Environment and Planning B: Planning and Design **2001, volume 28(4) July, pages 481 - 498**

Emergent Design: a crosscutting research program and design curriculum integrating architecture and artificial intelligence

Peter Testa (1), Una-May O'Reilly (2), Devyn Weiser (3), Ian Ross (3)
School of Architecture and Planning, MIT (1)
Artificial Intelligence Lab, MIT (2) Emergent Design Group, MIT (3)
77 Massachusetts Avenue, N51-325 Cambridge, 02139, USA
URL: mit.edu/edgsrc/www Email: ptesta@mit.edu

ABSTRACT
We describe a design process, Emergent Design, that draws upon techniques and approaches from the disciplines of Computer Science and Artificial Intelligence in addition to Architecture. The process focuses on morphology, emphasizing the emergent and adaptive properties of architectural form and complex organizations. Emergent Design explicitly uses software tools that allow the exploration of locally defined, bottom-up, emergent spatial systems. We describe our Emergent Design software, inspired by concepts from Artificial Life, that is open-source and written in Java. This software is an integral part of a curriculum to teach Emergent Design that has original content and pedagogical aspects.

Introduction
Architecture has predominantly reaped benefits from nascent computational technology in the form of computer-aided design tools. However, the discipline still lacks sufficient tools that actively enrich and extend the design process. The notion that the elements of an architectural scenario (or site) have agency and local interactions determined by both their properties and proximity to other elements is strongly present in architectural design investigations. Elements of design are intended to emerge from consideration of the non-linear, interdependent factors of a scenario and the non-linear process of design. While this ALife perspective is present, to date architects lack adequate tools which would enable them to explore dynamical systems within a comprehensive and rigorous framework. We have coined the term Emergent Design (ED) for an approach to architectural design that emphasizes this (relational) perspective. It is characterized in the following ways:

- Given the complexity of a contemporary design scenario, the numerous situational factors of a scenario must be identified and their inter-relationships must be well understood even though they may not be well defined.

- An effective solution to a complex design scenario is achieved through a non-linear process of bottom-up experimentation involving independent, related or progressive investigations into architectural form and complex organization. This interactive process increasingly builds a complex solution that considers the numerous complicated, interdependent relationships of the scenario.

- A design solution derived in such a bottom-up, investigative style is advantageous because it retains explicability and has the flexibility to be revised in any respect appropriate to a change or new understanding of the design scenario.

- Computer software is an excellent means of performing bottom-up architectural experimentation. Despite a simple specification, a decentralized, emergent software simulation can yield complex behavior, exploit graphics capability to model organization and pattern, and can be written flexibly so that alternatives may be quickly examined.

Essential to our notion of Emergent Design is the integration of an effective and powerful software toolbox from the domain of Artificial Life (ALife) into the process of exploring spatial relationships through consideration of the architectural program and agency of primitive design components. In this paper we present our findings that the concepts of Emergent Design and the design of such a software toolbox provide guidance towards informed and innovative designs.

The paper proceeds as follows: First, we describe Emergent Design and discuss the synergy between Architecture and ALife. Next, we focus on the Emergent Design software process and software toolbox. The toolbox has been used in a MIT School of Architecture graduate design studio. To illustrate its capability and range we next describe in detail three investigations that used the toolbox followed by an appraisal of its performance. Finally we summarize and identify future work.

Emergent Design as a Decentralized Process

Emergent Design brings a synergy of new and existing ideas about design into focus at a time when Computer Science technology and Artificial Intelligence research can support the objectives of Architecture. Emergent Design is not entirely new. Architects have always sought to identify the elements of a problem scenario and understand them syncretically. Emergent Design is unique in exploiting and emphasizing the role of software designed for self-organizing spatial simulations. It is a decentralized style of thinking about problems and a way of evolving solutions from the bottom-up. Emergent Design emphasizes appraising and understanding individual behavior (where an architectural component is endowed with agency) as being influenced by other individuals in the system. The system view also incorporates recognition of levels (Resnick, 1994; Wilensky & Resnick, 1998) and the insight derived from understanding how complex, collective, macroscopic phenomena arise from the simple, local interactions of individuals. Architects continually strive to understand the complexity of a system. The recognition of levels in the system and the phenomena that give rise to the formation of levels provide system level insights that are influential towards arriving at an adaptive design. Examples of such complex adaptive systems can be found in both natural and synthetic environments. These forms may be urban configurations, or spatial and organizational patterns but all are evolved through generative probing and grouping in the space of possibilities.

Emergent Design differs from traditional design approaches that emphasize things in space as fundamental and time as something that happens to them. In Emergent Design things that exist (i.e. rooms, hallways, and buildings) are viewed as secondary to the processes through which they evolve and change in time. In this approach the fundamental things in the environment are the processes. Emergent Design seeks to formulate principles of architecture in this space of processes allowing space and time (Architecture), as we know it to emerge only at a secondary level.

