

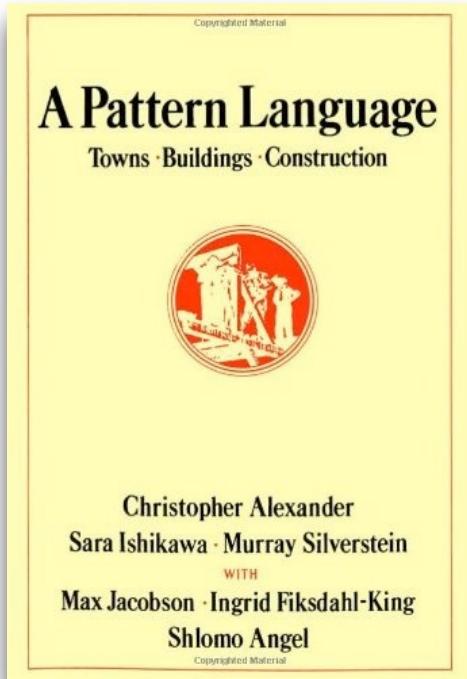
Software Architecture

FEUP-M.EIC-ASSO-2024

Ademar Aguiar, Neil Harrison

Patterns of Software

The Origins of Patterns, 1977



- Christopher Alexander in this book presents a pattern language, an ordered collection of 253 patterns.
- The goal was to enable non-experts to architect and design their own houses and communities with quality.
- Patterns describe current practice and provide vision for the future.
- Patterns integrate knowledge from diverse sources and link theory and practice.
- Patterns have a strong “bottom up” orientation.

<http://c2.com/cgi/wiki?ChristopherAlexander>

What is a pattern?

The elements of this language are entities called patterns. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

from *A Pattern Language*, Alexander et al, 1977

253 patterns

from global to particular problems

- 1. Independent Regions
- 2. The Distribution of Towns
- 16. Web of Public Transportation
- 18. Network of Learning
- 20. Mini-buses
- 43. University as a Marketplace
- 83. Master and Apprentices
- 134. Zen View
- 253. Things from your life

title

context

20 MINI-BUSES*

. . . this pattern helps complete the LOCAL TRANSPORT AREAS (11) and the WEB OF PUBLIC TRANSPORTATION (16). The local transport areas rely heavily on foot traffic, and on bikes and carts and horses. The web of public transportation relies on trains and planes and buses. Both of these patterns need a more flexible kind of public transportation to support them.

problem



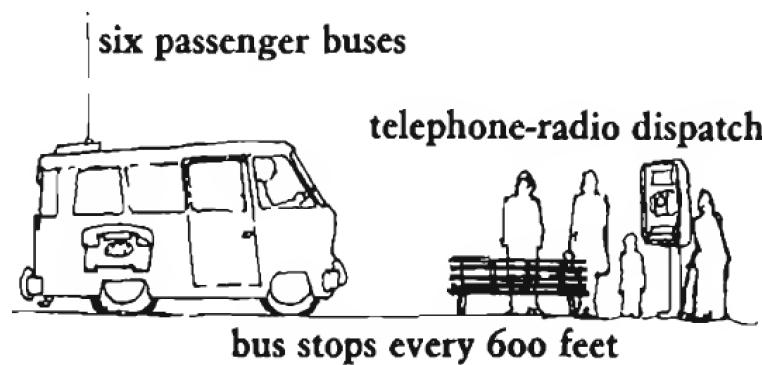
Public transportation must be able to take people from any point to any other point within the metropolitan area.

