OBEUS: Object-Based Environment for Urban Simulations

Itzhak Benenson, Shai Aronovich, Saar Noam
Department of Geography and Human Environment,
University Tel Aviv, 69978, Tel-Aviv, Ramat-Aviv
bennya@post.tau.ac.il

Background

City is a complex of objects of different nature, spatial extension and hierarchical rank. Land parcels, buildings, street segments, firms, public institutions, and householders can be considered as "atomic" objects, while social and economic groups, neighborhoods, and the city as a whole can be considered as ensembles of atomic objects, and their properties are mostly determined by the properties of their compounds. Objects behave, that is emerge, interact, change their properties and location, and vanish in time, and the overall dynamics of the urban system is an outcome of objects' simultaneous behavior.

What make City different from the other Multi-Object Systems (MOS) are the *inherently spatial* relationships between the objects. For example, an important component of landowners' estimate of real estate value is the potential for development of the estate's neighborhood. Consequently, a landowner's decision to change the usage of a parcel strongly depends on the land uses of the neighboring parcels. By the same token, householder migration and decisions depend not only on the properties of the real estate and the family's economical abilities, but also on the properties of the physical and human environments at current and potential locations. For both landowner and householder the neighborhood in vicinity of the location and (quite vaguely defined) wider neighborhoods are important for making commercial or residential decisions.

It is not surprising, then, that the generalized Cellular Automata (CA) models, which are based on the neighborhood relationships, provide realistic approximation of urban land use dynamics and most of the recent developments in urban modeling concern changes in land use and have been implemented within this framework (White, Engelen, Uljee, 1997; Torrens, 2000). The existing generalizations concern the lows of automation, but usually do not alter the spatial representation of CA as the square grid of cells and consequent uniform definition of the neighborhood. CA framework

easily adapts the object-based view on urban infrastructure. All we have to do is to define explicitly the neighborhood relations for each infrastructure object. These relationships can be easily determined within standard vector-GIS framework; we can do that based on polygon coverage of land parcels, Voronoi tesselation of urban land constructed on the base of building centroids or graph of the urban road network (Halls et al, 2001; Benenson, Omer, Hatna, 2001). This extension, however, does not call off the principal limitation - the *immobility* of infrastructure elementary units - which prevents the adequate representation of city *social components* by means of CA framework.

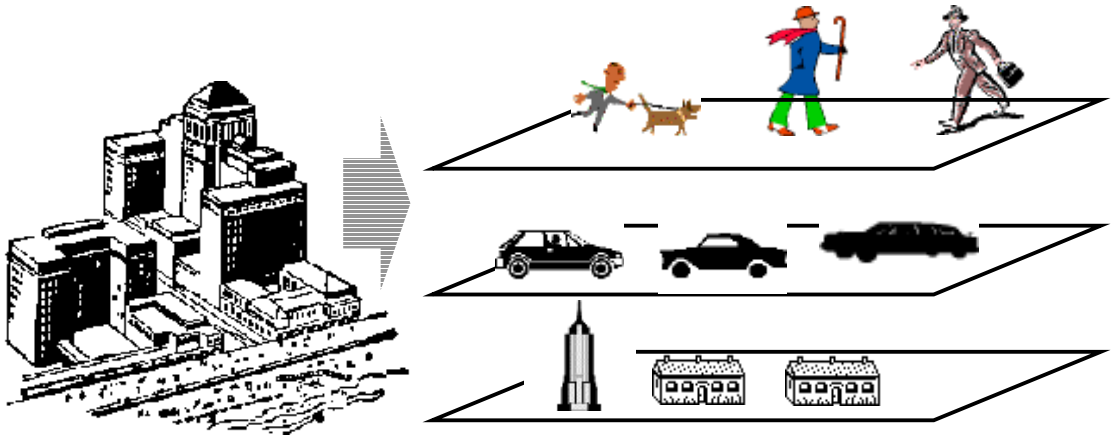
Urban social environments are created by *mobile* humans agents and institutions. The behavior of social objects depends on the properties of neighbors and neighborhoods just as CA models suppose, but, as opposed to land parcels, the standard reaction of human agents to unsatisfactory neighborhoods is a *decision to migrate*. However, in spite of the long history of urban migration studies, object-based modeling of the urban social milieu is much less developed relative to CA-modeling of land usage; intensive modeling of Multi-Agents Systems (MAS) in sociology concentrates on non-spatial relationships, and almost fully ignores the spatial aspect of urban life.

