# A GENERIC HOUSE DESIGN SYSTEM BASED ON MULTI-AGENTS:

*Eexpertise of Generating Architectural Plan*

LIBIAO
*School of Architecture, Southeast University, Nanjing, China*
*Jz_studio@126.com*

**Abstract.** This paper presents the process and decision of producing software named "*Gen_house*" that generates high quality sketches of architectural design tasks. The result of a successful project combining research, development and education in both Europe and Asia is achieved in order to ease the practice demand of considering multiple aspects within a design process. The software employs principles and methods of self- organization, agent based solutions and natural sciences, which brings them to the field of architectural design.

## Introduction

Employing principle of "*multi-agents*", which hint a strategy of producing design systems of a more "*event-driven*" nature rather than a "*rule-driven*" one, the author developed a dynamic system named "*Gen_house*" for housing design. Over the past years (even decades), there have been several attempts to do this, and using several different approaches, such as "*shape grammars*", "*genetic algorithms*", "*constraint propagation*" and etc. However, the impact on practice of these paradigms has been low so far. Latent reasons behind of such a low impact lies on the lack of computer implementations. Particularly, the expectation of grammar formalisms or similar rule-based systems to pervade design software mostly remained only an illusion.

"*Gen_house*", applied "*Natural Language*" for the processing, focused on technical difficulties faced on its development. "*Natural Language*" designs and builds the software, which analyze, understand, and generate languages that humans use naturally, so that eventually we could address our computer as though we were addressing another person. Principles and formal similarities can be found in nature. The field described in terms of "bionic" is scientifically discussing the logics of nature's design and its potential technical application for human artefacts. Some of the logics found in bionic can be reproduced in software by applying algorithms. Such algorithms can be found as well in the contemporary calculation of membrane structures as done e.g. by Ove Arup, Partners or the Institute for Lightweight structures at the University Stuttgart, Chair of CAAD at the Swiss Federal Institute of

Technology in Zurich and any other professional institutions which is in field of generative design all over the world.

## Intentions of the "Gen_house": Fundamentals and State-of-the-Art

### 2.1. "TOPOLOGICAL-DIAGRAM" AND ARCHITECTURAL FUNCTION

Architectural function, which is an essential subject, plays an important role during architectural design. Different architectural types hold different functional relationships; nevertheless, it could be described by "topological-diagram" (Fig. 1 left), which expresses the relationship of all the rooms or architectural function district without the information of their area. Base on "topological-diagram", varies of different architectural layouts and architectural spaces could be designed by architects.

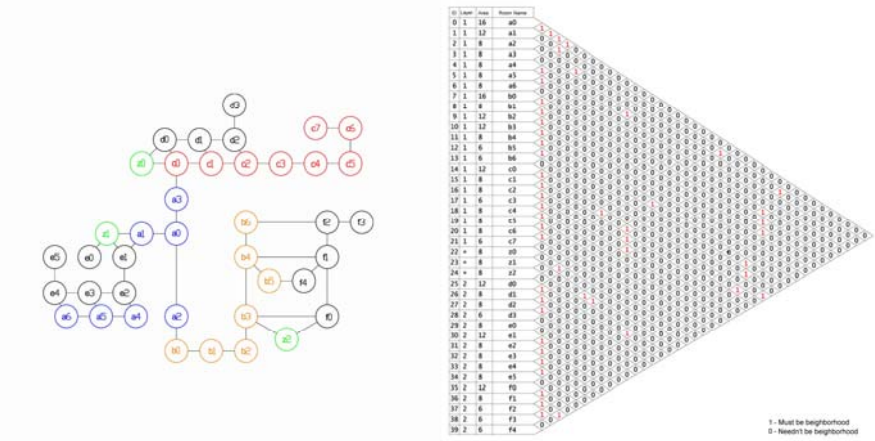

*Figure 1.* Architecture topological relation (left) and its adjacency matrix (right)

"Topological-diagram" integrating behavior psychology, architectural requirements and other professional fields, such as civil engineering, in order to hold balance of architectural function design. During the traditional architectural activities, architects can organize architectural functional relationship through their working experience or references, and then certain outcomes based on the area-needed of each room, environment context, energy saving and etc. will be found. It will take a long period of time for architects to implement architectural layouts of a complicated "topological-diagram". Multiple architectural plans could be generated based on one topological-diagram.