discussion

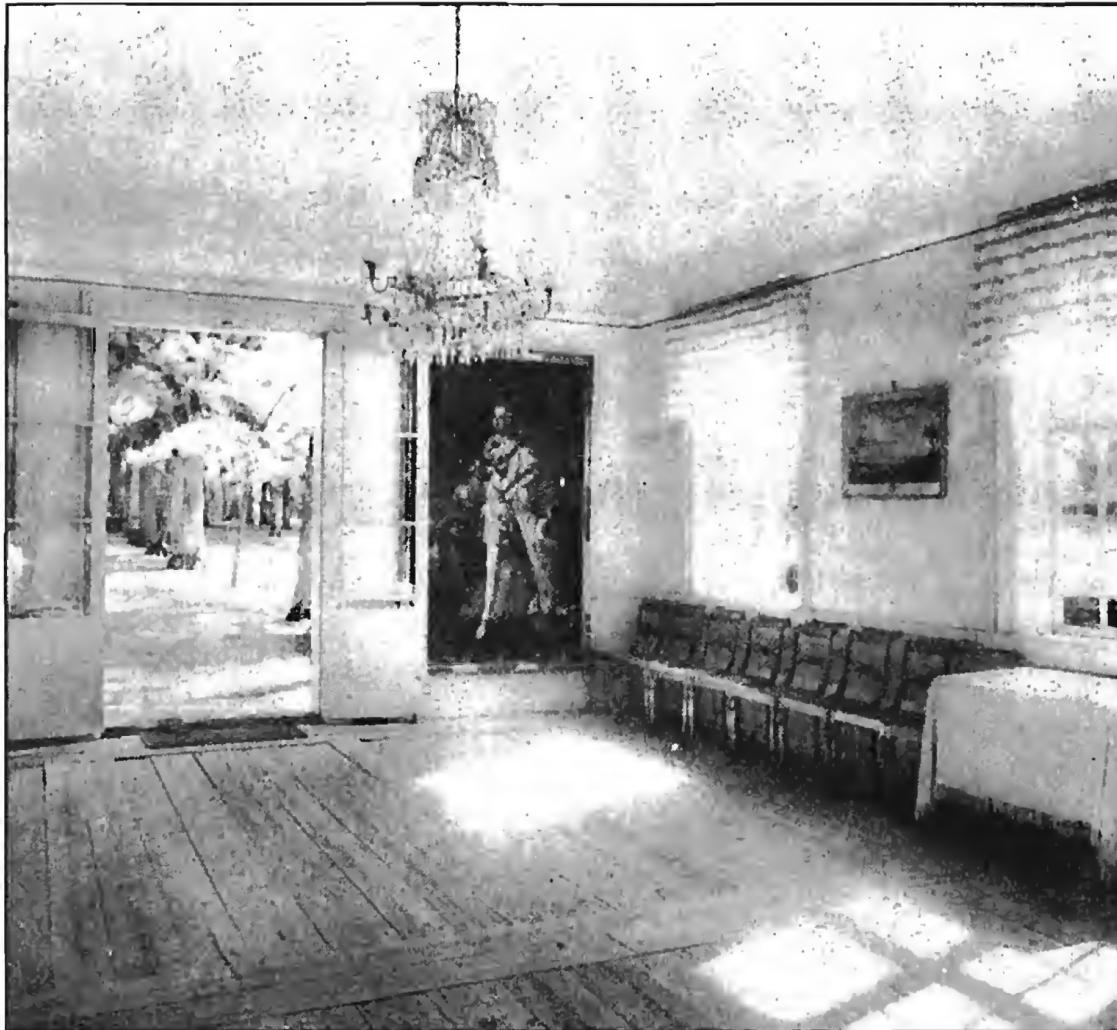
Buses and trains, which run along lines, are too far from most origins and destinations to be useful. Taxis, which can go from point to point, are too expensive.

