The Journal of Open Source Software

Homemaker: software for adaptive domestic design

Bruno Postle [1]

1 Independent Researcher, UK

DOI: 10.21105/joss.07711

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @JJ
- @abhishektiwari
- @arfon

Submitted: 28 March 2024
Published: 08 April 2026

Summary

The Homemaker stack of software evolves domestic buildings using a modified version of Christopher Alexander's Pattern Language as fitness criteria. Multiple buildings can be evolved in parallel, buildings with multiple-storeys, non-orthogonal plots, and courtyard layouts are supported. The software shows how evolutionary computation provides a design method complementary to the theory of Pattern Languages, thereby addressing a major criticism of Pattern Languages that they do not offer a practical design method.

Statement of need

The problem: Christopher Alexander's 'A Pattern Language' (Alexander et al., 1977) describes principles for creating humane, liveable built environments and has profoundly influenced fields beyond architecture (notably Software Design Patterns (Postle, 2019)). Despite being a best-seller with many advocates, very few people actually use it to design buildings. Even Alexander acknowledged this, calling it "my biggest failure" (Hopkins, 2010). Pattern Language theory lacks a practical design method—Alexander identified the missing element as iterative adaptation through many small steps, or "unfolding" (Alexander, 1979), but provided no computational framework for implementing this process.

The gap in existing solutions: Commercial generative design tools (e.g., Archistar, Spacemaker AI, TestFit) optimise for financial metrics like return on investment and regulatory compliance, not human-centred design quality. Academic 'AI' architectural design research (Caetano et al., 2020) similarly focuses on improving efficiency of commercial property development. There are apparently no tools that implement Pattern Language theory computationally, meaning the theory cannot be empirically tested—we cannot determine whether buildings designed according to Pattern Language principles would actually be judged viable and desirable by architects and potential occupants.

This software addresses the gap: Homemaker implements evolutionary computation as the iterative design method that Pattern Language theory was missing. The software generates building designs that evolve to fit constrained sites (irregular plots, adjacent structures, access limitations) whilst maximising compliance with configurable architectural patterns from Alexander's work. The fitness function balances usable floor area against pattern-based quality metrics and material costs. This enables researchers to empirically test whether Pattern Language theory is sound by evaluating whether high-scoring evolved designs meet professional architectural standards and human needs.

The software generates validated 3D Building Information Models (IFC format) suitable for actual construction workflows. Current functionality focuses on domestic buildings that maximally occupy their sites—typical of dense urban infill contexts where buildings adapt to constraints analogously to historic compact urbanism. Active development is extending the system to support arbitrary building types by accepting a spatial programme (room types, required areas, and connectivity requirements) as input, enabling the evolution engine to design

buildings beyond the residential domain.

Brief description

The Homemaker system (Postle, 2013) consists of a collection of software components intended to be run unattended. The system evolves multi-storey building designs using a Pattern Language (Alexander et al., 1977) as fitness criteria for selection in a Genetic Algorithm.

Buildings are represented using a novel binary tree data structure (Postle, 2024), with each division holding orientation and ratio numbers indicating a geometric partition of a 'room' into two smaller 'rooms'. The fitness function implementation consists of selected patterns from the Pattern Language assessing the model in the context of its local environment. The fitness calculation ultimately maximises a single value derived from this 'quality', multiplied by floor area, and divided by a simplified cost function representing a physical quantity of building materials.

The presumed construction process of a historic vernacular building is that it starts with a single room and grows by accretion and subdivision, with periodic removal of failed and unwanted parts of the structure. To simulate this, a 'load-bearing wall' form-language is used: buildings are nested agglomerations of rooms; each room is a four-sided quadrilateral (not necessarily orthogonal); walls are vertical; the geometry of storeys below informs the geometry of storeys above. A novel binary tree representation, with each division holding orientation and ratio numbers indicating a geometric partition of a 'room' into two smaller 'rooms', can be used to describe such a building. The graph/tree is recursive, so any subdivision of rooms and storeys can be represented. Each leaf-node in the tree is a room with an interchangeable 'usage' (kitchen, bedroom etc..) that can explored by optimisation. Typical historic vernacular buildings closely fit this model and can usually be represented with such a binary tree.