The necessary next step in urban modeling is the simultaneous modeling of infrastructure and human compounds of the city, merely because they develop simultaneously in reality. Multiple existing agent-based modeling environments (www.agentbuilder.com/AgentTools) do not support this intention, maybe because each software is apparently built to fit to the specific domains of applications the developer has in mind and regarding urban dynamics we don't have such. This paper presents, thus, first steps of the developing of the Object-Based Environment for Urban Simulations (OBEUS).

The first and foremost goal of OBEUS is to directly incorporate spatial relations into the urban modeling scheme and to provide the modeler with the handy tools for simultaneous processing of immobile and mobile agents. One can say that OBEUS simply merges cellular-automata and multi-agent simulation approaches. At the same time, at the current stage of object-based urban modeling, as no commonly accepted formalization exists, the modeling environment should remain as open as possible to

empower maximal freedom in formulating qualitatively different model rules and relationships. That is why OBEUS is a “research-oriented” environment, such as the currently developing MUML and EVO modeling systems (www.syslab.ceu.hu/maml/; omicrongroup.org/evo/), which allow the user to formulate and code the laws of his/her specific model (and leaves him with the responsibility to do that properly).

In order to construct OBEUS, we must represent urban space conceptually. Our basic representation of the urban environment as a set of layers is given in Fig. 1, which generalizes our earlier two-layer view on the urban structure (Portugali, Benenson, Omer, 1997; Portugali 2000)



The goals of the development of OBEUS are as follows:

- To simultaneously process immobile and mobile urban objects
- To account for the spatial relationships between urban objects of all types
- To provide tools for the representation and modeling of
  1) The establishment, changes and destruction of infrastructure objects
  2) Migration and adaptation of social objects.
  3) The emergence and disappearance of spatial objects at higher levels of urban hierarchy

Below we present the main features of OBEUS. Examples of formulating specific models within OBEUS will be presented in the lecture.

The representation of urban spatial structure in OBEUS

Spatial relationships between urban objects are represented in OBEUS according to the following principles:

1. Atomic objects can be mobile (*agents*), immobile (*estates*), and non-located (e. g. landlords, currently not implemented).
2. Objects and spatial relations between these objects are represented according to the concept of Entity-Relationship Model (ERM) of the database theory.

- Spatially located objects having the same properties and, thus, belonging to a certain *class* of objects, are simultaneously considered as GIS-type *layer*.
- The ERM table of (many-to-many) spatial relationships between objects of two layers is divided into "pieces" associated with each one of the objects. Each objects contains the list of the objects of the other layer it is related to and the attributes of this relationship. In this way each relationship is stored twice.
- The ERM table of (one-to-many) neighborhood relationships between objects of the same layer is divided into pieces in the same manner; each object contains list of neighbors and attributes of this relationship.

3. Each mobile object is located in relation to (by pointing to) an immobile object.
4. Non-located objects are related to (by pointing to) a list of immobile objects.

