socioeconomic group and areas exhibiting similar land use. Ensemble is a dynamic entity that emerges in space; the membership to a social, economic, or demographic group in the city is defined by objects' attributes, as in the case of ethnicity, income, or age in a case of householder objects.

As we stated, unitary objects and ensembles are insufficient for representing real-world urban systems. To construct such a system we introduce software objects that represent real-world *relationships* maintained between urban entities. These classes of relationship objects constitute the third basic component of OBEUS.

Let us consider unitary, ensemble and relationship software objects in detail.

4.2. Fixed and non-fixed unitary urban objects and their location

We distinguish between two main groups of unitary objects: those that are spatially *fixed* (immobile) after being established, that is, households, houses, lots and street segments, and those that are spatially *non-fixed* (mobile), that is, householders, cars and firms. However, while 'fixed' and 'immobile' are treated here as synonyms, the notion of 'non-fixed' is broader than 'mobile.' For example, landlords, whose role in the model is to possess property, are non-fixed because their properties can change, but the movements of landlords themselves may be irrelevant. There are different kinds of objects within each group; each kind is represented by objects of a specific software class. The mode of location depends on the group to which the class belongs.

Fixed objects are located *directly*, in a manner similar to vector GIS, that is, by means of a coordinate list. The list contains details of the object's spatial

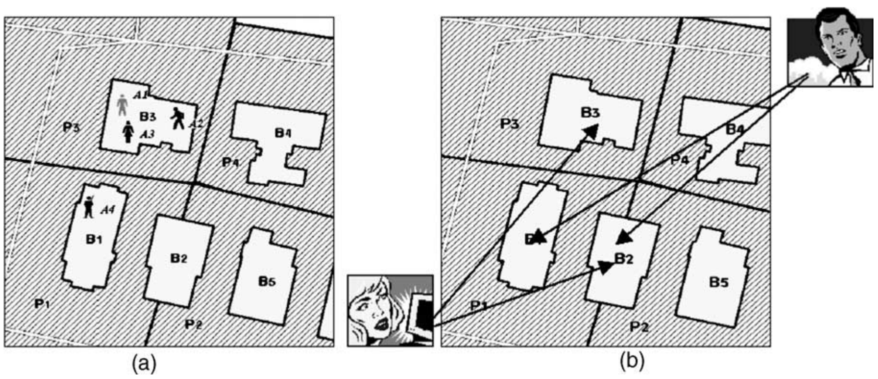


Fig. 2. Location of non-fixed object by pointing to fixed ones: (a) objects which location is important (householders); (b) objects which location is unimportant (landowners).

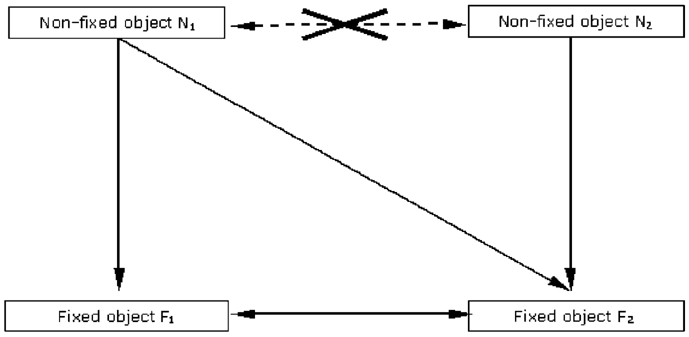


Fig. 3. Possible ways of object location in OBEUS.

representation—vertices of edges, centroid, minimal bounding rectangle, and so on. Features of planar GIS layers or 3D CAD-models can represent urban objects in spatially explicit models. For theoretical models, the points of a regular grid usually suffice.

We locate non-fixed urban objects by pointing to one or several fixed ones. For example, householders are located by pointing to their habitats (Fig. 2a). Non-fixed urban objects, whose location is unimportant, are nonetheless related to fixed ones by pointing; for example, landlords point to their properties (Fig. 2b).

We do not allow in OBEUS for direct relationships between non-fixed objects (Fig. 3). The reasons are nothing but utilitarian. First, spatial relations between fixed objects are also fixed after these objects are established; thus, they can be estimated and updated only when new fixed objects are created. Second, relationships between non-fixed objects can be easily obtained as a superposition of relationships of the non-fixed–fixed and fixed–fixed type. Third, re-estimation of non-fixed–non-fixed relationships can become unwieldy if many non-fixed objects change their location simultaneously.

All urban models we are aware of can be interpreted in terms of fixed–fixed and non-fixed–fixed relationships. The latter are sufficient for representing the relationships between non-fixed objects in models of residential dynamics (Benenson, 1998; Benenson et al., 2002; Portugali et al., 1994, #32; Portugali et al., 1997); pedestrian dynamics (Batty, DeSyllas, & Duxbury, 2002; Blue & Adler, 2001; Gipps & Marksjo, 1985; Turner & Penn, 2002; Weifeng, Lizhong, & Weicheng, 2003); and automobile traffic (Chowdhury, Santen, & Schadschneider, 2000; Hidas, 2002; Nagel, Rickert, Simon, & Pieck, 2000; Nagel & Schreckenberg, 1992; Schadschneider, 2000; Wolf, 1999).

To obtain the flavor of models where direct relationships between non-fixed objects is essential, we forward the reader to descriptions of the dynamics of flocks and fish schools (Brogan & Hodgins, 1997; Levine & Rappel, 2001; Reynolds, 1987; Toner & Tu, 1998; Vicsek, Czirok, Ben Jacob, Cohen, & Schochet, 1995) in addition to the 'Boids' site, http://www.red3d.com/cwr/boids/.

4.3. Management of time in OBEUS

OBEUS architecture utilizes the discrete time approach and implements two synchronization modes.

Synchronous. In this mode, all objects are assumed to change simultaneously. The calling order of the objects has no influence in this mode, but conflicts arise when agents compete over limited resources, as in the case of two householders trying to occupy the same apartment. Resolution, if ever, of these conflicts depends on the model's context, a decision OBEUS leaves to the modeler. It is worth noting that the logic of synchronous updating often pushes conflicts further down the time path. If two mutually avoiding agents (i.e. the probability to leave the location for each one increases when the other is nearby) occupy adjacent locations and simultaneously leave them at a given time-step, then in synchronous mode nothing can prevent occupation of these locations by another pair of avoiding agents.

Asynchronous. In this mode, objects change in turn, with each observing the urban reality as left by the previous object. Conflicts between objects are thereby resolved; instead, the order of updating (often, but not necessarily, random) is critical as it may influence results. OBEUS demands that the modeler define an order of object actions according to several templates; random sequence, sequence in order of some characteristic, and object-driven sequence are currently being implemented.