Architecture and Artificial Life

There are numerous concepts in the field of Artificial Life (ALife) (Langton, 1989; Langton et. al., 1992) that are advantageously applicable to an architectural design process which emphasizes system-level and constituent understanding. In ALife every component of a system, including elements of the environment, is conceptualized as being capable of agency. Thus, a component of the system may or may not act. Acting is not necessarily actually undergoing a change of internal or external state. It also encompasses the way in which a component may exert influence on the state of the environment or other components. Agency between components implies that their interactions create a level of dynamics and organization. Organizations that dynamically define themselves on one level can themselves exhibit agency and, thus, a new level of organization can form as a result of the lower level dynamics. Levels are not necessarily hierarchically organized. They may, in fact, be recognizable by a particular perspective from which the entire system is viewed.

Architectural systems have numerous levels and each level is interdependent with others. For example, an office building can have a level in which the components are moveable and semi-fixed relative to movement patterns and more stable space defining elements or infrastructures. Different local organizations (work groups) emerge in response to changing organizational structures and projects. Contemporary non-hierarchical work environments when studied closely may be effectively conceptualized as living systems with numerous levels of interdependent organizations.

As shown by the office building example, in general, architectural systems are very complex. One type of complexity encountered is that small, simple, “local” decisions, when coupled with other similar decisions, have large, complex, global effects on the outcome of a design. For example, a choice about locating a work group solely on one floor has ramifications in the choices of circulation allocation, and work equipment placement. These choices, in turn, influence later choices such as material selection or building infrastructure assignment. ALife studies systems in a manner that highlights and elucidates the consequences of locally defined behavior. ALife models are defined in terms of component agency and decentralized component interactions. The running of an ALife system consists of playing out the local interactions and presenting views of the complex global behavior that emerges. Thus, the

Emergent Design version of an ALife system is one in which the environment and components are architectural and spatial in nature. The "running of the system" consists of designating components, defining their agency in local terms and then tracking the macroscopic outcome of their interaction.
Both Architecture and ALife find it necessary to consider the influence of non-determinism in the outcome of complicated system behavior. ALife simulations can be used by architects to model non-determinism. The systems can be defined so that certain behavior has only a probability of occurring. Then different runs of the system will show the result of such non-determinism. In addition, both Architecture and ALife are very aware of how an outcome is sensitive to details of initial conditions. ALife simulations can be defined with parameterized initial conditions and run with different parameter values for the initial conditions. They allow architects to study the impact of initial conditions.

ALife has a powerful capacity to model spatio-temporal phenomena and to represent these conditions via computer graphics. Visualization conveys actual movement, changes in a modeled environment or, the influence of spatial proximity. For example, a designer may want to investigate the implication of a spatial constraint. How does morphology interact with distribution of program requirements, height restrictions, circulation area, and day lighting? Or how will a limited set of forms interact with a given site boundary? ALife simulations facilitate the interactive investigation of many possible spatial outcomes. While ALife models abstract the physical into visualization, the computer is extremely fast and flexible in exploring and modeling a vast design space. These tools can be employed in parallel with more traditional visualization tools or used directly to produce physical models for evaluation via CAD/CAM technology.

Emergent Design Software Process
Emergent Design is a process that, by high-level description, could theoretically be pursued without the convenience of software. However, convenience aside, ALife software is what truly makes Emergent Design powerful. Emergent Design is innovative and advantageous because it incorporates state-of-the-art software technology plus ALife research concepts into a design process. Without the aid of software and the speed of simulations that it facilitates, one would not be able to achieve a fraction of the depth, breadth and richness of designs.

There are two steps in using the toolbox:

1. Define the goals of an investigation concerning how spatial elements may be configured within a bounded area (i.e. site). Specify initial conditions (e.g. initial quantity of elements, size, scale, conditions of site). Identify the elements and the relationships among them. State how elements influence each other and under what conditions they interact. Both the site and its elements can exhibit agency. This identification forms a functional description of the simulation.

2. Using the existing set of Java class methods in the software toolbox, specialize the tool with software that implements the element and site behavior. Run the simulation from initial conditions and investigate the outcome. Usually a simulation has non-deterministic behavior so outcomes from multiple runs are assessed. Based on the outcome, return to either initiative to improve or refine.

The process is practical and not complicated. In the first step, goals and initial conditions are defined. It encourages thinking of elements and site as "active" or imbued with agency. In fact, this style of thinking is very common. The architect says "this is what happens when *a* comes into contact with *b* or, when *c* comes close enough to *d* or when the corner of any element of class *e* touches the boundary, when the bounded area is full", etc. Unfortunately we have noted that designers stop considering agency when they move from thinking to explicitly designing because they can not actualize behavior or manage to 'play it out' in complex, non-linear circumstances. We have found that using the software toolbox reverses this trend.
The second step involves software programming, customizing the toolbox for a specific investigation, and running and evaluating the simulations. Multiple runs can be gathered and analyzed for general properties or one particular run may be scrutinized because it shows something unanticipated or interesting. It is possible to find that the initial conditions are too constraining or, conversely, too unconstrained. Often, in view of the results, the architect revises the initial conditions to try out different relationships and actions. If the outcomes are satisfactory, they are now used to take the next step in the larger process of solution definition.

