An adaptive approach to domestic design

Bruno POSTLE
Free Software developer
bruno@postle.net

ABSTRACT

This paper discusses the need for an adaptive and iterative process for the design of ordinary domestic buildings that meet human needs. The approach described here is to use Pattern Languages as fitness criteria to guide an iterative design optimisation. To complement this, a Form Language for domestic buildings is proposed that is physically buildable, sufficiently flexible, and suitable for evolution through mutation and crossover. This evolutionary method has been implemented as Free Software, and the results are comparable with historical informally constructed buildings.

Keywords: housing, adaptive architecture, evolution, genetic programming, informal design, Pattern Languages
INTRODUCTION

Designing a house is a simple matter: we find a ‘designer’ who can assess the requirements, then mix some knowledge with a bit of creativity, and wait for them to come up with a ‘solution’. This is so straightforward and obvious; the same process applies for practically all the ‘things’ that people make — but why then is our built environment such a relentless disaster, with nothing but suburban sprawl on the one hand and high-rise stacks on the other?

You could argue that these building types bring the highest financial return, and that our cities are just a mirror of our political economy. But just maybe there is also some flaw in the way buildings are designed?

This paper documents an ambitious project to find a new way to design a measured, humane and beautiful environment, and to make this method available as a set of tools that any of us can use.

INFORMAL URBANISM

To start with there is a simple idea — it has long been noted (Jacobs, 1961) that among other things, certain features of historical urbanism, such as narrow streets, short blocks, and high densities, are very well suited to pedestrian movement — when buildings are closer together, there are consequently more destinations within a walkable distance from any one point, and that in turn a pedestrian oriented city has many advantages over a city where a car is necessary for all activities.

However even where this understanding of urbanism is uncontroversial, most approaches to city ‘design’ favour a rectangular grid of streets and buildings. Rectangular grid urbanism has some big advantages: firstly it is a great administrative convenience, lengths and areas are easy to calculate, counting plots of land, and street lights, and everything else in-between is more practical with a grid; secondly, identical repeating plots of land allow the reuse of generic modular designs.

Contrast with an informal city layout that follows the landscape with no overall master-plan, this would be spectacularly more complex to design and construct — every building would need to fit a completely different environment, essentially each building would have to be designed independently — a task that is intellectually demanding and time-consuming to say the least.

So, rectangular grids populated with generic modular designs are very scalable and practical to master-plan — but how do you impose an ‘ideal’ rectangular grid on a city when real landscapes have hills and streams, existing buildings and roads, odd property boundaries, interesting views, delightful places, vegetation and wildlife, and any number of other things that can and ought to be preserved or enhanced?

There is some precedent, many historical cities and towns are ‘informally planned’, or rather they were not apparently ‘planned’ at all, they look like no effective central authority placed the streets and buildings ‘rationally’. The pattern of development in these places appears to be such that each building and street happened ad hoc to suit the day-to-day needs of its citizens, using resources to hand, and responding to just their immediate environment in terms of available water, drainage, light, security and access.
Such historical cities typically have streets in a web-like network that features a larger proportion of ‘T’ junctions rather than crossroad junctions, multiple alternative routes, irregular shaped plots, and building construction right up to the street edge. Informal street layouts apparently closely follow topography, such as hills and streams, they follow previous structures and property lines, they follow existing vegetation, and they follow the routes that people want to use.

This paper is not going to provide recipes for creating or managing such informal layouts, but it is tackling the other half of the problem — given that we have an informal street, block and plot arrangement as is the case in most cities around the world, and that we are unable to use a modular repetitive design approach — how do we go about populating this layout with buildings?

THE DESIGN PROBLEM

Automating the design of buildings seems an impossible task given that it takes people years to become proficient, and even then many just repeat a small repertoire of successful design strategies.

Repeating previous practice has a real disadvantage; it requires that each new task has to be similar to a previously successful task. The temptation here, when faced with a complex difficult problem, is to attempt to reduce it into a something more familiar. This approach suits large developments where the site can be treated as a blank slate and populated with repeating units of known value — i.e. there is a constant pressure to change the problem to suit the solution, to aggregate small plots so they can be rationally subdivided, to find straight lines to string a series of identical units, or to leave areas of empty space between regular shaped buildings and an irregular boundary.

For a design process that can suit a wide range of sites and produce genuinely responsive solutions to each project, the most obvious and demonstrably successful is to codify the design process into a series of rules. José P. Duarte shows that it is possible to codify the work of the architect Alvaro Siza Vieira using a decision tree for the logic and shape grammars for the geometrical representation (Duarte, 1999).