4.4. Relationships between urban objects as software objects

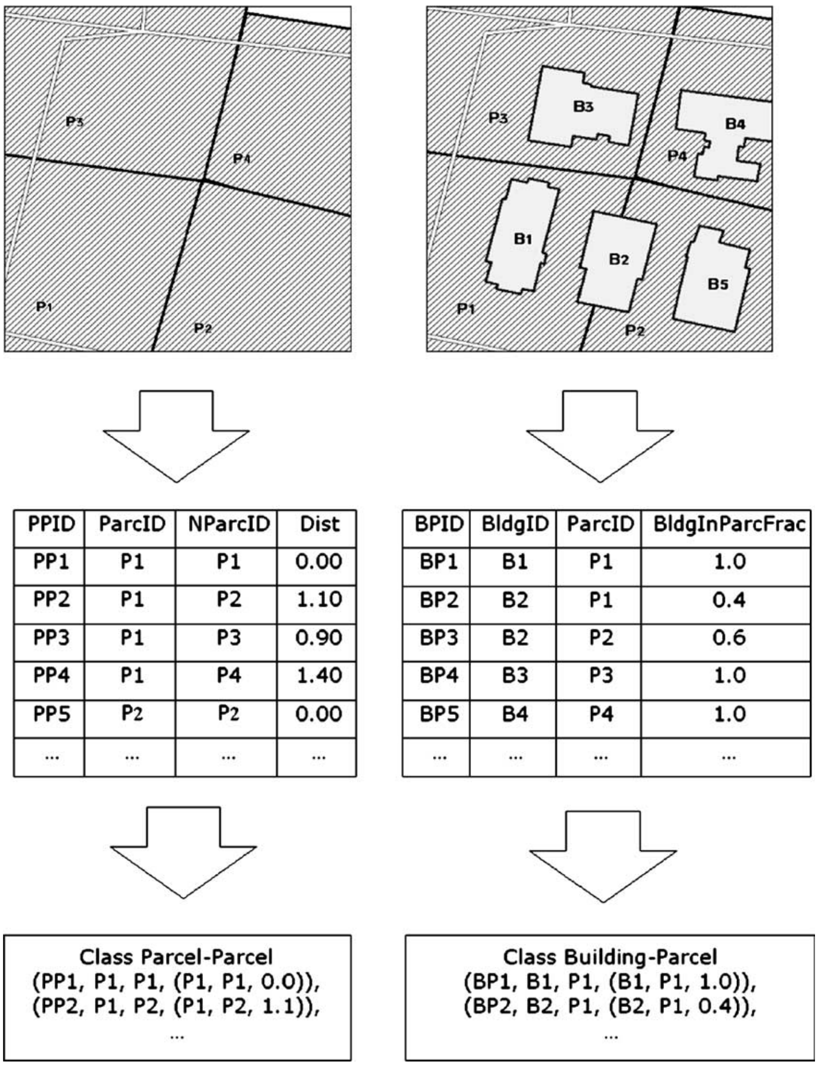
In OBEUS, we combine the requirements of ERM and OO approaches by considering relationships between unitary objects of types A and B as a new class of objects—an AB_Relation—that encapsulates the relationship's attributes. Objects of classes A and B maintain references to objects of class AB_Relation and vice versa. Fig. 4 presents examples of apparent transformation of neighborhood relationship between parcels, and the overlap relationship between parcels and buildings retrieved in GIS and stored according to an ERM model into an object of AB_Relation class. It is worth noting that by so doing, we implement the Object-Oriented Database (OODB) view on the urban system (Booch, 1994).

To make OBEUS appropriate for the geographic modeler, we impose some essential limitations on the semantics of the relationships held between urban objects. The objective of these constraints is to preserve updating consistency and utilize the fixed/non-fixed dichotomy. First, we assume that relationships between fixed objects are also fixed, that is they do not change after being established. Second, as mentioned above, in OBEUS we do not permit direct relationships between non-fixed objects. Non-fixed unitary objects in OBEUS can be related to fixed objects only; in this way locating of non-fixed objects is a special case of relationship. Third, we encapsulate the methods that alter relationships within non-fixed objects and assume that non-fixed objects have exclusive update rights, while the related fixed objects can only query the relationship's attributes. One implication of these constraints is that neighborhood relationships are defined for fixed objects only. The householder's neighbors, for example, are thus defined as inhabitants of the houses adjacent to the householder's house.

Direct relationships between non-fixed objects are undoubtedly important in the real world. For example, aggressive pedestrians can clash when several are spatially packed in a shop's queue. Nonetheless, this relationship is adequately represented for all pedestrians by simply pointing to the shop itself. We are simply unaware of high-resolution urban models that account for direct relationships between non-fixed objects: As already mentioned, in the great majority of the important situations we are aware of, relationships between non-fixed objects are traced via relationships between these non-fixed objects and fixed objects.

4.4.1. Leader–follower pattern of relationships in OBEUS

OODB theory pays considerable attention to the problem of managing relationships between objects. The previous example of a property whose ownership is



PPID	ParcID	NParcID	Dist
PP1	P1	P1	0.00
PP2	P1	P2	1.10
PP3	P1	P3	0.90
PP4	P1	P4	1.40
PP5	P2	P2	0.00
...	...	...	...

BPID	BldgID	ParcID	BldgInParcFrac
BP1	B1	P1	1.0
BP2	B2	P1	0.4
BP3	B2	P2	0.6
BP4	B3	P3	1.0
BP5	B4	P4	1.0
...	...	...	...

Class Parcel-Parcel
(PP1, P1, P1, (P1, P1, 0.0)),
(PP2, P1, P2, (P1, P2, 1.1)),
...

Class Building-Parcel
(BP1, B1, P1, (B1, P1, 1.0)),
(BP2, B2, P1, (B2, P1, 0.4)),
...

Fig. 4. Examples of 1:*N* neighborhood relationship and *M*:*N* relationship between objects of two different types and their implementation in OBEUS.

shared by two landlords and the conflict arising when one landlord wishes to sell her share while the other does not want to transmits some of the problem's flavor. Temporal GIS faces the same problem regarding the updating of connected or adjacent features whose shape can change over time (Miller, 1999); for instance, when the shape of one of two contiguous polygons of the continuous coverage must be changed. This problem applies to all three of the relational types, that is, 1:1, 1:*N*, or *M*:*N*.

The problem of managing relationships has no one general solution; instead, several are proposed, each depending on the nature of the objects and the relationship's semantics. One can refer to the computer science literature for complex situations (Peckham, MacKellar, & Doherty, 1995). In OBEUS, we follow the development pattern proposed by Noble (2000). To retain consistency of relationships, an object on one side, termed the *leader*, is responsible for managing the relationship. The other side, the *follower*, is comprised of passive objects. The leader provides an interface for managing the relationship, and invokes the followers when necessary.

There is no need to establish leader or follower "roles" in a relationship between fixed objects once the relationship is established. In OBEUS, in relationships between a non-fixed and a fixed object, the non-fixed object is always the leader and is responsible for creating and updating the relationship. For instance, in a relationship between a landlord and her property (when ownership cannot be shared), the landlord initiates the relationship and is able to change it. In relationships between a car and a parking place, the car is the leader.

Non-fixed objects are natural candidates for leader in the case of a M:N relationship as well. For example, landlords who can share ownership of several parcels are the leaders in a landlord–parcel relationship, just because they make the decisions concerning their property in the real world. Another example can be a relationship between an apartment and its residents, when more then one resident can occupy the apartment. The mobile householders are the relationship's leaders in this case as well; they decide when to vacate the apartment.

It is worth noting that M:N relationships are difficult to implement and might incur significant overhead both in terms of development time and performance. We permit M:N relationships in OBEUS but tend, when possible, to consider the one-to-many alternative as a substitute for the same relationship.

The leader and follower should be defined separately for each relationship. In addition to than consistency, the latter ensures that the algorithm implemented for relationship updating will result from *intentional* thought. There is no proof that the majority of real-world urban situations can be imitated by the *leader–follower* pattern, although we are not aware of any instances of urban models where more complex patterns (circular or multiple causation, for example) are applied.