Emergent Design Software
The software toolbox is a collection of Java classes that can be used to generate complex spatial organizations that exhibit emergent behavior. It is an open source, continually evolving toolbox for building Architecture-based ALife applications. From the application user's perspective, the application runs in a window in which there are two elements.

1. The display that graphically represents the current state of the simulation or "run".

2. The Graphical User Interface (GUI) that allows the user to interact with the simulation by altering display options, changing simulation behaviors, and otherwise modifying the simulation.

The toolbox, as software (i.e. from a programming perspective), provides a general framework that can be engaged as is, can be supplemented with new classes, or can act as a group of superclasses that will be specialized. The toolbox software (i.e. classes and methods associated with classes) can be conceptually divided into three parts: **foundation**, **specialization,** and **applet**. The purpose of abstracting foundation, specialization and applet software is to enable the implementation of different applications that, despite their differences, still use a base of common software. The software is conveniently described in terms of the objects and their behavior we conceived as integral to any emergent design application. For clarity, we will italicize Java object classes in the descriptions that follow.

The **foundation** software defines classes that implement the environment of the simulation that is termed a *site*. A *site* represents the two dimensional region of architectural inquiry. Any polygon can define a *site's* extent. A *site* has no inherent scale - the user determines the scale. A *site* is defined generally so that it can be conceived in a variety of ways. For example, as a dynamic urban diagram to study traffic flows, or, as a bounded interior space to study time-based interactions among activities.

The *site* can be refined in terms of its sub-parts and in terms of the architectural elements that are imposed on it. We term a sub-part, in general, a *piece* and an architectural element, in general, a *zone*.

A *zone* is an object that is imposed on the *site*, rather than a component of it. The bulk of an application consists of the applet creating *zones* and enabling their interaction among each other (which results in *zone* placement, removal or movement) based on different rules, preferences, and relationships. Some *zones* are placed on the site upon initialization. Others are placed during the run, according to the behavior dictated by the *site*, its *pieces* or other *zones*. A *zone* may move during a run or even be removed from the *site*. A *zone* may be represented by a polygon or a line. A *zone* can represent a table, room, building, farm, city, or whatever else the architect wishes to represent on the *site*. A *zone* can be modified to change its color, translate, scale, and skew it, or change its type. In terms of its display properties, a *zone* can be transparent (either partially or fully), and can lie on top of or below other *zones*.

The *site* consists of *pieces*. As displayed, the *site* is subdivided into an array of uniformly sized rectangles. The dimensions of these rectangles are specified in terms of pixels (the smallest modifiable units on the computer monitor). Thus, the display may consist of a single large *piece*, or as many *pieces* as there are pixels in that area. By choosing the dimensions of a *piece*, the level of granularity for the simulation is chosen. Resolution has implementation consequences. The finer the resolution, the slower the speed of a run (because of display costs). A *piece* has its private state. That is, it can store information about various local properties of the *site*, as well as information about the *zone* superimposed on it.

Spatial organization occurs on the *site* via agency of the *site* and *zones*. A *zone* can move itself on the *site* according to criteria based on local conditions such as *zones* nearby, the properties of a *piece* or conditions of the *site*. For example, all *zones* may be initially randomly placed on the *site* and then queried in random order to choose to move if they wish. The decision of a *zone* to move will depend on its proximity to other *zones* and the properties of the *piece* it sits upon. In a different perspective, but one still generally realizable, the *site* can direct the incremental, successive arrangement or placement of *zones* on its *pieces*. This direction is according to a set of behavioral directives that are likely to take into account emerging, cumulative global conditions (i.e. available free space, current density) as well as local preferences that reflect architectural objectives in terms of relationships.

We have implemented a particular means of property specifications for a *site* that we call a *siteFunction*. A *siteFunction* is a general means of representing various qualities or quantities that change across the *site* and which may be dependent on time. It could, for example, be used to model environmental factors such as light, sound or movement patterns. A *siteFunction* is a function of four variables [x, y, z (3D space which a piece defines), and t (time)]. One can create all sorts of interesting relationships between functions and other functions, as well as between *zones* and *siteFunctions* or vice versa. By creating feedback loops (where the output of, say, a function is used as input to another function, or to a *zone*, whose output could be in turn given as input to the object from which it received input), one can very easily set the stage for emergent behavior between human creations (*zones*) and the environment (*siteFunctions*).
The foundation software is intentionally very flexible, and can be used to model almost anything of a spatial nature. By defining a group of *zones* and *siteFunctions*, and defining ways for them to interact with each other locally, one can set up the circumstances that lead to extremely complex, unexpected global behavior. The toolbox can thus be employed in investigations that strive for aesthetics, that predict development's effect on the environment (and vice versa), or that have other purposes. One could use ideas from ecology, chemistry, physics, and other disciplines to formulate the ways in which interaction can take place in the simulation.