solution

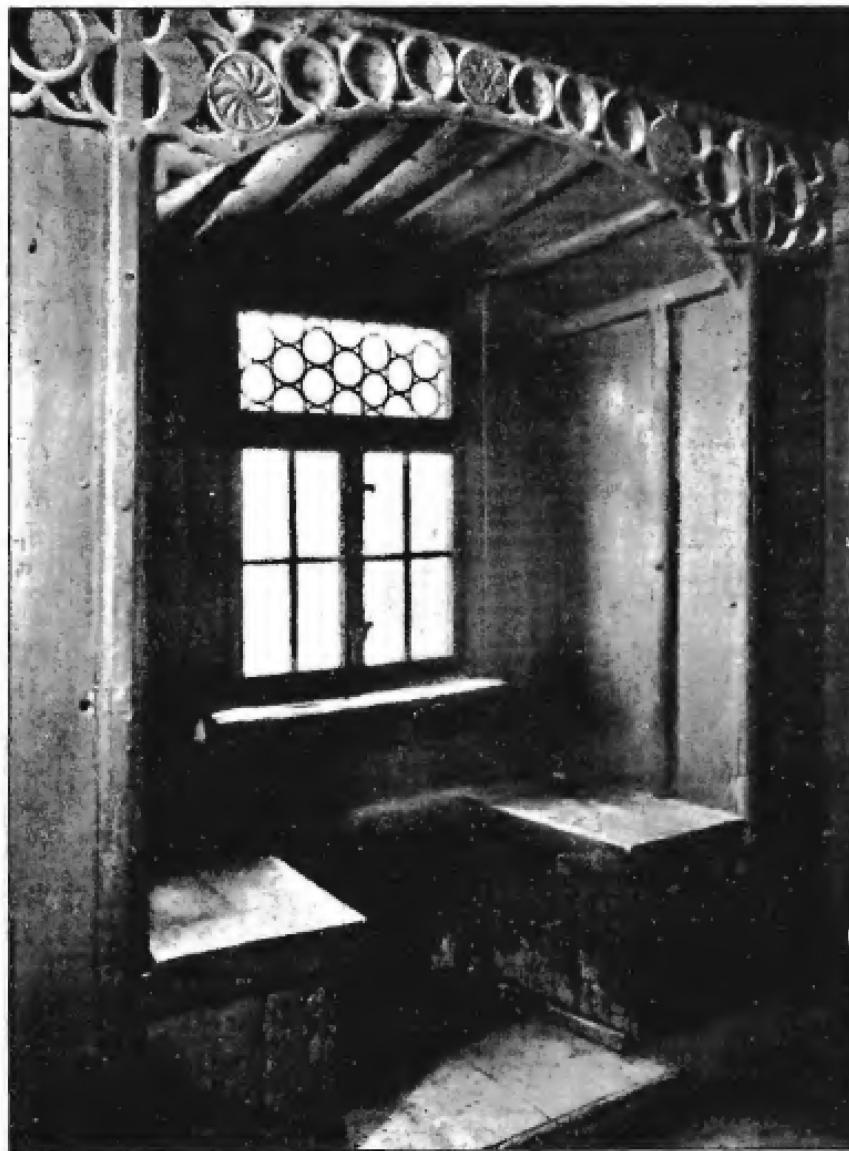
Establish a system of small taxi-like buses, carrying up to six people each, radio-controlled, on call by telephone, able to provide point-to-point service according to the passengers' needs, and supplemented by a computer system which guarantees minimum detours, and minimum waiting times. Make bus stops for the mini-buses every 600 feet in each direction, and equip these bus stops with a phone for dialing a bus.



I 59 LIGHT ON TWO SIDES
OF EVERY ROOM**



I 80 WINDOW PLACE**



251 DIFFERENT CHAIRS



253 THINGS FROM
YOUR LIFE*



QWAN

Quality Without A Name

Fifteen Properties

The degree of life which appears in a thing depends upon the life of its component centers and their density. Following are fifteen structural features which he has identified as appearing again and again in things which have life.

Together, these fifteen properties identify the character of living systems. They are the principal ways in which centers can be strengthened by other centers. They are not independent, but rather rely on and reinforce each other. Things which are more whole, which exhibit more life, will have these fifteen properties to a strong degree. Conversely, the things in this world which are most lifeless will have these properties to the least degree.

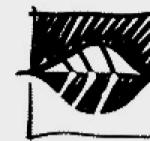
1. Level of Scale



5. Positive Space



9. Contrast



13. The Void



2. Strong Centers



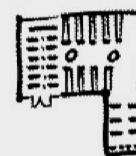
6. Good Shape



10. Graded Variation



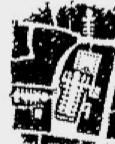
14. Inner Calm



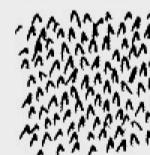
3. Boundaries



7. Local Symmetries



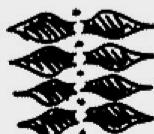
11. Roughness



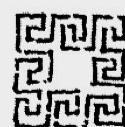
15. Not-Separateness



4. Alternating Repetition



8. Deep Interlock



12. Echoes



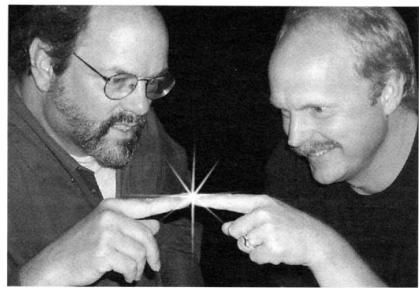
Quality attributes in Software Architecture

Supportability	Modifiability	Scalability	Reusability
Maintainability	Testability	Availability	Security
Interoperability	Usability	Reliability	Performance



<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

Software Patterns, 1987..1994..



A new software patterns discipline started by a group later called “The Hillside Group”, 1993.



Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (aka Gang of Four), published the first book on “Design Patterns”, the “bible”, 1994.



- Kent Beck & Ward Cunningham, “Using Pattern Languages for Object-Oriented Program”, OOPSLA '87, 1987.

<http://wiki.c2.com/?HistoryOfPatterns>

The screenshot shows a web browser window with a title bar "People Projects And Patterns". The main content area features a decorative graphic on the left and the following text:

People Projects And Patterns

"Who -- People, What -- Projects, and How -- Patterns"

People.

People who are important to the practice of Software Development, that is... Not every famous philosopher belongs here; perhaps not even every software developer who has written a book. But if someone has said something important on the topic of Software Development or Patterns, by all means, tell us what important things they said.

On people pages - we describe individuals like Christopher Alexander or Kent Beck. People don't always write their own pages. There are too many noteworthy people to expect that to happen. Likewise, don't take what's there too seriously. If you find something you know to be wrong or inappropriate, take the time to edit it. Be kind and use understanding as some folks are new to this.

- [PeopleIndex](#)

Projects.

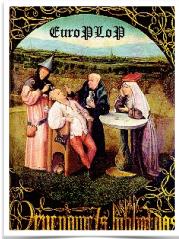
We believe patterns spread from person to person as they work together on projects. Just what are those projects? Look for project pages like [MacApp](#) or [HotDraw](#) or [SmalltalkSummer](#). Don't look for secrets, and don't write any yourself.

- [ProjectIndex](#)

Patterns.

Now these are the real gems. We're looking for that common knowledge that's so uncommon. For example, [CommandObject](#) makes undo and redo easy while [WindowPerTask](#) addresses updating issues in early [ModelViewController](#) (MVC). [ModelRendererView](#) describes a variation on the theme. These pages won't necessarily contain the usual parts of a [WrittenPattern](#). We're just labeling ideas so we can study how they flow.

PLoP Conferences



Friday, Jul 07, 2017

www.hillside.net/conferences

Pattern Languages of Programs (PLoP) Conferences

[Login](#) [Text Size](#)

 THE HILLSIDE GROUP

Hilltop Books Contact Conferences Patterns Vision Wiki

MAIN MENU

- ▶ Hilltop
- ▶ Books
- ▶ Contact
- ▶ **Conferences**
 - GuruPLoP
 - PLoP
 - Chili PLoP
 - EuroPLoP
 - Meta PLoP
 - Asian PLoP
 - ParaPLoP
 - Scrum PLoP
 - Sugarloaf PLoP
 - Viking PLoP
 - Other Conferences
 - Shepherding
 - Useful Documents
 - PLoP Conference Proceedings
 - PLoP Paper Template
- ▶ Patterns
- ▶ Vision

You are here: Home Conferences

PATTERN LANGUAGES OF PROGRAMS (PLoP) CONFERENCES

 We have compiled our collected PLoP experiences into a series named [How to Run PLoP](#)

The Hillside Group Sponsors many different conferences such as: **PLoP**, **EuroPLoP**, **AsianPLoP**, **ScrumPLoP**, **VikingPLoP**, **SugarLoafPLoP**, **UP**, and **ChiliPLoP**. These conferences focus on writing groups to better improve patterns through group exposure. Each conference offers advanced topics for the more adept pattern writers. Participants have the opportunity to refine and extend their patterns with help from knowledgeable and sympathetic patterns enthusiasts.

[Conference Proceedings](#)



PLoP



Pattern Languages of Programs (PLoP™) conference is a premier event for pattern authors and pattern enthusiasts to gather, discuss and learn more about patterns and software development... [Learn More](#)

CHILIPLoP



The 13th Annual ChiliPLoP features "hot topics" for experienced folks... [Learn More](#)

EURO PLoP

SUGARLOAF PLoP



SugarLoafPLoP brings together researchers and practitioners whose interests span a remarkably broad range of topics, who share an interest in exploring the power of the pattern form... [Learn More](#)

Patterns do and do not...

Patterns do...

- provide common vocabulary
- provide “shorthand” for effectively communicating complex principles
- help document software architecture
- capture essential parts of a design in compact form
- show more than one solution
- describe software abstractions

Patterns do not...

- provide an exact solution
- solve all design problems
- only apply for object-oriented design

Patterns can be...

- non-generative (e.g. Gamma patterns)
 - observed in a system
 - descriptive and passive
- generative (e.g. Alexander patterns)
 - generate systems or parts of systems
 - perspective and active

Thank you!