4.4.2. Advantages of introducing relationships as software objects

The distinction between objects and relationships, and employment of the leader–follower pattern, offer immediate advantages for software applications. For example, it provides an identical framework for CA and explicit GIS-based land-use models, both of which are based on neighborhood relationships. For square CA grid and von Neumann or Moore neighborhoods, the degree of neighborhood relationship is either 1:4 or 1:8, while for White and Engelen's (1993) constrained CA, the degree of neighborhood relationship is 1:113, based on neighborhoods of radius 6 around the cell. The only difference between CA models and GIS-based representation of land units based on Voronoi or other partitions of a plane (Benenson et al., 2002; Flache & Hegselmann, 2001) is the variation in degree of neighborhood relationship from parcel to parcel, which is identical for cells of the CA. The distance between the cell and its neighbor, the length of the common boundary and other factors can be considered attributes of a *neighborhood relationship* object (a member of the AA_Relationship class).

Relationships between fixed objects should be initialized when fixed objects are initiated and after that retrieved only. Relationships between non-fixed and fixed objects are initiated or eliminated according to requests from the former because they are always leaders within the relationships.

4.5. Population as a container for meta-data

Urban objects always deal with information found on levels above the individual and local level. Human residential agents, for example, have access to newspapers where lists of apartments for sale are published; these lists are properties of the set of habitat objects as a whole. To capture this comprehensiveness, we introduce *population* software objects into OBEUS.

4.5.1. Population and self-organizing ensembles

Each urban simulation eventually approaches the enigma of self-organization (Portugali, 2000). In OBEUS, we attempt to grasp one aspect of self-organization only, namely, the emergence of ensembles of closely located unitary objects exhibiting similar characteristics. We call these ensembles *domains*. Domains intuitively represent industrial or commercial areas, rich or poor neighborhoods, or areas displaying a specific architectural style. We assume that domains can be determined by fixed predicates, defined on unitary objects. Stated formally, the set of unitary objects D_C form a domain, satisfying criterion C if

1. For each $d \in D_C$, a sufficient number of d's neighbors (but not necessarily d itself) satisfy criterion C.
2. D_C contains a sufficient number of unitary objects.

A simple algorithm recognizing emerging domains and following their evolution is proposed (Appendix A). This algorithm is very efficient because any object belonging to a domain defines it in its entirety. Fig. 5 presents a simple illustration of "Dark gray" domain D_G, with criterion G demanding that "more than 50% of the neighbors in a cell within a 3×3 Moore neighborhood are dark gray." Note that, first, the criterion can be arbitrarily complex, and, second, the domain can consist of several continuous parts.

The importance of domains for representing urban phenomena is evident. Let us return to housing. A suburb containing many expensive apartments is considered an expensive neighborhood; the value of other estates consequently increases. One can investigate the consequences, for the city, of the positive feedback generated by the unsatisfied demand of wealthy agents who enter the area, buy properties, and subsequently reinforce these areas as 'expensive' domains. Alternatively, an expensive domain can disappear when wealthy agents became aware of a nearby polluting

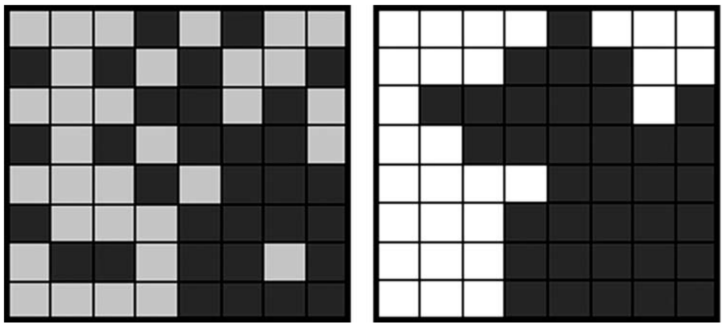


Fig. 5. Illustration of domain criterion. Cell belongs to 'Dark gray' domain D_G if 50% or more of its neighbors within a 3×3 Moore neighborhood are dark gray.

factory and migrate out of the area. Another domain criterion may be based on the age of urban objects (e.g., the age of buildings), an idea utilized in deltatron CA (Candau, Rasmussen, & Clarke, 2002; Clarke, 1997).

Domains are obviously limited to capturing 'foreseeable' self-organization because their criteria are defined a priori. To weaken this limitation, criteria can be modified during experiments according the effects 'observed' or searched for on the screen (Fig. 7).

4.5.2. Population time versus time of unitary objects

In OBEUS, we distinguish between *population time* and *time of unitary objects*. Population objects are updated (that is, their domain criteria are tested, and the domains are recognized, updated, or eliminated with respect to the state of the unitary objects) in the course of population time, which is independent of the time of unitary units and follows its own synchronization agreements. It is natural to expect population time to be inherently 'slower' than the time of the unitary objects. Hence, methods managing population domains are applied less frequently than methods managing unitary objects. It is worth noting that the concept of different times for objects of different types can easily be generalized and that objects of each type can be considered within their own time frame. In OBEUS, we accept the convenient assumption that there exists a minimal time unit τ; we can then express the time units of different objects and populations in units of τ.

4.5.3. Relationships between domains and unitary object

As with non-fixed objects, domains in OBEUS can be related to fixed objects only. These relationships can capture properties such as distance between unitary fixed objects and the domain; several definitions of distance based on objects' and domains' centroids, boundaries, etc. can be evidently applied. Analysis of available urban models demands consideration of domains of fixed as well as non-fixed objects (low-cost dwellings and poor householders, for example).

Domains can always change, which automatically makes them leaders in the relationship with unitary objects. As in the case of non-fixed objects, OBEUS ignores direct relationships between domains and non-fixed objects or domain–domain relationships.

5. Abstract classes of OBEUS

5.1. Abstract base classes

Abstract OBEUS classes represent all the basic objects defined above: unitary urban objects, populations, relationships, and domains. These classes are organized in two-level hierarchy:

Entity represents unitary objects, and is inherited by
- *Estate* representing fixed (immobile) objects, and
- *Agent* representing non-fixed (mobile) objects

Domain represents sets of predicates that define domains.
Relationship represents relationship objects and is inherited by
- *Estate–Estate*,
- *Agent–Estate*,
- *Domain–Estate*

Population represents meta-properties and meta-methods for objects of a specific type.

Fig. 6 presents the hierarchy of OBEUS abstract classes and their main methods by means of a UML diagram (Booch, Rumbaugh, & Jacobson, 1999).