The **specialized or behavioral** software defines aspects that are particular to specific simulations. It facilitates the definition of tools such as models of growth, or different kinds of dynamic relationships between objects in the simulation.

Each software toolbox application is run from its applet. The applet is where the emergent states of the simulation are processed and interpreted. It uses two foundation classes: Display and Site. The Display class implements the graphical display of the site and its updating as the run proceeds and emergent dynamics occur. The perimeter is drawn and then color is used to indicate the properties associated with either pieces or zones of the site.

Case Studies of Emergent Design
We have introduced Emergent Design as a graduate design studio in the MIT School of Architecture. As instructors, we select a thematic project drawn from a real world design scenario that enables students to explore a small number of particular applications of bottom-up, self-organizing or agent-based models that have computational simulation potential. The students themselves extend the initially supplied software toolbox. To assist collaboration, the project is maintained and shared via the World Wide Web.

A project generally involves architectural problems related to the convergence of several activity programs such as new working environments open to continuous reorganization wherein dynamic systems interact, or, to design problems related to serial systems and pattern structures such as housing and community design. Students model and simulate factors such as patterns of use, development over time, and environmental conditions. In each case one objective is to explore how the combinatorial dynamics among simple building blocks can lead to emergent complexity.

In a spring 1999 studio course students proposed new models of housing and community for the rapidly expanding and increasingly diverse population of California’s Central Valley. Projects investigated emergence as a strategy for generating complex evolutionary and adaptive spatial patterns based on new relationships between dwelling, agriculture, and urbanism.

Student teams developed three sites/morphologies: Field, Enclave, and Rhizome. Designs operated at both the community scale (120 dwellings) of pattern formation and detailed design of a prototype dwelling or basic building block. A primary objective was to work toward typological diversity and spatial flexibility planned as combinatorial systems using variable elements.

With respect to the Emergent Design software, each team specified the local spatial relationships and worked to author a simulation that satisfied local constraints and behaved according to local relationships, and by doing so, exhibited coherent emergent global behavior. Emphasis was placed on procedural knowledge and the dynamics of spatial relationships. The design of these simulations forced the students to thoroughly examine their specifications and the consequences of them. The natures of the dynamics of the processes were inseparable from the spatial layout of the architectural programs. The students were not trying to create a "best" answer to the problems posed to them.

Instead, they aimed to explore whole systems of morphology by examining many different simulation runs.

Field Project

The field morphology presented the issue of integrating urban housing with farmland. Overall housing density needed to remain fairly low. The existing houses on the site were widely distributed across it, and the team wanted to add more residences while preserving this spare agricultural aesthetic. They wanted to minimally disrupt the already established feel of the site while allowing it to support a larger population.

Thus, the team's goal was to determine how such new housing could be aggregated. They wanted to create a distributed housing system, one where there was not a clear dense-empty distinction. Furthermore, they would not be satisfied with simplistic regular patterns of housing. In employing an agent-based, decentralized, emergent strategy, they conceptualized a dynamic simulation in which each newly placed house would be endowed with a set of constraints and preferences and the house would behave by changing its location within the site in order to fulfill its preferences. They wanted to appraise the non-uniform housing aggregations that can arise from easily expressed, naturally suggested interaction constraints. Interaction elements included farms, schools, roads, and the site’s zones.

In the first simulation designed by the field team 150 new houses are placed down at random on the site. The area taken up by these new houses is approximately 1% of the total area of the whole site. Once all of the new houses have been placed, they react with one another according to an attractivity rule that the field team wrote for the houses in order to express their preferences regarding proximity to each other. The rule states that 1) a house 3 to 5 units from another moves towards it, 2) a house 1-2 units from another moves away from it, and 3) a larger collection of houses will move according to the same conditions but as an aggregate.

The simulation was intended to run indefinitely until stopped by the designer. Several patterns of behavior emerged from the application of this attractivity rule:

1. Based on initial conditions, the houses would form small groups, and would travel "in formation" much like a flock of birds. That is, although each house in the group would travel relative to the site, the houses would not move relative to each other.

2. Sometimes, a group of houses would find a steady state, and simply oscillate between two (or more, if there was more than two houses in the group) orientations.

3. The various groups of houses would sometimes cross paths when travelling across the site. When this happened, the groups would sometimes merge, and sometimes take houses from each other. These collisions would also often alter the trajectories of the groups involved.

4. Groups tended to expand as the simulation went on. Groups sometimes (but not often) broke apart when they encountered the edge of the site, or interacted with another group.

5. Groups sometimes became “anchored” to existing (not newly placed) housing. This was likely because newly placed houses were attracted to all types of houses, and since existing houses are stationary, the newly placed houses within the influence of existing houses would not move out of range of the existing houses.

The aim of the attractivity rule was to produce housing patterns that preserved a distributed pattern. At the same time the patterns should form micro-communities of small groups of houses. While the houses in a micro-community were expected to be socially grouped, ample space between them was desired. After examining a large number of runs of this simulation, it was apparent that the rule achieved the desired results.