### 2.2 INTENTION OF "GEN_HOUSE"

The spatial object and their relation are described by topological- diagrams. Moreover, the properties of rooms' areas and their topological relations can be shown as the adjacent matrix of agents (Fig. 1 right). Based on the above points, the aim of "Gen_house" is to optimizing the "problem solving" process in housing design. Furthermore, it is considered to have a high value for future development of housing design. Applying the tool of our software,

architects can obtain varies of outcomes according to a “topological-diagram” within a little period of time(See Fig. 2).

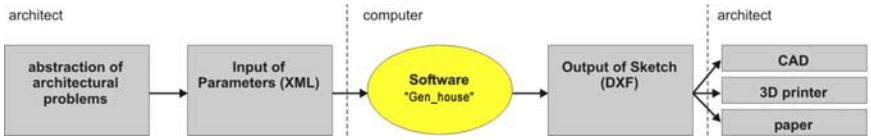


*Figure 2.* Flowchart of the working process.

The relationship between each architectural unit in the “topological-diagram” such as “servant room– kitchen” in a villa or “guard room–cashbox” in bank could be symbolized as abstract nodes as “a” and “b”. Meanwhile, attaching information of their area to turn “a” and “b” into agents, which perform as instances of class in computer program. The desired states described by “topological- diagram” as desired goals of architects can be implemented by “*Gen_house*”. Undoubtedly, XML grammar could represent the adjacency matrix structure according to the nodes of rooms and their edges, but how to find the method to implement the adjacency matrix and present it as a dynamic process?

## Researching process

According to the principle of “bottom-up”, “*Gen_house*” is to make: 1) every agent with its area properties find its neighbors; 2) every rectangle as an agent supposed to find its neighboring rooms, and move towards them to get a “*Target States1*” or “*Target States2*”.

### 3.1 METHODOLOGY AND ALGORITHM

#### *3.1.1 “Target States1”: neighbor-defined rectangles (agents)*

When one rectangle intersects with its defined neighbours, there are eight possibilities for two rectangles overlapping each other (Fig. 3). For example, if the rectangle_i as an active agent intersects with rectangle_l, the


*Figure 3.* Eight possibilities of intersection

possibilities can be classified from the directions of moving, such as the corner or the width/height rates of their intersection.

Assisting by the classes of “Rectangle”, it is easy to figure out the width and height of the intersectional rectangle, and move to “Target States1”

directly. We defined “Target States1” of neighboring defined agents as follows:

- The moving direction of active rectangle is based on its position related to inactive rectangle and the width/height rate of intersection with it.
- Active rectangle must move orthogonally.
- The two rectangles **must** also be overlapped with each other with a defined thickness (thickness of the wall).

Now the active rectangle will find its suitable moving direction but randomly, and move towards the “Target States1” (See Fig. 4).

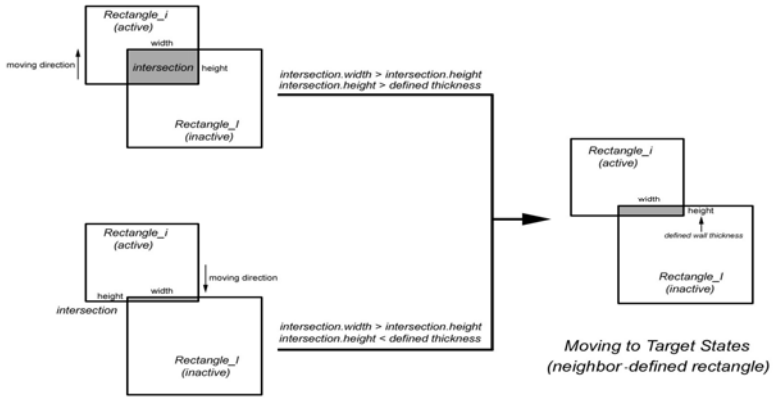


*Figure 4.* Active rectangle moving to its “Target state1”: neighboring defined agents

*3.1.2 “Target States2”: neighbor-undefined rectangles (agents)*

If two rectangles that are not supposed to be neighbours but overlapped during the dynamic process, both of them must move to a “Target States2” immediately as showed in Fig. 5. Their moving directions are similar to neighbour- defined rectangles, and they move orthogonally according to the width/height rate of intersection rectangles. Therefore, the rules keep all the agents of rectangles in a distance during their moving process.