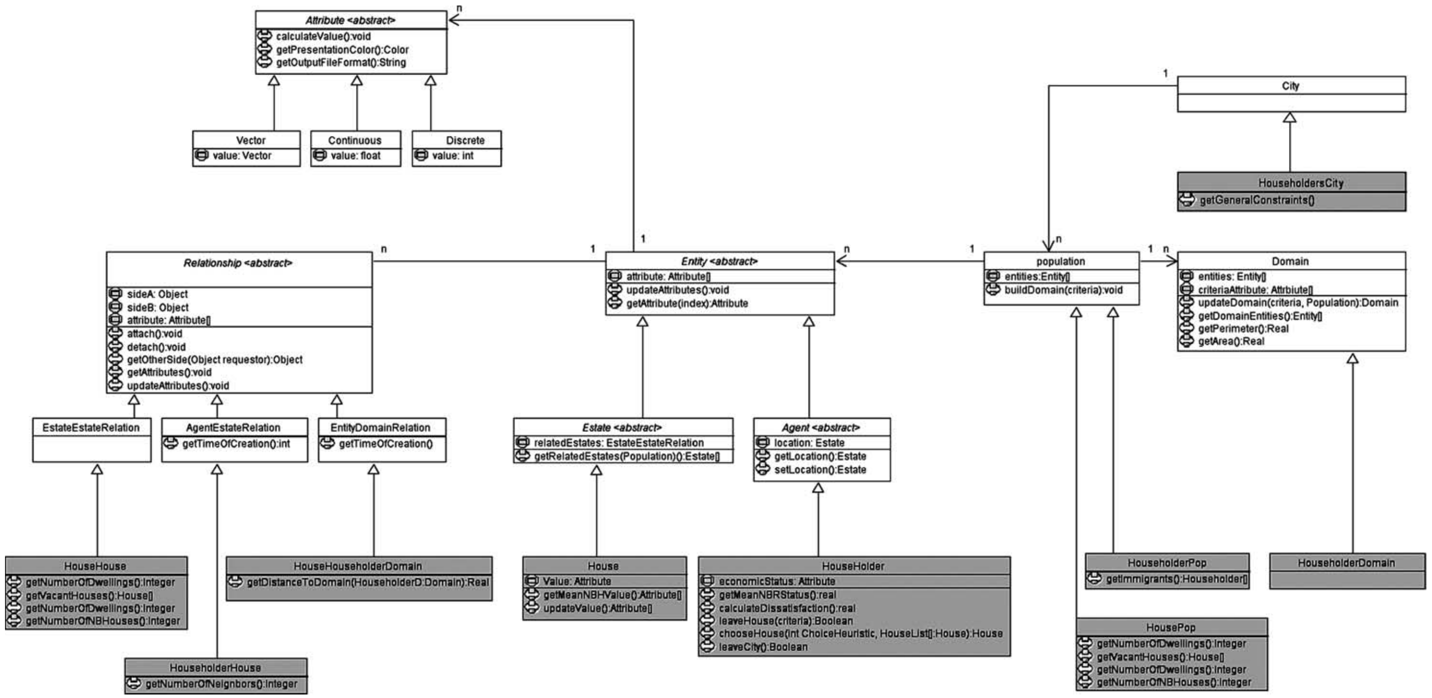
It is worth noting that none of the urban models we are aware of accounts for more than two of these relationship classes; the majority account for only one if any relationship class.

5.2. The 'City' as a singleton

Some of the properties of the urban system relate naturally to the city as a whole. These properties concern relationships between populations' attributes, as, say, the mean number of apartments in a building. The software class *City* represents the city as a whole. The software manages a single instance of the City class, a *singleton* according to Gamma, Johnson, and Vlissides (1995).

5.3. Class attribute

An abstract class *Attribute* is not directly related to the idea motivating this paper but it is useful and convenient. In OBEUS, attributes are typed, and common data types are supported, as are arrays. Attribute arrays are of the same scalar type so as to support internal comparisons between the array's different elements (for example, different components of the householder income).



Attribute <abstract>
calculateValue():void
getPresentationColor():Color
getOutputFileFormat():String
Vector
value: Vector
Continuous
value: float
Discrete
value: int
City
HouseholdersCity
getGeneralConstraints()
Relationship <abstract>
sideA: Object
sideB: Object
attribute: Attribute[]
attach():void
detach():void
getOtherSide(Object requestor):Object
getAttributes():void
updateAttributes():void
Entity <abstract>
attribute: Attribute[]
updateAttributes():void
getAttribute(index):Attribute
population
entities:Entity[]
buildDomain(criteria):void
Domain
entities: Entity[]
criteriaAttribute: Attrbiute[]
updateDomain(criteria, Population):Domain
getDomainEntities():Entity[]
getPerimeter():Real
getArea():Real
EstateEstateRelation
AgentEstateRelation
getTimeOfCreation():int
EntityDomainRelation
getTimeOfCreation()
Estate <abstract>
relatedEstates: EstateEstateRelation
getRelatedEstates(Population)():Estate[]
Agent <abstract>
location: Estate
getLocation():Estate
setLocation():Estate
HouseHouse
getNumberOfDwellings():Integer
getVacantHouses():House[]
getNumberOfDwellings():Integer
getNumberOfNBHouses():Integer
HouseHouseholderDomain
getDistanceToDomain(HouseholderD:Domain):Real
House
Value: Attribute
getMeanNBHValue():Attribute[]
updateValue():Attribute[]
HouseHolder
economicStatus: Attribute
getMeanNBRStatus():real
calculateDissatisfaction():real
leaveHouse(criteria):Boolean
chooseHouse(int ChoiceHeuristic, HouseList[]:House):House
leaveCity():Boolean
HouseholderPop
getImmigrants():Householder[]
HouseholderDomain
HousePop
getNumberOfDwellings():Integer
getVacantHouses():House[]
getNumberOfDwellings():Integer
getNumberOfNBHouses():Integer
HouseholderHouse
getNumberOfNeignbors():Integer
n
1
n
1
1
n
1
n
1
1
n

Fig. 6. Hierarchy of OBEUS abstract classes (transparent blocks) and extension of OBEUS for residential dynamics model (gray blocks).

6. OBEUS-housing—the object-based approach as applied to residential dynamics phenomena

The aim of user-level classes of OBEUS is to represent models of specific urban process. What follows is an application of OBEUS for the modeling of housing dynamics, the field of application in which we have maximal experience (Benenson, 1998; Benenson et al., 2002). The specific residential dynamics model we consider is based on the stress–resistance hypothesis (Phipps, 1988). According to this hypothesis, residential choice consists of three stages. At the first stage, householder agents estimate their dissatisfaction regarding their current location and decide whether to move it or not. If yes, at the second stage a householder agent scans vacant habitats, estimates dissatisfaction at each of them, and constructs a list of vacancies feasible for relocation. At the third stage, an agent tries to relocate and does so if the potential habitant is still empty. We assume further that householder agents act synchronously at each stage.

We will shortly illustrate an OBEUS implementation on two versions of the model. The first is an abstract model, aimed at studying self-organization of rich and poor neighborhoods in the city as an outcome of the interplay between the economic status of householders and the price of housing. The second, an explicit spatial model of residential dynamics in the Yaffo residential area of Tel-Aviv, aims at simulating the spatial dynamics of the Arab and Jewish population over the period 1955–1995. The results of these studies have already been published partially or in full. The abstract version (Benenson, 1998, 1999; Portugali & Benenson, 1995, 1997) is treated first; the explicit version (Benenson et al., 2002) follows.

6.1. User-level classes

Interfaces of user classes deal with the objects specifically associated with a model, in this case housing, that is, updating dwelling objects, relocating householder objects, managing domains representing areas of relatively homogeneous population and so on.

The classes implemented at the user level are as follows (Fig. 6):

City is inherited by
- *HouseholderCity*

Population is inherited by
- *HousePop* and
- *HouseholderPop*

Estate is inherited by
- *House*

Agent is inherited by
- *Householder*

Estate–Estate relationship is inherited by
- *House–House*

Agent–Estate relationship is inherited by
- *Householder–House*

Domain is inherited by
- *HouseholderDom*

Domain–Estate relationship is inherited by
- *HouseholderDom–House* relationship.