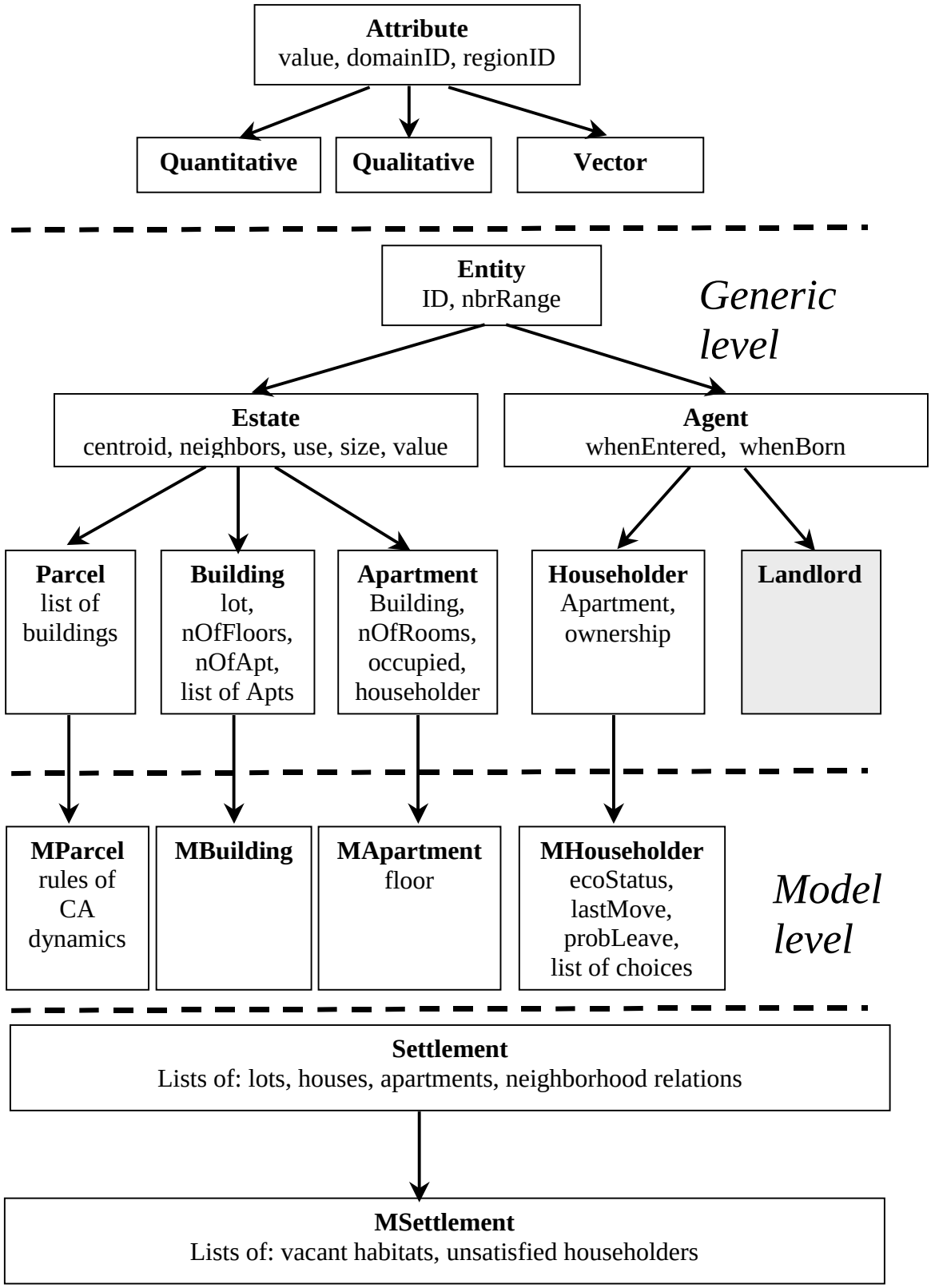
As a result, spatial relationships of atomic object are encapsulated within the object itself. Relations between immobile objects determine relations between mobile and non-located objects. OBEUS does not contain built-in tools for constructing spatial relationships. Tables of relationships between objects of different layers and between an object and its neighbors are constructed in advance, based on the spatial structure of the city and initiated during object construction.

In addition to atomic spatial objects we allow for one more kind of spatial unit, which we call ***Domain***. Intuitively, domain represents areas of "poor" or "rich" population, "industrial" or "commercial" areas, and so on. Below we discuss the concept of domain in details.