The team's second simulation was a parallel investigation into how new housing could be distributed in the pre-existing farmland site. It explored expressing the proximity preferences with the mechanism of feedback (using a behavior's output as its next input). As a simple experiment, the site was tiled with 144 squares of dimension 10x10 that were each designated as a strong attractor, weak attractor, weak repellor, or strong repellor. These designations were assigned randomly with equiprobability. Houses

occupying approximately 2% of the site were then randomly placed on top of these patches. At a time step, each house would move according to the strength of the attractor and repellor patches it was close to. Attractor patches made houses go to the center of the patch at varying speeds (depending on the intensity of the patch), while repellor patches made houses go radially outward. In addition, each patch would newly update what type of attractor or repellor it was. This update was the crux of the feedback in the system. The new patch designation was determined by the patch's most recent state and by the ratio of the overall house density across the site to the density of houses on the specific patch.

Two instances of feedback were investigated. In one case, if a patch had a density that fell some threshold below average, it became less repulsive or more attractive by one designation; that is, strong repellors became weak repellors, weak repellors became weak attractors, etc. If a patch had a density that was some threshold above average, then the opposite would happen; patches would attract less and repel more.

This case demonstrates negative feedback - this is feedback that lessens the behavior of the simulation. Such simulations can usually run forever, with only minimal periodicity. A relatively uniform density is maintained across all of the patches, due to the feedback mechanism. In this case, extremes are usually successfully avoided. Few patches have very small or large densities, and those that do usually take measures to change this. Also, few patches are strong repellors or attractors; they are usually of the weak kind, since their densities usually do not vary significantly.

In the other case of feedback, the patch's properties were intensified by the feedback (i.e. positively). If a particular patch had a high density of houses on it, it would become more strongly attractive, which would tend to draw more houses to/near the center of it which would in turn make it more attractive. Patches with low densities would become increasingly more repulsive. This case favored extremes. Patches fell into one of two states indefinitely. Either they became strong attractors that captured all of the houses in their vicinity, or strong repellors that were empty.

In the field team's case, the negative feedback simulation proved far more useful than the positive feedback simulation. Negative feedback facilitated the notion of distributed housing, while positive feedback created large empty expanses of farmland intersticed with densely packed regions of housing - contrary to the distributed aesthetic. Although the negative feedback simulation created a distributed housing condition, the condition it created lacked structure; houses did not form coherent groups.

The notion of feedback could have been explored much more extensively. This simulation was more of an exploration into general concepts than a specific investigation. One could vary many of the parameters in the simulation (e.g. the number/size of the patches, how the patches relate to existing structures on the site, how many types of patches there should be, whether there should be a neutral patch that is neither an attractor or repellor, introducing indeterminism into how patches change states or how houses move, setting the initial conditions less equitably, etc.). The notion of feedback could be explored from a viewpoint more central to housing, instead of patches of land by having each house serve as an attractor or repellor, depending on how densely populated its neighborhood is. Nonetheless, with the design and goals the team pursued, the results were satisfactory.

Upon seeing multiple instantiations in the first two investigations, the team decided to refine their goals. The results of the first and second investigations created emergent aggregations that, while non-uniform, were too nondescript. By the aggregations being so amorphous and distributed, they actually disrupted the initial site conditions more than a structured aggregation. This outcome was very much unanticipated. It was only realized after exploring many runs of the simulations. It was decided that the notion of distributed housing should be combined with some greater notion of form and directionality.

Thus, in a final simulation, the team chose to restrict the placement of houses to a set of designated farms. After experimenting with a scheme where the designated farms were chosen at random, the team decided the randomness introduced undesirable isolation. That is, they wanted the designated farms to either touch or only be separated by a road. Furthermore, they wanted the non-developable area carved out by the non-designated farms to be somewhat path-like (as opposed to node-like, or central), in order to architecturally recognize a crop rotation scheme. This was accomplished by choosing a farm at random to be non-developable (in contrast to designated as eligible for housing), then having that farm randomly choose one of its neighbors to be non-developable as well. Control would then be passed to the newly chosen neighbor farm that would repeat the process. In the case

where a non-developable farm was completely surrounded by other non-developable farms or borders, it passed control back to the farm that chose it. Control was recursively passed backwards until one of these non-developable farms had a neighbor that had not been chosen yet. The first farm would be colored the lightest shade of green, with each successive farm being a darker color of green. The color progression was intended to be suggestive of a possible crop rotation scheme. Once the area threshold has been reached for non-developable area (which is somewhere over 50% for the field team), this part of the simulation ends.