The above structure is still 'neutral' regarding the objects' attributes. The abstract version of the model investigates the influence of self-organizing domains of 'reach' and 'poor' householder agents. For studying 'economic' self-organization attributes of *House* objects, the relevant attribute is its *value*, whereas the relevant attribute of *Householder* objects is their *economic status*. We do not include here details of value and income dynamics because they influence the objects' attributes only.

To construct relationship objects of the *House–House* class we assume that houses $\mathbf{H}_n$ that are the immediate neighbors of the given house H and the immediate neighbors $\mathbf{H}_{nn}$ of each of $\mathbf{H}_n$ are accounted for. The influence of all neighboring houses is assumed the same. Different definitions of the immediate neighbors are employed.

To investigate the self-organization of residential distribution in the city, criteria of domain should be given, and we use the following case: The house has more than **f** percent of neighboring houses, for which the mean income of the householders is above **p**th and below **q**th percentiles of the distribution of city householders by income ($\mathbf{f}, \mathbf{p}, \mathbf{q}$-parameters, $\mathbf{p} < \mathbf{q}$). We investigate the model for different sets of $(\mathbf{f}, \mathbf{p}, \mathbf{q})$; usually, three criteria C_i, $i = 1, 2, 3$, are employed with pairs of $(\mathbf{p}_i, \mathbf{q}_i)$ denoting non-intersecting intervals of 'low,' 'medium,' and 'high' economic status. The width of the $(\mathbf{p}_i, \mathbf{q}_i)$ interval and the minimal number of houses in continuous part of domain vary in model scenarios. The typical value of **f** is about 0.5–0.8.

The methods employed at the user level are specified in line with the stress–resistance hypothesis. The basis for hypothesis, dissatisfaction function is implemented as the *calculateDissatisfaction* method of *Householder* class. The method calculates dissatisfaction of an agent depending on:

- mean price of dwellings in the house and in the neighboring houses (method *getMeanNBHPrice* of the *House* class);
- mean economic status of the neighbors (method *getMeanNBRStatus* of the *Householder* class);
- distance to closest domain D_i defined by criterion C_i (method *getDistanceToDomain* of the *House* class).

Boolean *leaveDecision* of the *Householder* class returns 'true' or 'false,' at the first stage of migration behavior, depending on the decision of the householder agent to try to resettle or remain in the house.

We assume that householder agents have access to information regarding vacant houses throughout the city (we call this information 'global') when choosing new residences. The *getVacantHouses* method of the *HousePop* class returns a list of vacant houses that satisfy certain criteria, which, just as in the case of a dissatisfaction function, are based on the properties of the householder and properties of the other householders in the house, house itself, neighboring houses, neighboring householders, and domains existing at that moment.

Finally, the householder has to choose to occupy one of the houses on the list or cancel the choice. The *chooseHouse* method of the *Householder* class chooses a house from the list returned by the *getVacantHouses* method. If succeeded, the existing relationships of the *Householder–House* class are destroyed and new ones are created by *setLocation* method inherited of the *Agent* class. The *chooseHouse* method depends on the choice heuristic, for example, choose the vacancy having the lowest value of *calculateDissatisfaction*, choose the vacancy inversely proportional to values returned by *calculateDissatisfaction*, etc.

The model operates in synchronous mode. The time unit for updating unitary objects (houses and householders) is three months, and this time unit determines the time-related model parameters: the probability that a householder will leave the house, for example. The time unit for domain updating is set equal to one year (that is, four of the unitary objects' time units).

6.2. Example of the explicit model of housing

The explicit model of the residential dynamics of the Yaffo area of Tel-Aviv is based on the interplay between the ethnicity of the householder, of the neighbors and the architectural style of the house (Benenson et al., 2002). We suggest in this model that ethnicity of neighbors influence householder dissatisfaction; in addition, agents of different ethnicity prefer houses of different architectural style, that is *House* objects are characterized by their *architectural style* and *capacity*, both exported from the high-resolution GIS of the area (Benenson et al., 2002), while *Householder* objects by their *ethnicity*.

The neighbors of the house are defined as above, based on immediate and second-order neighbors. The neighbors are defined by Voronoi coverage, constructed on the base of houses centroids. Houses are considered as immediate neighbors if they are not too distant from one another, their Voronoi polygons have common boundaries and they are located on the same side of the main road. We do not account for domains in the explicit model.

All methods applied in the abstract version of the model remain valid, but are modified to incorporate new parameters pertaining to the objects. The time units for unitary objects and populations are, as above, three months and one year, respectively; parallel updating is employed. The model's results were published in (Benenson et al., 2002).

To investigate the model per se, different non-spatial initial distributions of house and householder properties and different rules of status and value changes were investigated. In the explicit, real-world version, house properties were obtained from the GIS layers of Yaffo's houses. Householder agents were initialized on the basis of distributions of the socio-economic characteristics of Tel-Aviv families in 1955 and located in houses on the basis of correlations between apartment and householder properties. The model results were compared to the aggregate data available at three time points within the period 1955–1995 and with geo-referenced data at resolution of separate family available from Israel's 1995 population census and exhibit very good fit (Benenson et al., 2002).

To indicate the advantages of the object-based approach, we stress again that besides the differences noted above, the properties and methods of the model's user-level classes in abstract and the explicit versions are identical.

6.3. Initialization of objects in OBEUS-housing

Initialization methods are always encapsulated in user-level classes. In the abstract model, they are usually very simple; for example, *House–House* relationship objects are initialized on the base of Moore 5×5 neighborhood. In spatially explicit models, OBEUS classes are initialized on the basis of GIS information and usually demand external preprocessing within GIS, as we did above based on the coverage of Voronoi polygons, constructed on the basis of house centroids.

6.4. Performance of OBEUS-housing

The object-based approach demands multiple updates of relationship objects and domains; these updates raise concerns about the models' performance. In OBEUS-housing for example, updating relationships between householders and houses following residential migrations and re-estimation of domains can cause performance problems when the numbers of householders and houses reach urban standards of 10^4–10^6; as the city of Tel-Aviv, which contains about 1.5×10^5 householders and 2×10^4 houses. Tel-Aviv dimensions correspond to an abstract model on a 150×150 grid with cell capacity of 6. Another dimension of the simulation—the number of the neighbors—resulted in 25 neighbors in the abstract version (including house as a neighbor of itself), while in the explicit version, where Voronoi polygons were employed, the average number of neighboring houses was reduced to 14 on average. The number of domain criteria that are regularly tested—three in the abstract version of a model—presents one more dimension.

According to the stress–resistance hypothesis, the model step consists of looping over householders, estimating dissatisfaction and the probability of migrating for each, constructing lists of vacancies for those who decided to migrate, concluding with relocation if possible. Agents' and houses' properties are then synchronously updated, and domains re-estimated every four iterations.

The estimates obtained in the abstract version of OBEUS-housing are quite satisfactory. The time required for a model step by a single 1.7-GHz Pentium 4 processor is of an order of several seconds. Moreover, it is important to note that the *average number of relationships per object* does not change with the growth in the number of houses or householders in the city. Therefore, we can guarantee that

OBEUS performance decreases linearly with the increase in the number of urban objects.

7. Conceptual comparison of OBEUS to other agents software

SWARM and SWARM-based software, such as MAML (Gulyas et al., 1999), EVO (Krumpus, 2001) or those made from scratch, as RePast (RePast, 2003), are most popular in urban and social agent-based simulations. Many other agent-based software libraries (Anonymous, 2002) are available. Compared to them, two basic features are characteristic of OBEUS.