A specific advantage of a binary tree data structure is that branches can be 'grafted' from one model to another, an analogue of crossover or recombination in sexual reproduction of biological systems - this tree grafting is a standard technique in Genetic Algorithms. The fitness function consists of selected patterns from the Pattern Language assessing the model in the context of its local environment (the daylight field formed by surrounding buildings). The fitness calculation ultimately maximises a single value derived from this 'quality', multiplied by floor area, and divided by a simplified cost function representing a physical quantity of building materials. A population of models, each a variation of the same building, is produced through mutation, crossover and culling. When multiple buildings are evolved in parallel, the daylight field environment is periodically updated based on the geometry of the buildings as they change.

The software stack consists of: Homemaker, a queue manager for evolving multiple buildings in parallel; Urb, the data model for a single building; and Molior, File::IFC and File::DXF, which are used to generate a 3D BIM model on completion of the evolution.

Updates

Recent updates to the software include:

- 'Sahn' Space Type, a courtyard circulation space prevalent in traditional Arab houses (Hakim, 2013)
- A configuration system allows users to tailor pattern fitness parameters and building costs for individual and groups of buildings.
- The Algorithm::Evolutionary (Merelo Guervós et al., 2010) Perl module replaces the previous Makefile-based system for driving evolution in populations.

- The Homemaker queue manager allocates resources for the parallel evolution of multiple buildings.
- Space centrality is calculated using the average path length over the circulation graph to all other spaces.

Challenges

This software is fundamentally non-interactive. User control is limited to the setting initial parameters, though the resulting IFC BIM model is readily editable in Native IFC software such as BonsaiBIM (Moult, 2025). Homemaker uses a binary tree to describe the building, but this is conceptually hard for a human to manipulate directly as it has no visual relation to a building design. The data structure is instructions to build (a genotype), rather than the finished geometry (a phenotype). A separate but related project not described here is the 'Homemaker add-on' (Postle, 2023) that provides an entirely interactive experience based on the same principles, using non-manifold spatial geometry and the Topologic library (Jabi & Chatzivasileiadi, 2021).

Evolutionary design in this manner seems to be unsuitable for placing smaller buildings on larger plots, or larger 'towers in a park' designs, the constraint of occupying the entire space appears to be necessary. For a more suburban typology, pre-prepared standard designs distributed in a 'cookie-cutter' manner would likely achieve the desired result more efficiently. Homemaker is also relentlessly domestic, producing good ordinary buildings, these buildings don't show the hand of an architect, and as such may not be perceived as Architecture.

Comparison with Other Generative Design Approaches

Commercial generative design tools such as Archistar (Archistar, 2025), Spacemaker AI (Autodesk Spacemaker, 2025), or TestFit (TestFit Inc., 2025) primarily optimise for financial metrics like return on investment and regulatory compliance. Based on publicly available documentation, these tools evaluate design success through economic feasibility rather than optimising for human-centred design criteria. Homemaker also differs from notable academic approaches:

Shape Grammar systems (Duarte, 2005; Stiny & Gips, 1972) focus on formal composition rules rather than the human experiential qualities central to Pattern Language. Space Syntax methodologies (Hillier & Hanson, 1984) operate predominantly at the urban scale, analysing city-wide movement patterns and visibility networks. While Homemaker does analyse spatial connectivity, it operates at the building scale and is underpinned by an evolvable binary tree model structure particularly suited to evolutionary processes. Urban procedural tools like CityEngine (CGA Shape Grammar) (Müller et al., 2006) and DeCodingSpaces (Koenig et al., 2018) focus primarily on street networks and block configurations rather than building-level design patterns.