Implementation

The three abstract base classes of OBEUS are (Fig. 2)

- ***Entity***, representing elementary objects;
- ***Attribute***, representing entity's traits and;
- ***Settlement***, singleton representing city as a whole


Attribute
value, domainID, regionID
Quantitative
Qualitative
Vector
Entity
ID, nbrRange
Generic level
Estate
centroid, neighbors, use, size, value
Agent
whenEntered, whenBorn
Parcel
list of buildings
Building
lot, nOfFloors, nOfApt, list of Apts
Apartment
Building, nOfRooms, occupied, householder
Householder
Apartment, ownership
Landlord
MParcel
rules of CA dynamics
MBuilding
MApartment
floor
MHouseholder
ecoStatus, lastMove, probLeave, list of choices
Model level
Settlement
Lists of: lots, houses, apartments, neighborhood relations
MSettlement
Lists of: vacant habitats, unsatisfied householders

The next level represents common classes of urban objects, which are supposed to be independent of the user's model. We consider here an example of the "City of Householders", where four classes are sufficient: one of agents (***Householder)*** and three of infrastructure objects (***Parcel***, ***Building*** and ***Apartment)***. The objects of these four classes hold inherent *properties* and *spatial relationships* to both the objects of the same and other classes. Their interfaces are aimed at estimating the state of the neighborhood.

The model-specific classes ***MSettlement***, ***MHouseholder, MParcel, MBuilding*** and ***MApartment*** inherit corresponding common classes, adding user model properties and methods.

Some notes regarding the root classes

1. Components of the vector attribute must be of the same scalar type in order to permit comparison between different components of the vector.
2. The properties of the attribute object represent its value and the parameters necessary for its updating (say, rate of change or growth)
3. ***Settlement*** contains global characteristics of the city, aimed mainly at increasing model performance. For example, the "City of Householders" contains lists of all apartments, a list of vacant apartments, and a list of parks and other public areas.

Domains as emerging spatial units

The ***Domain*** is aimed at representing emerging urban spatial units, which satisfy, according to a given attribute, user-defined criteria - areas of "poor" population, for example. Domains are not considered to be self-existing objects; each entity refers to the zero or more domains it belongs to. A domain is not necessarily a continuous area, and we call its continuous parts ***regions***. Intuitively, the region is an area where "most" of the objects satisfy criteria.

For example, the criterion of expensive building might be that "the price of more than half of the apartments in a building is above 3000$ per m^2"; each house then may be labeled as expensive or not. On the base of this criterion we can define "a domain of expensive dwellings", which can consist of several regions.

A domain is defined by pair (attribute, criterion) and regions of the domain built according to the given criterion do not overlap. Different criteria based on the same attribute can determine overlapping domains - of “expensive” and of “non-cheap” dwellings, for example. It is currently assumed in OBEUS that domains have the same meaning for all agents and all agents “recognize” it in the same manner.

To recognize domains, a spatial algorithm that tests continuity of the area covered by immobile objects satisfying given criteria is necessary. Simple algorithms for determining domains and distance between an object and domain are described in Appendix.

Once found, the importance of the domain (say, "expensive area”) is quite clear. The possibility that the prices of parcels or buildings close to this area will also increase is higher compared to that of the standalone expensive building. Following this positive feedback, the “expensive region” will further expand in a case of unsatisfied demand of “rich” householder agents. It is worth noting that regions of the domain or the domain as a whole may vanish if the prices of apartments there decrease for any reason.

Concluding remarks

The maturation of the object-based approach to urban modeling and simulation brings into relief a more general problem, whose solution can be delayed no longer. Namely, with the rising number of published models, our ability to compare them diminishes. Publication does not provide a solution, for its goal is to explain the key model mechanisms, and avoid discussing “unimportant” details; there is no way to publish the code of a simulation program as well. As a result, the reader is never certain of the differences between models, not to mention the nightmare of each simulation research, when researchers concentrate on marginal factors and miss the factors that really influence the result. The solution is quite clear – to make urban simulation models comparable, we must provide the ability to rerun models without support of the author.

Such a possibility is self-evident when urban dynamics is described by a system of equations serving as a shareable reference. One can easily distinguish between the equation-based models by comparing their analytical presentations and parameter values. Unfortunately, the development of urban modeling leads to the conclusion that equations do not work well in the city; and the rise of object-based simulations is the reaction to this weakness. Nevertheless, no matter which modeling tools are used, the demand for a shareable reference is no less strong if we want to advance the issue from the domain of art to the domain of engineering. The OBEUS environment provides a reasonable way to resolve this problem. Its success depends on its flexibility and the only way to test that is to interpret existing and new models within the OBEUS.