Conceptually, we aim at capturing spatial dynamics of urban systems as self-organizing systems. Consequently, we require that each unitary object be located either directly or indirectly, we divide the model world into populations that reflect GIS layers, and introduce domains to capture self-organizing entities at the above-unitary level.

In practice, we tend to manage the urban system as closely as possible to an OODB. Domains enable a simple but powerful approach to on-the-fly capturing of spatial ensembles, the latter being basic elements of modern urban theory. Isolation of relationships enables reuse of software objects and incorporation of new population into the model without modifying the existing software objects.

Last, but not least, we do not tune system capabilities to anything beyond simulation of urban spatial dynamics. One can easily provide examples when OBEUS logic becomes inconvenient (e.g., complex non-spatial economic models).

The existing implementation of OBEUS is a prototype; we still are not ready with the shareware version. The project's progress can be followed on http://eslab.tau.ac.il.

8. Discussion: solutions and problems inherent to the object-based approach

8.1. Solutions

The object-based approach is simple and flexible. Most importantly, every published high-resolution dynamic urban model we are aware of, if described in sufficient detail, can be directly interpreted in an object-based approach. Most urban models consider fixed objects and, in consequence, fixed relationships only. The models that consider mobile and, more generally, non-fixed objects are few in number; they all satisfy the OBEUS framework. Stated briefly, the limited form of interactions and self-organization that OBEUS permits satisfies modelers' demands for the capture of urban system complexity. It is thus not surprising that despite its many limitations, OBEUS fits all the *urban* models known to us.

Attributes of flexibility and simplicity are exceptionally important: with the increase in the number of urban models published, our ability to *compare* models and their results fades. As Berec (2002) discusses in relation to ecological models, the reasons for this incapacity are quite obvious: Model publication aims at explaining ideas while it avoids discussion of the "unimportant" details of application; in addition, simulation software code cannot be published. As a result, the reader is never certain about the differences between models, synchronization mode, relationship management and the way in which self-organization is accounted for. Nor should we neglect the modeler's nightmare when researchers concentrating on marginal factors miss those that 'really' influence outcomes. The only possible solution is to increase comparability, that is, to provide the *ability to rerun simulation models* without the support of their authors. This can be accomplished only if researchers share the same *approach to software development*, a goal supported by the object-based paradigm.

Being down to earth, we appreciate that conceptual advantages are insufficient to convince a modeler to travel along the object-based lane. The decision depends on a number of fine points we wish to convey:

1. In the object-based approach, implementation is identical—literally—for the case of an abstract cell-space and real-world GIS-based environment. The user can thus combine investigation of abstract ideas regarding the model mechanisms, structure of space, and so forth, with applications to the real city, just as we did when investigating residential dynamics in Yaffo.
2. A very important aspect of computer experimentation is coherent treatment of events in time. The challenge lies not only in creating a mechanism that will manage concurrencies but also a mechanism that makes modelers aware of their implicit assumptions about how multiple threads of time interact within their experimental setups. The object-based approach forces the user to make explicit assumptions regarding the concurrency of object changes and interactions. Other researchers can subsequently reproduce their results.
3. Domains capture emergence and self-organization in urban space.
4. Cellular Automata are directly adaptable to the object framework. All we need do is represent neighborhood relations between cells by means of a neighborhood relationship class.
5. The object-based approach directly adopts Multi-Agent Systems defined on a cellular space.
6. Predefined aggregate objects such as administrative regions can be directly incorporated into object-based models. Despite their semantically aggregate meaning, formally, such aggregate objects can be considered as fixed unitary objects and managed in the standard way, including their relationships to unitary objects, they consist of, such as parcels
7. Inherently continuous urban characteristics—topography or pollution, for instance—do not demand modification of the scheme. Despite their internal continuity, these characteristics are always stored and updated in discrete time and space, based on TIN or lattice formats. To fit continuous fields to the

object-based model, we simply consider these fields in discrete space as determined by the objects themselves.

8.2. Problems

There are two problems that currently seem to be especially salient:

(1) The user interface for formulating 'rules of behavior' for urban agents objects:

Our discussion of the object-based approach and its realization in an OBEUS environment intentionally ignores the user interface. Let us consider the 'output' and 'input' ends separately.

Output. At this stage we delay investing in the output part of the OBEUS environment. On the one hand, it is impossible to attain the sophistication of commercial spreadsheets, statistical packages, and GIS; on the other, the model environment and the presentation software are easily linked. As a result, the output part of the OBEUS interface contains only three components: presentation of spatial pattern (Fig. 7), graphs of temporal dynamics of some predetermined integral statistics (as population averages) and writing characteristics of objects and domains into the output files. We assume that the user will further process these files with statistics and presentation software.

Input. The situation is similar regarding initialization of the model's numeric parameters yet qualitatively different regarding the rules of object behavior. We would argue that the next step in the transition of urban modeling from art to engineering is incorporation of a *language for formulating objects behavior* into OBEUS. Java, which we employed in the current application, like any other standard programming language, demands expertise that urban modelers do not necessarily have. The solution may lie in a meta-language (Galton, 2003) and the capacity of many existing computer languages regarding the 'behavior' of urban objects' relocation (Schumacher, 2001), should be tested, something that remains to be done.

(2) Changes in the shape of unitary objects:

The assumption that fixed objects do not change their shapes nor, ipso facto, their relationships, is a fundamental tenet of the proposed framework and satisfies the majority of current demands. At the same time, it limits the applicability of the approach precisely because in reality, land units sometimes do change, for example, when agricultural parcels are transformed into urban parcels. The only rigorous model of re-partition we are aware of, that proposed by Semboloni (2000), assumes that two adjacent parcels can accumulate 'partition potential' and produce a new parcel centered at the midpoint of the edge connecting the parcels' centroids. The neighborhood of a new parcel is then rebuilt *locally* according to the Voronoi tessellation algorithm. Local repartition does indeed seem to be the most prevalent change that land parcels, buildings and road segments undergo from time to time. The question is, then, should we include such or a similar possibility within the general framework? We have not yet arrived at a definite answer to this question. Arguments 'pro' include the importance of local repartition for urban dynamics and the possible influence exerted by other urban processes on the repartition intensity.