Homemaker's distinctive contribution is its use of evolutionary computation with Pattern Language as fitness criteria. This computational approach makes Alexander's influential but unwieldy theory implementable for actual building design, addressing qualitative aspects of human experience. The binary tree representation of building layouts enables evolutionary processes while preserving the fundamental structure of load-bearing wall architecture. This represents an approach distinct from approaches that prioritise financial optimisation or formal composition.

Acknowledgements

Christopher Alexander (1936-2022) was the inspiration for this work.

References

Alexander, C. (1979). *The timeless way of building*. Oxford University Press. ISBN: 9780195024029

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: Towns, buildings, construction*. Oxford University Press. ISBN: 9780195019193

Archistar. (2025). *Archistar platform*. https://www.archistar.ai

Autodesk Spacemaker. (2025). *Spacemaker: Site planning and analysis*. https://www.autodesk.eu/content/autodesk/global/en/products/spacemaker/overview.html

Caetano, I., Santos, L., & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research*, *9*(2), 287–300. https://doi.org/10.1016/j.foar.2019.12.008

Duarte, J. P. (2005). Towards the mass customization of housing: The grammar of Siza's houses at Malagueira. *Environment and Planning B: Planning and Design*, *32*(3), 347–380. https://doi.org/10.1068/b31124

Hakim, B. S. (2013). *Arabic-Islamic cities building and planning principles*. London Routledge 2013. https://doi.org/10.4324/9780203037874

Hillier, B., & Hanson, J. (1984). *The social logic of space*. Cambridge University Press. https://doi.org/10.1017/CBO9780511597237

Hopkins, R. (2010, December 23). *An interview with Christopher Alexander. Transition culture*. https://www.transitionculture.org/2010/12/23/exclusive-to-transition-culture-an-interview-with-christopher-alexander/

Jabi, W., & Chatzivasileiadi, A. (2021). Topologic: Exploring spatial reasoning through geometry, topology, and semantics. In S. Eloy, D. Leite Viana, F. Morais, & J. Vieira Vaz (Eds.), *Formal methods in architecture* (pp. 277–285). Springer International Publishing. https://doi.org/10.1007/978-3-030-57509-0_25

Koenig, R., Beilik, M., Knecht, K., Abdulmawla, A., & Fuchkina, E. (2018). *New methods for urban analysis and simulation with Grasshopper - using DeCodingSpaces-toolbox*. 65–68. https://doi.org/10.52842/conf.ecaade.2018.1.065

Merelo Guervós, J. J., Castillo, P. A., & Alba, E. (2010). Algorithm::Evolutionary, a flexible Perl module for evolutionary computation. *Soft Computing*, *14*(10), 1091–1109. https://doi.org/10.1007/s00500-009-0504-3

Moult, D. (2025). *Bonsai BIM add-on* (Version v0.8.2). https://bonsaibim.org/

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural modeling of buildings. *ACM SIGGRAPH 2006 Papers*, 614–623. https://doi.org/10.1145/1179352.1141931

Postle, B. (2013). An adaptive approach to domestic design. *Journal of Biourbanism*, *II (1&2/2013)*, 31. https://doi.org/10.6084/M9.FIGSHARE.6267401.V2

Postle, B. (2019). On pattern languages, design patterns and evolution. *New Design Ideas*, *3*(1), 44–52. http://jomardpublishing.com/UploadFiles/Files/journals/NDI/V3N1/PostleB.pdf

Postle, B. (2023). *Homemaker add-on* (Version 2023-12-16). https://github.com/brunopostle/homemaker-addon

Postle, B. (2024). *DOM file format specification*. https://bitbucket.org/brunopostle/homemaker/src/master/DOM_SPEC.md

Stiny, G., & Gips, J. (1972). Shape grammars and the generative specification of painting and

sculpture. *Information Processing*, *71*, 1460–1465.

TestFit Inc. (2025). *TestFit: Site feasibility and building design software*. https://testfit.io