Again there is a difficulty with this approach — all possible eventualities need to be considered in these rules, it requires a master architect such as Vieira and a constrained problem-space, such as limitations on the geometry of building plots, and a relatively small number of possible permutations for any one site.

The approach we are taking here is different; the idea is to take clues from the same processes that form buildings in historical cities and towns: i.e. rather than developing the design in a single step, start with a simple building and modify it iteratively using small incremental improvements, i.e. a generative process (Mehaffy, 2008).

Such an approach needs two things: a geometric description of the building that is suitable for incremental adaptation, it also needs a method for evaluating models constructed with this description.

Christopher Alexander and his collaborators introduced the concept of Pattern Languages for the built environment in A Pattern Language (Alexander, Ishikawa, & Silverstein, 1977). This Pattern Language can be best described as a series of rules for guiding decision making, each
Pattern describes an archetypal situation or problem and is accompanied by a solution. The Pattern Language doesn’t necessarily provide a working method for us to use these rules for actual designing, but provides a very effective system for evaluating of any part of the built environment — in machine optimisation terms the Pattern Language provides ‘fitness criteria’.

**EVOlUtion**

The rest of this paper describes a process that ‘designs’ buildings in a context of the site, the local topology, street frontage, and the daylight field of the surrounding buildings — ultimately, although not necessarily, this process will also ‘design’ buildings for us as the individuals who will inhabit them.

There are two good reasons why we would ‘design’ buildings to fit the context of adjacent buildings. One is obvious, we would like better buildings, and the neighbouring buildings determine how we can do this — neighbours block sunlight or overlook for example, and we need to respond to this in the placement of our rooms and windows.

The other not-so-obvious reason is that the relationship between neighbouring buildings can, and should, be deeply intertwined — for this intertwining we need co-evolution:

A patch of urban fabric develops over a timescale of centuries with individual buildings being replaced, added or extended one at a time; each building is inevitably formed in the context of the existing neighbouring buildings and responds to them. Over generations, every building is recreated perhaps multiple times, but the incremental change of the urban fabric as a whole is more modest as time goes on. The result, if the number of generations is sufficiently large, approaches a convergence where any two selected buildings can be said to be symbiotic, literally ‘living together’.

This is a simplified history of the development of an informally planned settlement; the process takes hundreds of years and, we believe, accounts for the unique evolved character of many historical cities. However it implies a steady-state or slowly changing society; it doesn’t help us in a world where cities are growing at an unprecedented pace. With 65 million more city dwellers each year and the total urban population expected to increase by 2.6 billion between 2011 and 2050 (United Nations, 2011), there is no opportunity for modern cities to evolve in the historical manner — unless the lifetime of buildings is reduced such that they are replaced at a much higher rate.

An alternative is simulation: by designing buildings in a virtual environment, repeatedly tearing them down and rebuilding, the process can be freed from the cost constraints of physically having to construct anything. However, there is still a significant design cost — would anyone be prepared to produce the vast number of designs necessary given that only a small percentage would ever be constructed?

What is needed is a machine that can design high-quality buildings that meet human need, in context, consistently, and unattended. With such a machine, we can achieve the ability to extend and create modern towns and cities with all the positive attributes of historical informal urbanism.

As a solution this paper presents the outline of a method for designing individual buildings that is both adaptive to context and fully scalable. This method goes to the heart of the failure...
of the modern construction industry to match the quality of historical cities as places to live, despite all the obvious failings of these historical cities in terms of inadequate services and crumbling technology.

FORM LANGUAGES

Alexander, and the physicist Nikos Salingaros (Salingaros, 2006) have noted that Pattern Languages need complementary Form Languages. A Form Language is a building vernacular, a construction and planning method that is flexible enough to implement a Pattern Language and adaptable enough to allow a building to evolve to suit new requirements in the future.

So Alexander’s Pattern Language is ideal as governing fitness criteria in our machine. What we then need is a Form Language that has a geometry that can be described mathematically, but which is also adaptable enough for an adaptive design method.

What follows is a brief description of a Form Language that is fully adaptive and adaptable, it has some advantages: individual buildings can be summarised mathematically in a compact form; it isn’t based on a grid, so sites and buildings are not required to be right-angled; it creates geometries that make some kind of structural sense, so it describes buildings that are buildable — finally, an indication that it is on the right track is that it is possible to use this Form Language to describe the layout of many historical vernacular buildings.