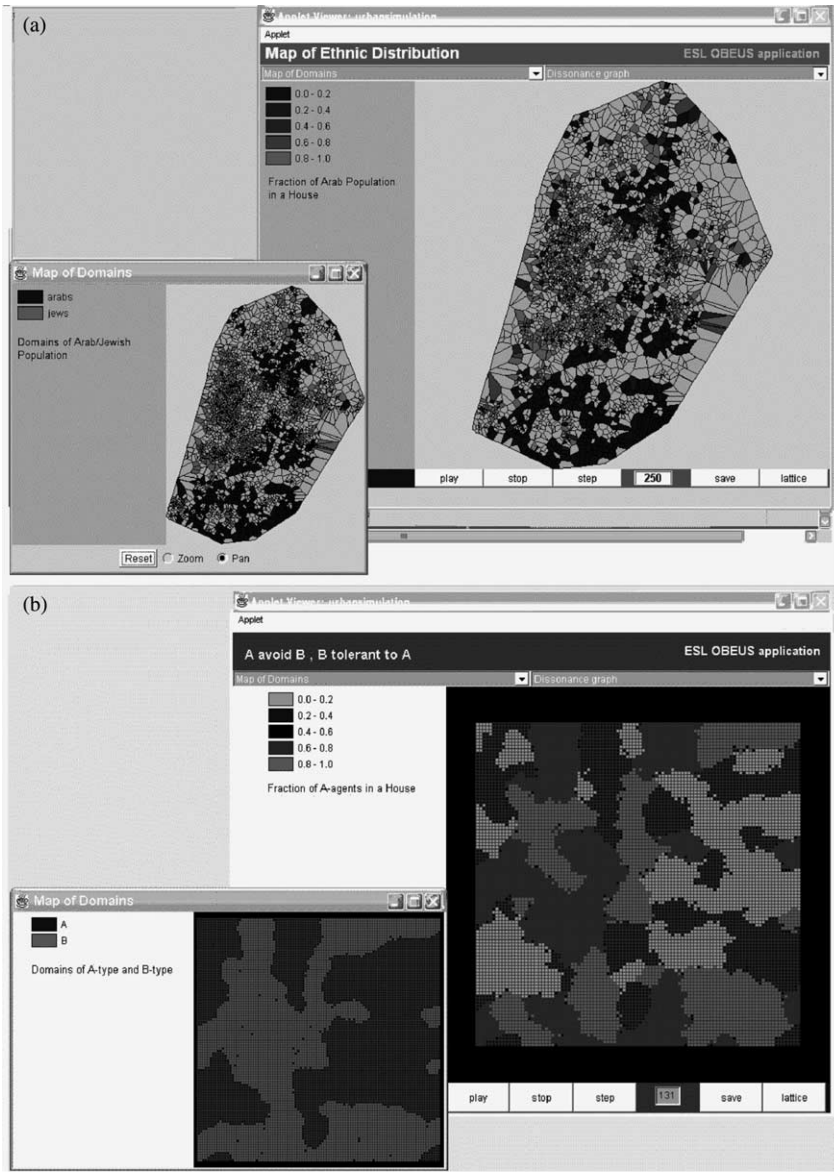


Fig. 7. Elements of OBEUS interface—maps of objects and map of domains: (a) real-world model of Jewish-Arab residential dynamics in Yaffo; (b) abstract version of residential dynamics model.

The major argument 'con' is the overhead incurred by extension of model classes, extensions left unemployed in most applications. Nonetheless, we do not foresee any significant problem in introducing local repartition into the methods of fixed object classes, just because the real-world rules of repartition are simple and never bring up the problem of concurrency. More examples of urban models that demand repartition should be accumulated before reaching a decision, which might be coexisting lighter and heavier versions of the software.

9. Conclusion

Several decades ago, dynamic regional modeling began with a common reference, i.e., the system of differential or difference equations that describe the change of the aggregate state variable of regions over time. This meant that one could easily distinguish between equation-based models by comparing their analytical presentations and parameter values. With time, we abandoned the regional framework in favor of high-resolution simulations. This evolution, however, revives the problem of sharable reference, which is compulsory if we want to advance urban modeling beyond art to engineering. We argue that the object-based platform provides a starting point for progress as well as the basis for the next step: development of an urban simulation language.

Appendix A. Simple algorithm of domain recognition

For criterion C and corresponding domain D_C, let us classify a building as expensive if the following criterion is fulfilled:

C: *The value of more than half of apartments in the building is above the 90th percentile of the distribution of apartment values in a settlement.*

Let us mark the buildings satisfying C as C-TRUE and the rest as C-FALSE.

The domain D_C of "expensive dwellings" consists of continuous parts, here called regions. An algorithm for constructing domain D_C is based on the idea that each region $R_C \subseteq D_C$ contains at least one C-TRUE building B. The procedure that constructs R_C—the region containing a C-True building *Ent*—is as follows:

```
buildRegionAroundEntity(Criterion C, entity Ent, float $F_{CThreshold}$, int $N_{CThreshold}$)
    Construct empty temporary domain R;
    Insert Ent into R;
    While there are new entities in R {
        Get next entity currEnt from recently included into R
            Get list currNBRH of neighbors of currEnt
        Calculate fraction $F_C$ of C-TRUE Entities in currNBRH
        If $F_C > F_{CThreshold}$ then {Include currEnt in R}
        Else remove currEnt from R}
    Calculate $N_R$-number of buildings in R
    If $N_R > N_{RThreshold}$ then {Mark all buildings in R as belonging to $D_C$}
    Else {drop R}
```

Based on buildRegionAroundEntity $\underline{(\ )}$ we can easily construct all regions R_C satisfying criterion C, that is full domain D_C:

```
BuildDomain(Criterion C, float $F_{CThreshold}$, int $N_{CThreshold}$):
    Loop by all entities Ent in a settlement {
    If Ent is marked as C-TRUE then {
        If Not ($Ent \in D_C$) then {buildRegionAroundEntity(C, Ent, $F_{CThreshold}$,
        $N_{CThreshold}$)}
}}
```

Remarks.
1. Domain may contain holes.
2. Entities can belong to domain D_C despite being C-FALSE.
3. Threshold value $F_{\text{CThreshold}}$ should be sufficiently high to reflect intuitive understanding of a domain as an area where *most* entities satisfy the criterion.
4. The value of $N_{\text{CThreshold}}$ determines the minimal size of region.

References

Allen, P. M., & Sanglier, M. (1979). A dynamic model of growth in a central place system. *Geographical Analysis*, 11(3), 256–272.

Anonymous (2002). Agent-based software. Available from: www.agentbuilder.com/AgentTools.

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

Batty, M., DeSyllas, J., & Duxbury, E. (2002). The discrete dynamics of small-scale spatial events: Agent-based models of mobility in carnivals and street parades. Available from: http://www.casa.ucl.ac.uk/working_papers/paper56.pdf.

Batty, M., Xie, Y. C., & Sun, Z. (1999). Modeling urban dynamics through GIS-based cellular automata. *Computers, Environment and Urban Systems*, 23, 205–233.

Benenson, I. (1998). Multi-agent simulations of residential dynamics in the city. *Computers, Environment and Urban Systems*, 22, 25–42.

Benenson, I. (1999). Modeling population dynamics in the city: From a regional to a multi-agent approach. *Discrete Dynamics in Nature and Society*, 3(2-3), 149–170.

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

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

Besussi, E., Cecchini, A., & Rinaldi, E. (1998). The diffused city of the Italian North–East: Identification of urban dynamics using cellular automata urban models. *Computers, Environment and Urban Systems*, 22(5), 497–523.

Blue, V. J., & Adler, J. L. (2001). Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B—Methodological*, 35, 293–312.

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

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The unified modeling language user guide*. Reading, MA: Addison-Wesley.

Brogan, D. C., & Hodgins, J. K. (1997). Group behaviors for systems with significant dynamics. *Autonomous Robots*, 4, 137–153.

Candau, J., Rasmussen, S., & Clarke, K. C. (2002). A coupled cellular automata model for land use/land cover dynamics. In 4th International Conference on Integrating GIS and Environmental Modeling (GIS/EM4), Banff, Alberta, Canada.

Chowdhury, D., Santen, L., & Schadschneider, A. (2000). Statistical physics of vehicular traffic and some related systems. *Physics Reports*, 329, 199.