References

1. Benenson I, Omer I, Hatna E, 2001, "Entity-Based Modeling of Urban Residential Dynamics The Case of Yaffo, Tel-Aviv", accepted for *Environment and Planning B*
2. Halls P J, Bulling M, White P.C.L., Garland L., Harris S. 2001, "Dirichlet neighbours: revisiting Dirichlet tessellation for neighbourhood analysis", *Computers, Environment and Urban Systems* v. 25 p. 105-117
3. Portugali J, 2000 *Self-Organization and the City* (Springer, Berlin)
4. Portugali J., Benenson I., Omer I. (1997) "Spatial cognitive dissonance and sociospatial emergence in a self-organizing city" *Environment and Planning B* **24** 263-285
5. Torrens P M, 2000 "How cellular models of urban systems work", CASA Working Paper 28, http://www.casa.ucl.ac.uk/working_papers.htm.
6. White R, Engelen G, Uljee I, 1997, "The use of constrained cellular automata for high - resolution modeling of urban land-use patterns" *Environment and Planning B* **24** 323-343

Appendix: Simple algorithm of determining domain and distance to it

To give an example of criterion (C) and corresponding domain (D_C), let us classify a building as expensive if

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

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

The algorithm of constructing domain D_C of "expensive dwellings" 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-region containing C-True building *Ent* is as following

void buildRegionAroundEntity(entity Ent, float $F_{CThreshold}$*, int* $N_{EThreshold}$*)*
Construct empty temporary region R;
Insert Ent into R;
While there are new entities in R {
Loop by entities currEnt recently included into R
Get list $NBRH_R$ *of neighbors of currEnt*
Calculate fraction F_C *of C-TRUE Entities in* $NBRH_R$
If $F_C > F_{CThreshold}$ *then {Include* ***ALL*** *entities from* $NBRH_R$ *in E}*
}
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, we can easily construct the whole domain D_C:

void BuildDomain():
Loop by all entities Ent in a settlement {
If Ent is marked as C-TRUE then {
If Not(Ent $\notin D_C$*) then {buildRegionAroundObject(Ent,* $F_{CThreshold}$*,* $N_{EThreshold}$*)}*
}}

Remarks

1. Domain D_C can consist of more than one region R_C; the region can contain holes.
2. The entity can be included into D_C in spite of being C-FALSE.
3. $F_{CThreshold}$ should be sufficiently high to reflect intuitive understanding of a domain as an area where most of entities satisfy the criterion.
4. The other criteria besides the fraction of C-TRUE entities in $NBRH_R$ can be considered, for example, the fraction of overall area of C-TRUE entities among neighbors.
5. $N_{RThreshold}$ determines the minimal size of domain's regions

If an entity "remembers" its state regarding domains, that is, for example, a building B "remembers" whether it belongs or not to domain of expensive dwellings at the beginning of a current iteration, then we can easily modify the method buildDomain() above to account for the memory (changes marked bold):

void BuildDomain():

Loop by all entities Ent in a settlement {

If Ent is marked as C-TRUE then {

If Not(Ent ∉ D_C) then {buildRegionAroundObject(Ent, $F_{CThreshold}$, $N_{EThreshold}$)}

}

If Ent belongs to currently built D_C then {inc(Ent.Memory)}

Else {dec(Ent.Memory)}

If Ent.Memory > $M_{Threshold}$ then {Ent.belongs = TRUE}

Else { Ent.belongs = FALSE}

}

To determine the distance Dist(Ent, D_C) between an object Ent located at (X_o, Y_o) and the domain D_C, consisting of regions R_{Ci}, of area S_i we do the following:

1. Estimate the geometric center (X_i, Y_i) of each region R_{Ci}
2. Estimate the radius r_i of a circle, which area equals S_i:

$$r_i = \sqrt{(S_i/\pi)}$$

3. Estimate Dist(Ent, D_C) as $Minimum_i${Distance((X_o, Y_o), (X_i, Y_i)) - r_i}, where Distance denotes Euclidian distance between points.