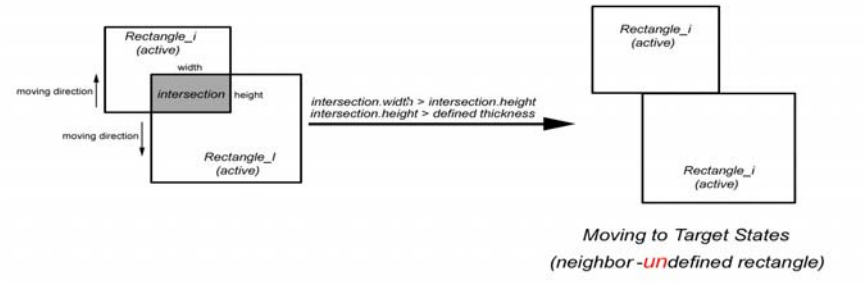


*Figure 5.* Active rectangle moving to its “Target state2”

*3.1.3 Repel and attract*

Rules for neighbor defined agents:

- Do not intersect: Rectangle_i move to rectangle_l — attract;
- If they Intersect: Moving directly to “Target States1” — repel without attract.

Rules for neighbour undefined agents:

- Do not intersect: “return”, and do nothing — without repel or attract;
- If they intersect: Moving directly to “Target States2” — repel without attract.

Repel and attract supply a useful method for the whole procedure, not only making the neighbor-defined rectangles find their adjacent rectangles, but also keeping the neighbor-undefined rectangles don’t intersect each other. All the rectangles could find their suitable positions by employing the different rules for neighbor-defined and neighbor-undefined agents.

## 3.2 THREE PHASES OF THE SOFTWARE

We employed a dynamic processing for every agent till it gets its desired area set beforehand. Nevertheless, some concepts and methods, such as “Target States”, “moveToNext ()” was applied in the experiment. A fictional architect has to design an apartment house; he/she looks for the inner and external parameters such as room- size, dependencies. External parameters are: orientation to sun, climate, location, culture and etc. Then the architect types the values of the parameters into the “my_function.xml” file, which is the first contact with the software our program. The structure of the software is achieved by three phases presented as follows:

### *3.2.1 Phase1: All the agents initialize a global position*

Dots delegating as agents of rooms or building districts emerge in a defined scope randomly; a global position relationship of the agents is generated by employing “attract and repel” which is similar to string in physics. There are two different relations for every two agents– neighbor defined and undefined, and two different distances are defined according to needs of the relations: The agents of neighbor- defined must keep their distance within a range (such as 50-200 mm); on the other hand, neighbor-undefined agents must keep a farther distance (such as 250 mm or more). As a result of this global position achieved, during phase 2 of expanding all agents, the neighbor-defined agents will intersect each other earlier than neighbor- undefined ones. The process of phase1 is presents by Fig. 6-A.


*Figure 6.* Three phases of the program

### *3.2.2 Phase2: All the agents expanding base on global position of phase1*

For each room, a figure of desiredArea divides tempArea is simulated to *pressure* inside (desiredArea/ tempArea); it makes them trend to expanding till they obtain their desiredArea. The four walls of the room are examined in turn, if the room intersects with other rooms, the decision of whether to grow or shrink a room by moving the sharing wall is based on the different

*pressure* inside the rooms. If the *pressure* from inside is less than the *pressure* from outside, no expansion takes place; on the contrary, the wall of the room will expand by moving the sharing wall. By the means, the wall of the agents expanding and detecting whether intersects with other rectangles: moving the wall or not according to the pressure inside, till the pressure inside all room equal "1", which means that every room obtain its desired area. The process of phase2 is presents by Fig. 6-B.

*3.2.3 Phase3: Optimizing the eventual shape and neighboring relationship*
There are two processes for optimizing the result of phase2. Firstly, the program optimizes to make "*patch area*" less, "*patch area*": circumambient area of all rectangles minus area of all rectangles, which is also included in all the gaps among rectangles (Fig. 7 *patch area*: gray color district). Backup


*Figure 7.* "Patch area" (Gary colour)

all the rectangles; changing the length of copied rectangle's width or height randomly in turn; meanwhile, calculating the global "patch area". If getting more "patch area", the program returns and nothing is done, otherwise, setting position of the rectangle to the copied the one. Second, during phase2 the neighbor-defined agents may be detached because of their interaction, therefore, all the rectangles must detect whether they intersect their neighbor-defined rooms, if not, move to them until they intersect. The processes of optimization insure the relationship of "topological-diagram". Phase3 is presents by Fig. 6-C.

## Testing the program

After achieving the program, we tested "*Gen_house*" in some kinds of architectural types, such as small house as villas, banks and others. The program showed a run time suitable for real-time applications; it had capacity to generate many different results within a short time. The procedural method for generating man-made artifacts dramatically reduces the need for human labor in the field of early stage of architectural planning design.