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

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

Couclelis, H. (1997). From cellular automata to urban models: New principles for model development and implementation. *Environment and Planning B—Planning & Design*, 24(2), 165–174.

Erickson, B., & Lloyd-Jones, T. (1997). Experiments with settlement aggregation models. *Environment and Planning B—Planning & Design*, 24(6), 903–928.

Ferber, J. (1999). *Multi-agent systems: An introduction to distributed artificial intelligence*. Harlow, UK: Addison-Wesley.

Flache, A., & Hegselmann, R. (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*, 4(4), <http://www.soc.surrey.ac.uk/JASSS/4/4/6.html>.

Galton, A. (2001). Space, time, and the representation of geographical reality. *Topoi*, 20, 173–187.

Galton, A. (2003). A generalized topological view of motion in discrete space. *Theoretical Computer Science*, 305, 111–134.

Gamma, E. R., Johnson, H. R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Boston, MA: Addison-Wesley.

Ginot, V., Le Page, C., & Souissi, S. (2002). A multi-agents architecture to enhance end-user individual-based modeling. *Ecological Modelling*, 157(1), 23–41.

Gipps, P. G., & Marksjo, B. (1985). A micro-simulation model for pedestrian flows. *Mathematics and Computers in Simulation*, 27, 95–105.

Gruer, P., Hilaire, V., Koukam, A., & Cetnarowicz, K. (2002). A formal framework for multi-agent systems analysis and design. *Expert Systems with Applications*, 23, 349–355.

Gulyas, L., Kozsik, T., & Corliss, J. B. (1999). The multi-agent modelling language and the model design interface. *Journal of Artificial Societies and Social Simulation*, 2(3).

Hidas, P. (2002). Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research Part C*, 10, 351–371.

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

Krumpus, M. A. (2001). Overview of the Evo Artificial Life Framework, Omicron Group. Available from: http://omicrongroup.org/evo/overview/html/overview.html.

Lee, D. B. (1973). A requiem for large scale modeling. *Journal Of The American Institute of Planners*, 39(3), 163–178.

Lee, D. B. (1994). Retrospective on large-scale urban models. *Journal of the American Planning Association*, 60(1), 35–40.

Levine, H., & Rappel, W. J. (2001). Self organization in systems of self-propelled particles. *Physical Review E*, 63, 208–211.

Lorek, H., & Sonnenschein, M. (1999). Modelling and simulation software to support individual-based ecological modelling. *Ecological Modelling*, 115, 199–216.

Miller, H. J. (1999). Measuring space-time accessibility benefits within transportation networks: Basic theory and computation procedures. *Geographical Analysis*, 31(2), 187–213.

Nagel, K., Rickert, M., Simon, P. M., & Pieck, M. (2000). The dynamics of iterated transportation simulations. Available from: http://www.santafe.edu/sfi/publications/Working-Papers/00-02-012.pdf.

Nagel, K., & Schreckenberg, M. (1992). Cellular automaton model for freeway traffic. *Journal of Physique I (Paris)*, 2, 2221–2229.

Noble, J. (2000). Basic relationship patterns. In N. Harrison, B. Foote, & H. Rohnert (Eds.), *Pattern languages of program design* 4 (pp. 73–89). Addison-Wesley.

Noth, M., Borning, A., & Waddell, P. (2003). An extensible, modular architecture for simulating urban development, transportation, and environmental impacts. *Computers, Environment and Urban Systems*, 27, 181–203.

Peckham, J., MacKellar, B., & Doherty, M. (1995). Data models for extensible support of explicit relationships in design databases. *VLDB Journal*, 4, 157–191.

Phipps, A. G. (1988). Rational versus heuristic decision-making during residential search. *Geographical Analysis*, 20(3), 231–248.

Phipps, M. (1989). Dynamic behavior of cellular automata under the constraint of neighborhood coherence. *Geographical Analysis*, 21, 197–215.

Polhill, J. G., Gotts, N. M., & Law, A. N. R. (2001). Imitative versus non-imitative strategies in a land use simulation. *Cybernetics and Systems*, 32(1–2), 285–307.

Portugali, J. (2000). *Self-organization and the city*. Berlin: Springer.

Portugali, J. (2004). Toward a cognitive approach to urban dynamics. Environment and Planning B, forthcoming.

Portugali, J., & Benenson, I. (1995). Artificial planning experience my means of a heuristic sell-space model: simulating international migration in the urban process. *Environment and Planning B*, 27, 1647–1665.

Portugali, J., & Benenson, I. (1997). Human agents between local and global forces in a self-organizing city. In F. Schweitzer (Ed.), *Self-organization of complex structures: From individual to collective dynamics* (pp. 537–546). London: Gordon and Breach.

Portugali, J., Benenson, I., & Omer, I. (1994). Sociospatial residential dynamics—Stability and instability within a self-organizing city. *Geographical Analysis*, 26(4), 321–340.

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

RePast (2003). RePast 2.0. (Software), Social Science Research Computing Program. University of Chicago, Chicago.

Reynolds, C. (1987). Flocks, birds, and schools: a distributed behavioral model. *Computer Graphics*, 21, 25–34.

Schadschneider, A. (2000). Statistical physics of traffic flow. *Physica A*, 285, 101–120.

Schumacher, M. (2001). *Objective coordination in multi-agent system engineering*. Berlin: Springer.

Semboloni, F. (2000). The growth of an urban cluster into a dynamic self-modifying spatial pattern. *Environment and Planning B—Planning & Design*, 27(4), 549–564.

Tobler, W. (1979). Cellular geography. In S. Gale & G. Ollson (Eds.), *Philosophy in geography* (pp. 379–386). Dordrecht: Kluwer Academic Publishers.

Toner, J., & Tu, Y. (1998). Flocks, herds, and schools: A quantitative theory of flocking. *Physical Review E*, 58, 4828–4858.

Torrens, P. M. (2000). How cellular models of urban systems work, CASA Working Paper 28. Available from: http://www.casa.ucl.ac.uk/working_papers.htm, CASA, UCL, 2000.

Torrens, P. M., & O'Sullivan, D. (2001). Cellular automata and urban simulation: Where do we go from here? *Environment and Planning B—Planning & Design*, 28(2), 163–168.

Turner, A., & Penn, A. (2002). Encoding natural movement as an agent-based system: An investigation into human pedestrian behaviour in the built environment. *Environment and Planning B—Planning & Design*, 473–490.

Vicsek, T., Czirok, A., Ben Jacob, E., Cohen, I., & Schochet, O. (1995). Novel type of phase transitions in a system of self-driven particles. *Physical Review Letters*, 75, 1226–1229.

Wagner, D. F. (1997). Cellular automaton and geographic information systems. *Environment and Planning B—Planning & Design*, 24(1), 219–234.

Weifeng, F., Lizhong, Y., & Weicheng, F. (2003). Simulation of bi-direction pedestrian movement using a cellular automata model. *Physica A*, 321, 633–640.

White, R., & Engelen, G. (1993). Cellular-automata and fractal urban form—A cellular modeling approach to the evolution of urban land-use patterns. *Environment and Planning A*, 25(8), 1175–1199.

White, R., & Engelen, G. (1997). Cellular automata as the basis of integrated dynamic regional modelling. *Environment and Planning B—Planning & Design*, 24(2), 235–246.

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

Wolf, D. E. (1999). Cellular automata for traffic simulations. *Physica A*, 263, 438–451.

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. New York: Academic Press.