In the remaining developable area, houses are placed randomly, ten at a time. **(Fig. 1)** After they are placed down, an attractivity rule is run, the houses' movement is determined, and they move accordingly. In this particular application of the attractivity rule, houses are attracted to schools (which are stationary) and other houses (both stationary existing ones and movable newly placed ones). When newly placed houses have moved, they become stationary. This rule was instated so that the simulation did not achieve a predictable result of all newly placed houses being as close as possible to each other and the schools. Instead, the field team wanted attraction to help achieve something subtler. They wanted houses to gather together in loose communities, and tend to be located near schools, but not always. Thus, each newly placed house was attracted to other houses within a five piece circular radius (unlike the square radius of their first simulation), and also attracted to the school that was closest to it (other schools would have no influence over it). The attraction is inversely proportional to distance, although any type of attraction could be used.

Attraction independent of distance and attraction proportional to distance were also tried, but attraction inversely proportional to distance yielded the best results. With this sort of attraction, zones form very strong local bonds, and are usually not affected by distant zones. This sort of behavior (strong local interaction, weak global interaction) is often a key component in bottom-up simulations that exhibit emergent behavior.

This simulation yielded the most satisfactory results to date. The housing arrangement was relatively unobtrusive and not formless. Still, the housing collections are not predictable regular patterns, but subtle arrangements, subject to the behavior instilled in each of the houses. It was also recognized that the experiment might be extended by using the negative feedback mechanisms explored in the second simulation. Instead of making houses stationary after they moved once, one might elicit more complex behavior from the simulation by allowing newly placed houses to move indefinitely, with a new rule to handle the inevitable dense clustering of houses around each other and schools. Unfortunately, time did not permit this extension to the investigation. The team felt the result they acquired adequately set them up to proceed with other aspects of the project design. **(Fig. 2, 3, 4)**

Enclave Project

The goal of the enclave team was to generate grids of houses with meaningful reference to the site's borders, each other, and the community space between them. Although the form and arrangement of houses could vary quite dramatically throughout the site, the houses on the site obey several adjacency constraints to engender the enclave's team idea of community (i.e. no house is isolated from all other houses, and a group of houses must number 4 or less). These constraints were satisfied by appropriate placement rules. In the case when satisfaction of one constraint violated another (this was unanticipated, and only noticed after the first simulation was built) a special rule was used. The site was given probabilistic behavior (i.e. a non-deterministic algorithm) to find a satisfying outcome by adding or removing houses from specific locations on the grid. Instead of dictating a specific number of houses to be placed in the site, the team simply chose a density preference. Each particular lot on the site has an approximately 70% chance of having a house built on it.

In the team's simulations a bifurcated house is used as the atomic housing unit. The prototypical house is composed of a rectangular body (the high-density residential part), attached to a square smaller body (the more open recreation/agriculture part). All units are oriented parallel to one another in the site. In the last of an evolving series of simulations designed by the enclave team, houses are sequentially placed horizontally, from top to bottom along the site's grid. **(Fig. 5)** They are placed from right to left (east to west). This placement order recognized the existing established town east of the enclave site. All houses on the top row of the site must have their smaller part oriented downward. This establishes another form of directionality for the site, and ensures that these houses' open areas are oriented towards the rest of the community, forming an envelope of sorts along the top border. The remaining placement orientations are determined randomly. **(Fig. 6)**

The team's design also called for a high level of interaction between the user and the simulation. They wanted the user to designate where houses would be placed, while still subject to the aforesaid constraints. They designed a system in which the user could at any time choose what should happen to the current empty lot (i.e. whether it should remain empty, have an upward oriented residence placed in it, or have a downward oriented residence placed in it). This system gives the user the ability to manually design portions of the site, while letting the simulation design other parts. This technique allows the user to narrow the search space of possible arrangements further by dictating the contents of any number of lots, and only allowing the simulation control over lots that the user passed by.

The final results represented the culmination of a variety of experiments with placement rules and site specifications. Most outcomes reflected their general goals. The team selected only a few of the self-organized outcomes to use in subsequent steps in their design process. The group profited from the speed of the computational experiment in terms of generating many outcomes from which to choose from. **(Fig. 7, 8, 9)**

Rhizome Project

The rhizome team sought to design a high density, multi-programmatic building within an existing urban area. This objective raised several issues. 1) How should the volume of the building be apportioned among programs? 2) How should a program be distributed within the building? 3) How should different programs be placed in a relation to each other within the building? 4) How could the multi-programmatic building seem to develop and expand into its functionality?

The rhizome team found that an agent-based concept (where each program exhibited behavior to satisfy its preferences) best expressed how the desired high density, multi-programmatic building could evolve. They required an algorithm that would incrementally fill unoccupied space in a building efficiently and according to the principles of program they deemed important. Their simulations include single dwellings, family dwellings, offices, gathering and public spaces. A preference table was developed to codify desired spatial relations among these programs (where each program had an affinity rating for every other program, including itself). Instead of starting with pre-defined volumes and rearranging them in space to help satisfy their spatial preferences, they began with an empty volume and filled it program by program, allowing successive program volumes to choose their neighbors according to their preferences.