A topological-diagram for small villas was set, which provided 11 rooms, and transferred the topological-diagram into the free grammar of "my_function.xml". The software will store various data into program, such as the area and its neighboring agents of each room, whether there must be a shared door between each two rooms and the ratio of "width/height". By this means, they can be called during its runtime. Choosing three results from

multi- overcomes; the program takes an ulterior step for architectural plan (Fig 8: up results).

In average, “Gen_house” can generate one result of 11 agents in approximate 1 minute. Providing by the software, architects could easily develop their plan in detail. (Fig. 8: below results, transformed the datum to CAD. Any details could be changed properly.)

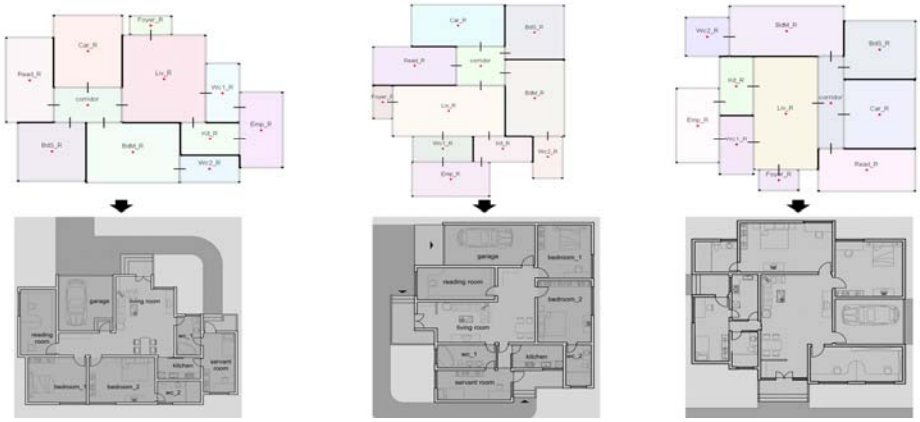

*Figure 8.* Three results

**Further Work of the “Gen_house”**

“*Gen_house*” allows output towards commonly known CAD-software such as AutoCAD, Microsftation, etc. because it exports to DXF-file. Therefore a partly finished plan - or in other terms: a high-quality-sketch is done by “*Gen_house*”. This usually takes days - something that the software “*Gen_house*” can do within minutes.

We implemented the algorithm by platform of ActionScript2.0 and hence are able to run on almost any platform, such as Apple, Windows, Unix due to its origin from “Macromedia Flash”. The software itself is a collection of separated applications of each fulfilling a specific task. But, on the other hand, when testing the program, we find ActionScript could only achieve a restricted numbers of agents, which influences its efficiency and further research of “Gen_house”. Most of architectural types will involve Java or C++ provided with more efficient should apply in the forthcoming research steps; new version of “*Gen_house*” is now developing under platform of Java. Secondly, we presented a procedural algorithm to generate plausible interior spatial configurations. Architects must pay attention to any other architectural essential factors, such as orientation to sun, distance to border, climate, location, wind, culture, etc which are defined as external parameters. Our method does focus more on context of external elements. Furthermore, a procedural method for modeling man-made artifacts for reduce the need of human labor in field of 3D model design is considered in next step.

## References

B Medjdoub, B Yannou: 2000, *Separating Topology and Geometry in Space Planning,* Computer-Aided Design, Volume 32, 39-61.

B Medjdoub, B Yannou: 1998, *A topological enumeration heuristics in a constraint-based space layout planning,* Artificial Intelligence in Design '98

Bruno Feijo, Paulo c. Rodacki Gomes, Joao Bento, Sergio Scheer, Renato Cerqueira, PUC-Rio: Distributed agents supporting event-driven design processes, Artificial Intelligence in Design '98, 557-577.

Jess Martin: Procedural House Generation: 2006, *A method for dynamically generating floor plans.* http://wwwx.cs.unc.edu/~eve/papers/EVEAuthored/2006-I3D-Martin.pdf, 2006

LiBiao and Schoch Odilo: 2005, *Computer Aided Housing Generation with Customized Generic Software Tools,* Proceedings of fifth China urban housing conference, Hong Kong, PP723-728.

Paul S. Coates AA Dipl.: 2004, *some experiments using agent modelling at CECA.* GA2004 - 7th GENERATIVE ART CONFERENCE/ EXHIBITION/ PERFORMANCES, Milano Italy. http://www.generativeart.com/papersGA2004/22.htm.

Sotirios Kotsopoulos, Haldane Liew: 2004, *A Studio Exercise in Rule Based Composition,* First International Conference on Design Computing and Cognition.