A basic Form Language for domestic design

This Form Language exists at the level of layout planning. Following is a brief description of its fundamental properties:

QUADRILATERAL SPACES The plan form of habitable spaces

The basic unit of the form is the quadrilateral, e.g. triangular and circular spaces are not supported, but L-shaped and T-shaped spaces can be assembled from multiple quadrilaterals. There is no requirement that corners are right-angled, so a rectangle, trapezium, or parallelogram can be used as a space.

An important attribute of a quadrilateral is that it can be hierarchically divided and subdivided into further quadrilaterals.

STRAIGHT WALLS Boundaries between spaces

A quadrilateral implies straight-ish edges. In this adaptive method we need to be able to re-combine two adjacent quadrilaterals to produce a single quadrilateral, so walls that run through a junction implicitly need to be continuous.

Another way of looking at this is, that for a fully adaptable building it should be necessary to demolish and reposition walls; if this results in a kink in the remaining wall then the Form Language is not fully adaptable.

These two ‘forms’ together are analogous to Alexander’s Pattern 191 THE SHAPE OF INDOOR SPACE (Alexander, Ishikawa, & Silverstein, 1977):
With occasional exceptions, make each indoor space or each position of a space, a rough rectangle, with roughly straight walls, near right angles in the corners, and a roughly symmetrical vault over each room.

(Note that the remainder of this paper refers to many further Patterns from Alexander; these will be indicated in the form ‘123 PATTERN NAME’).

Fig. 1 Graph of four rooms with crossed wall junctions.

Fig. 2 Graph of four rooms with T-shaped wall junctions.

Fig. 3 Schematic plan of Ardéchoise farmhouse showing subdivision.
T-SHAPED WALL JUNCTIONS The arrangement of boundaries

A wall junction that meets with a cross, meaning two walls that intersect and pass through each other, has some real disadvantages. In terms of layout this form is attractive since the regular grid on the plan layout feels ‘rational’, however:

In terms of an adaptive approach to planning, a crossed junction requires that in order to improve just one room by resizing, at least four rooms need to be resized. This isn’t very helpful — it is very unlikely that the incremental value of improving the first room will produce acceptable improvements in the rest of the rooms.

Adaptivity leads to adaptability, T-shaped wall junctions provide more circulation options. Imagine four simple spaces connected with crossed walls. The circulation graph has four nodes and four links; whereas four spaces connected with T-shaped wall junctions have a graph with four nodes and five links. Hence an adaptive form-language has no preference for cross-shaped wall junctions.

A Binary Tree

The Form Language described above has an implicit hierarchy — quadrilaterals can be further subdivided into smaller quadrilaterals. A process of division like this leads to an ‘unbalanced binary tree’ structure, which with appropriate parameterisation can describe all possible layouts of this Form Language.

Fig. 4 Binary tree showing subdivision of Ardéchoise farmhouse above, each coloured node represents a habitable space or room
The immediate advantage is that a binary tree is a data structure suitable for evolution through mutation and crossover. With a suitable fitness function that consists of the application of Alexandrian Patterns, we can maintain a population of designs and apply a selection process to drive optimisation.

Multiple levels in the building are implemented as stacked binary trees, although each level can have a different layout, the dimensional geometry itself is inherited from the level below — the result is that many walls and spaces will run through from one level to the next where necessary.

Since the Form Language is inherently buildable, the binary tree also describes construction sequence and structural form.

The principle is simple but also very versatile — the Form Language is not just suitable for a machine-driven adaptive design method, but the resulting plans are also fundamentally adaptable as a consequence. You can look at any of these generated plans and see how the physical building can be further adapted, modified or extended to suit changing needs.

**FITNESS CRITERIA USING PATTERN LANGUAGES**

So as noted, Alexandrian Patterns are not a design method in themselves, in machine optimisation terms the Patterns can be thought of as fitness criteria for evaluation of human needs met (as opposed to monetary value). However, the optimisation process consists of comparing options, and different options can have wildly varying build costs that also effect design fitness — so the fitness function looks like this:

\[
\text{Human Needs Met} = \text{on/bathrooms/outdoor space.}
\]

**WALL LENGTH** Placing freestanding wall segments.  

Wall segments need to be long enough to place a door, also for future adaptation. Therefore try to ensure wall segments are longer than 1.25m.
RIGHT ANGLE CORNERS Placing walls.
Corners of rooms need to be approximately right-angle. Therefore try and make corners 90° ±5.

Future Patterns not yet implemented
The above Patterns are already implemented; the following Patterns are under development (as of November 2013):

SUNNY PLACES Placing indoor and outdoor space for direct sunlight.
Direct sunlight in rooms and outdoor spaces is important (105 SOUTH FACING OUTDOORS, 138 SLEEPING TO THE EAST).
Therefore try and place spaces favourably in an orientated variation of the daylight occlusion field.