The rhizome team also specified dimensional constraints. First, they specified a bounding volume in which all of the programs must reside. They then specified volume ranges for each of the programs; that is, they dictated how many cubic feet could be dedicated to single dwellings, etc. They also determined volume ranges for instances of those programs (i.e. the height of a single dwelling should be between 8 – 10 feet, the width should be between...). The idea behind providing ranges instead of strict numbers was to enforce general pragmatic constraints, but to carefully not restrict the creativity of the process by enforcing hard numbers (that would be contrived in any case). All programs were to be rectangular 3D volumes. **(Fig. 10)**

The simulation starts at the bottom floor in a randomly chosen place that is adjacent to the border of the bounding volume. The site was an empty lot surrounded by other high-density buildings. They wanted their building to respond to its surroundings, which is why they chose to build it "from the outside in." A program was selected according to some preference. In this design simulation it was random, but could just as easily be set, or chosen according to some distribution. A volume that satisfies the applicable constraints was chosen, and the program built. This new zone was responsible for choosing the next program to be placed. It choose according to its affinity preferences. This simulation used a rank-based system (i.e. always choose the first choice first), but other systems have been explored including a percentage based system in which the chance of the first choice being chosen is dependent upon how strong a choice it is, relative to the other choices. This new program was assigned a permissible volume and placed adjacent to the zone that chose it. The process continued until all of the global minimum volume constraints had been reached.

When a volume had been chosen for a new program, an adjacency to the zone that chose it is picked at random. The new zone could attach one of its corners to any of the choosing zone's corners as well as at intervals 1/3 of the length of a side of the choosing zone. Some zones allowed for stacking; that is, the new zone may be placed directly above the choosing zone. The 1/3 side length granularity was chosen for two reasons: 1) Although the simulation did not place circulation spaces, gaps in between

different zones suggested these space be placed by the user. To enhance communication among residents the rhizome team sought to articulate more unconventional, large circulation spaces. By allowing these 1/3 side length offsets, they increased the likelihood of wider, more social circulation spaces. 2) By imposing this gradation, the user could more clearly see the relationships between adjacent zones. If gradations were allowed to vary on a pixel level, structures would be infinitely harder to visually evaluate. The 1/3 side length gradation ensured all of the gradations would clearly be qualitatively different from each other.

In many cases, the randomly chosen volume would not fit in the space chosen because it would overlap with some other already placed volume. In this case, the volume tried every possible adjacency with the choosing volume until it finds one that it will fit in. The 1/3 sidelight gradation ensured there were only 12 attachment points per zone, no matter what the volume's size. This made the check fast and relatively easy. If the volume could not fit any of the adjacency points, its dimensions reduced by a pre-specified decrement value and tried again. If the volume had reduced to its minimum possible size allowed by the constraints and it still could not fit, then the choosing volume relinquished control to a neighboring volume. In some cases, a program's first choice for adjacency already met the maximum volume constraints. In this case, the program went through its affinity list until finding a program that had not met its maximum volume constraints.

At present, the simulation has no hard structural constraints. That is, the finished structure may have zones on the third floor that do not have zones beneath them. This problem is avoided in several ways: 1) If the void underneath the unsupported zone is small enough, and not isolated from other circulation areas, the volume below an unsupported zone could be interpreted as circulation area. Alternatively, these leftover voids could be interpreted as building maintenance or storage areas. 2) If one makes the minimum volume constraints rather large, then the simulation will have no choice but to completely fill the bounding volume in order to satisfy these constraints. 3) Horizontal adjacency is almost always heavily favored to vertical adjacency for architectural reasons (a same-floor relationship is stronger than a different-floor relationship). This tends to cause the whole of the first floor to be filled up before the second floor is constructed.

Appraisal of Emergent Design Software

The diversity of the three different investigations (Field, Enclave, Rhizome) demonstrated the flexibility of the software. First, on a simulation level the three teams engaged the tool for investigations of different scale. The field team used the software to model housing aggregation at the site level. The enclave team studied the possible patterns given an atomic housing unit and the rhizome team investigated the programmatic layout of a single building.

Second, each team found a unique way of integrating the tool within a larger design process. The field team viewed design (conceptual) space as an open system that was continuously modified. They generated many ideas for simulations to study different aspects of housing aggregation. Some of the simulations evolved more or less independent of each other, while others were deeper inquiries into subjects that previous simulations had only touched upon. The field team modified the simulations very little. Instead of trying to alter parameters to achieve the desired result, they preferred to try completely new approaches. Computation was used in a parallel way, rather than a serial one, with the field team.

The enclave team chose to use the tool on one specific part of their design. Initially, they had a general idea of what behaviors that they wished to model, but not a full specification. After the first instantiation, it became clear they had not properly accounted for all of the sorts of interactions that could take place. Little by little, the enclave team developed a full specification of all of the behaviors. Once they finished implementing their specification, they took the results of their simulation (not a particular spatial arrangement of houses, but rather a set of spatial arrangements that satisfied the constraints of their simulation) and incorporated them with the rest of their design. The Enclave team’s use of the tool was far more serial than Field team. They had a very narrow domain of investigation, and subjected their tool to many stages of refinement and specialization.