NICE VIEWS Placing spaces to take advantage of scenery.
Buildings should be arranged to ensure that distant or beautiful views can be seen (192 WINDOWS OVERLOOKING LIFE).
Therefore try and place spaces favourably in a variation of the occlusion field with a ‘views’ overlay.

HALF HIDDEN OUTDOORS Placing outdoor space for privacy.
People need private outdoor space, but not too private (111 HALF-HIDDEN GARDEN, 163 OUTDOOR ROOM).
Therefore place outdoor spaces favourably in a new field representing ‘overlooked-ness’.

AVOIDING NUISANCE Placing indoor and outdoor space to avoid external disturbance.
Noise from busy streets is annoying and should be avoided. Therefore try and place spaces favourably in a scalar nuisance field that varies depending on street.

Notes on ‘Crinkliness’
Above we introduce a new ‘crinkliness’ quantity, this needs some explanation.
Two associated Alexandrian Patterns are 107 WINGS OF LIGHT and 159 LIGHT ON TWO SIDES OF EVERY ROOM (Alexander, Ishikawa, & Silverstein, 1977). The utility of the Patterns is not in doubt, but the all-or-nothing finality of the light on two sides rule is not well suited to an adaptive design process — it isn’t possible to incrementally change from a room with light on one side to a room with light on two sides.

What is needed is a Pattern that allows a continuum between a small or shallow room with light on one side, a larger room with light on two sides, or with a higher ceiling, and that dissuades deeper rooms that don’t have enough light even then. What works in an adaptive process is to calculate and assess the measure of crinkliness, which can be described as the potential for light:
**Crinkliness** = can share a direct connection in the resulting network graph. A *slide mutation* moves the dividing line (often a wall, although not necessarily) to resize the two child spaces.

A *swap mutation* exchanges the two child quadrilaterals and any of their children.

A *un-divide mutation* removes all children and converts a branch-node into a leaf node, a type is then assigned. This is equivalent to demolishing partitions.

Additional mutations *swap, add or delete* entire floor levels, each represented by a complete binary tree.

**Combination**

In addition to mutation operations we need to take advantage of another evolutionary process — *crossover* or *combination* performs a ‘cut and splice’ operation on any pair of buildings, randomly exchanging rooms, a group of rooms, or even entire levels. Two ‘child’ buildings are then created as a result and added to the population pool.

The following diagrams show some snapshots of the evolution of a small house, it has been limited to a single storey for clarity.

**RESULTS AND DISCUSSION**

The software is in a proof-of-concept form. Even so it is capable of demonstrating the feasibility of the approach. Below are the results of a more complex example of multiple multi-storey buildings interacting with each other as described in the introduction. The geometry you can see is raw and unaltered, it is visualised in perspective using the Blender ambient occlusion renderer without any manual editing. Hence there are no people, textures or any of the usual things you might see in an ‘architectural’ visualisation.

One unexpected, but nonetheless interesting, result is that the lower storeys of the generated buildings tend to have higher ceilings compared to upper storeys; this can be seen in the images below where shorter windows have been assigned to upper floors. This is a pattern that is often seen in historical buildings — but there is nothing in the code specifically to make this happen. There are other similar results; for instance there is no code specifically to make a courtyard as seen in the single storey example above. With evolution it can be very difficult to establish cause and effect. Although in this case ceiling height is closely related to the available light in the rooms via the **CRINKLINESS** Pattern; a higher ceiling means a larger external wall, which in turn means bigger windows and more light. So we can presume that since lower floors receive less light from the sky due to surrounding buildings, this leads to some pressure for higher ceilings — although there is another possible chain of causes: the **OUTDOOR SPACE** Pattern tries to ensure there is some outdoor space on every level, and the cost of building-covered space combined with bad daylight characteristics implies a pressure for buildings to become smaller as they go up; this can be seen as a ‘ziggurat’ effect. So perhaps when we see higher ceilings on lower storeys, this is to counteract the effect of deeper plan layouts. Of course, none of this may be happening, living rooms tend to be concentrated on lower storeys because of the **ACCESS VIA PUBLIC ROOMS** Pattern; they are generally bigger, and it could be that just this process is driving up the height of lower storeys.
Fig. 6 Population during the evolution of a single storey house, after 48 generations. Note that there is a large amount of variation and the layouts are quite simple.

Fig. 7 Population during the evolution of a single storey house, after 640 generations. Note that there is less variation, but the layouts are all more complex.
Fig. 8 Binary tree representations of the population shown above, after 640 generations.