The Rhizome team thought extensively about the preferred relationships between the various programmatic elements in their design, and encoded them in a "preference table". The team arrived at an algorithm that would try to maximize the satisfaction of these preferences. The Rhizome team's interaction with the toolbox was the most efficient of the three teams. They had the most concrete idea

of what they wanted and how to get it before the implementation started. This resulted in there being few changes to the software. The complexity and richness of the simulation's results reflected the Rhizome team's forethought into the design of the simulation.

Summary and Conclusions

In this paper we have described Emergent Design, a new approach to architectural design. Emergent Design stresses:

- the investigation of a design problem's elements and their inter-relationships
- a process of solution composition that works bottom-up and is guided by considering the design elements as a collection of interacting agents that give rise to emergent global coherence
- the exploitation of computation in the form of Artificial Life inspired software tools that can explore possible solutions in terms of dynamics, evolution, and the interaction of a collection of processes.

Central to the definition of Emergent Design is the exploitation of computation and computer software to explore design possibilities as dynamic agent-based simulations. We have introduced a curriculum centered on Emergent Design. The curriculum is project and studio based. It stresses interdisciplinary collaboration and a team-oriented focus on a broad yet concrete design project. It allows students to learn about and experience diverse roles within a design team. The course emphasizes building tools and ad-hoc working environments in relation to specific design intentions. This approach is achieved by engaging architecture as a set of dynamic and interacting material processes, and by having students learn the value of tool design. Emergent Design, because of its emphasis on dynamic systems and tool design, has the potential to change the way design in general is taught. One of our intentions is to show that traditional, rigidly defined roles such as "architect" and "programmer" are not as effective and flexible as "designer" roles which are personalized in terms of different types of information, demands or technical skills.

In a broader conclusion Emergent Design has already begun to prove itself as a powerful medium of communication for developing and diffusing transdisciplinary concepts. Computational approaches have long been appreciated in physics and in the last twenty years have played an ever-increasing role in chemistry and biology. In our opinion, they are just coming into their own in Architecture. Organization of architectural systems has reached an unparalleled level of complexity and detail. Modeling of architectural systems is evolving into an important adjunct of experimental design work. The revolution in computer technology enables complex simulations that were impossible to implement even a decade ago. Effective use of this technology requires substantial understanding of complex systems throughout all stages of the simulation process. (Kollman, Levin et. al., 1996). The relationship between simulation, mathematics and design ties architecture to more universal theories of dynamical systems. This larger theoretical framework and the techniques of investigation developed in Emergent Design provide a common framework for students in architecture, engineering and management but also researchers and students in disciplines of fundamental importance to the future of architecture including mathematics, computer science, artificial intelligence, and the life sciences. Emergent Design provides an interdisciplinary platform and a new model of design that has the potential to radically transform design education and practice.

Acknowledgments

We thank Professor Terry Knight for reviewing an earlier draft of this paper and the students of MIT Course 4.144, spring 1999.

References

Bonabeau, E.W, 1997, "From Classical Models of Morphogenesis to Agent-Based Models of Pattern Formation", *Artificial Life 3*: 199-211.

Bonabeau, E. W. and G. Theraulaz, 1991, "Why Do We Need Artificial Life?" in *Artificial Life, An Overview*, C.G. Langton ed. (MIT Press: Cambridge, MA).

Goldberg, D.E, 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley: Reading, MA).

Holland, J.H, 1975, *Adaptation in Natural and Artificial Systems* (University of Michigan Press: Ann Arbor, USA).

Kollman, P., and Levin, S., eds., 1996, "Modeling of Biological Systems, A Workshop at the National Science Foundation March 14 and 15, 1996" (National Science Foundation: Washington, D.C.).

Langton, C.G. ed., 1989, *Artificial Life, Santa Fe Institute Studies in the Sciences of Complexity Proceedings*, Vol. VI (Addison-Wesley: Reading, MA).

Langton, C.G., Taylor, C., Farmer, J.D., and S. Rasmussen, eds., 1991, *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity, Proceedings*, Vol. 10 (Addison-Wesley: Reading, MA).

Resnick, M, 1994, "Learning About Life", *Artificial Life Journal*, Vol. 1, No. 1, 2: 229-241.

Testa, P., and O'Reilly, U.M, 1999, "Emergent Design Studio", *Media and Design Process, ACADIA Annual Meeting Proceedings* (ACADIA: Philadelphia, PA): 338-339.

Testa, P., O'Reilly, U.M., Kangas, M., Kilian, A, 2000, "MoSS: Morphogenetic Surface Structure - A Software Tool for Design Exploration", *Proceedings of Greenwich 2000: Digital Creativity Symposium* (University of Greenwich: London): 71-80.

Wilensky, U. and Resnick, M, 1998, "Thinking in Levels: A Dynamic Systems Perspective to Making Sense of the World", *Journal of Science Education and Technology*, Vol. 8, No. 2.