Fig. 9 Undirected adjacency graphs of the population shown above, after 640 generations.
Fig. 10 Plan layout of fittest single storey house after 640 generations. The long number at the bottom is a hash ID that is unique to each possible layout.

Fig. 11 Binary tree of fittest single storey house after 640 generations.
Fig. 12 Undirected adjacency graph of fittest single storey house after 640 generations.

Fig. 13 3D view of fittest single storey house after 640 generations.
An important point that needs to be made is that the evolutionary optimisation process doesn’t arrive at equilibrium. There is no intention to try to make something ‘perfect’ or ‘complete’; the end result is simply a snapshot that has the potential for further growth over time using the same process, but in a physical built form.

The images below are of a small group of seven house plots covering a tileable $1/16$th hectare patch of land; this layout leaves about 30% of the space for a small square and alleys between the blocks. Houses have been iterated over multiple generations using the daylight occlusion field derived from the previous generation. The result is equivalent to a density of 112 houses per hectare, with an average internal floor area of 130m². This doesn’t allow room for much in the way of vehicle traffic, and is at a density where the fitness score for each building has begun to decline, so it isn’t really viable for an extensive part of a town. However, each house has a ground floor entrance, and has usable outdoor space on multiple levels, so it compares favourably to an equivalent development of apartment blocks.

This project could be seen as a partial answer to the challenge made to software developers by Christopher Alexander at the end of the 1996 keynote speech to the Institute of Electrical and Electronics Engineers (Alexander, The Origins of Pattern Theory, the Future of the Theory, And The Generation of a Living World, 1996). It isn’t a full answer to the problem of creating a living built environment, but the potential is there.

Fig. 14 Rendered view of a generated cluster of seven buildings.
Fig. 15 Rendered view of a generated cluster of seven buildings.

Fig. 16 Rendered view of a generated cluster of seven buildings.

Fig. 17 Rendered view of a generated cluster of seven buildings.
Fig. 18 Rendered view of a generated cluster of seven buildings.

Fig. 19 Rendered view of a generated cluster of seven buildings.

Fig. 20 Rendered view of a generated cluster of seven buildings.
Fig. 21 Rendered view of a generated cluster of seven buildings.

Fig. 22 Rendered view of a generated cluster of seven buildings.

Fig. 23 Rendered view of a generated cluster of seven buildings.
REVIEW OF ALTERNATIVE TECHNIQUES

Our approach is in contrast to existing ‘fractal’ city generators. One technique uses **split grammars** (Wonka, Wimmer, Sillion, & Ribarsky, 2003) to generate credible building façades. CityEngine (Müller, Wonka, Haegler, Ulmer, & Van Gool, 2006) uses a **shape grammar** approach to create road layout and building façades that look like real historical cities. Our work approaches the problem from the opposite direction; if by creating buildings from the ground up using an adaptive technique, using validated Patterns as fitness criteria, we produce buildings and cities that resemble historical urbanism — this is incidental rather than the intended result.

Other genetic programming approaches are (Jagielski & Gero, 1997) and (Krämer & Kunze, 2005) who also describe the use of **genetic algorithms**, i.e. mutation and crossover to evolve gridded floor plans. (Martin, 2006) describes a method for generating floor plans.

A non-genetic technique (Merrell, Schkufza, & Koltun, 2010) produces generated layouts and buildings, the results are very impressive but concentrate on reproducing the behaviour of architects in the building design process.

FUTURE WORK

More Patterns are needed in the fitness calculations; these new Patterns are typically of a larger and smaller scale than those already implemented. The external environment needs a richer representation, such as allowing for existing trees and other non-building context. The work with the daylight occlusion field can be extended to direct sunlight, in addition to indirect daylight. It can be extended to deal with ‘overlooking’, and ‘views’, although how we can define a ‘good view’ is not entirely obvious.

Smaller scale Patterns will deal with details such as placement of windows and doors which are currently chosen parametrically rather than optimised.
The fitness framework described here assigns quantitative values to each Pattern, this is a very flexible system, but the numbers chosen are based on experience and guesswork. More research is needed to refine these values and the ranges that are used in the Gaussian scoring system.

Currently 3D geometry is created in DXF, RIB and Collada format, but these are more suitable for visualisation than for construction management, future modules will generate more useful IFC (Industry Foundation Classes) data.

The reader will have noticed that all the examples shown here are of small-scale domestic housing. Further work to apply the same methods to small shops and other domestic-scale mixed-use buildings can be foreseen — but the design of a neighbourhood or larger part of a city with routes and public buildings is out of scope of this project.

REFERENCES


