

Evolving Pattern Language towards an Affordance Language

David Ing

Trito Innovation CoLab and Aalto University
<http://coevolving.com>

IBM Research Almaden
May 9, 2018
San Jose, California



Image CC-BY-SA: Celina Laurette (2017) *Escaping from Plato's Cave*



David Ing, 2018

Commercial experience

- Jan. 2018 - now ~ Cofounder, Scientist
Trito Innovation Colab; Toronto, Canada
 - A reconnaissance unit that explores technological and organizational changes ahead for an innovation initiative.
- Nov. 2017 ~ Author, researcher
Open Innovation Learning: Theory building
 - Published book as open access
 - Foreword by Jim Spohrer
- Aug.-Sept. 2015 ~ Managing Director
2013-Feb. 2014 ~ Managing Director
Healthcare EQ, Inc.; Toronto + Jamaica
 - Cofounder of multinational IBM Business Partner, focused on BPM in healthcare
- 1985-2012 ~ alumnus, 28 years of service
IBM Canada, IBM North America
 - Management consultant: IT Strategy, Business Transformation (1994-1997 in U.S.; 2000-2006 from Canada); led Seiad FOAK with Watson Research (1997)
 - Business architect, industry solution sales (2006-2011)
 - Researcher and instructor, Advanced Business Institute (1998-1999)
 - Market development specialist, retail industry, decision support (1988-1993)
 - Econometrician + data scientist, IBM Canada HQ (1995-1998)
 - Websphere Technical Sales (2011-2012)

Academic experience

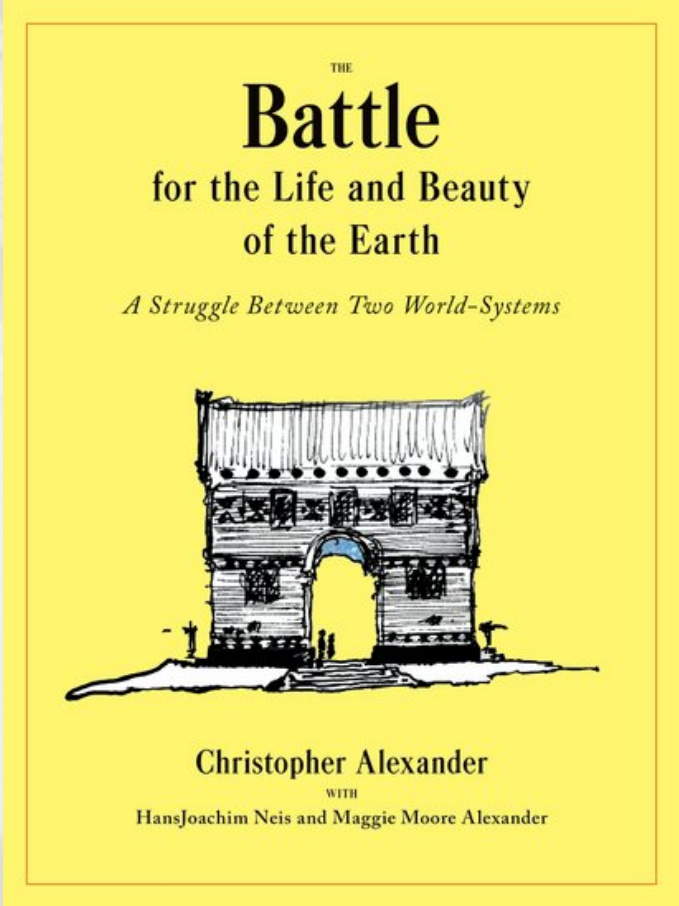
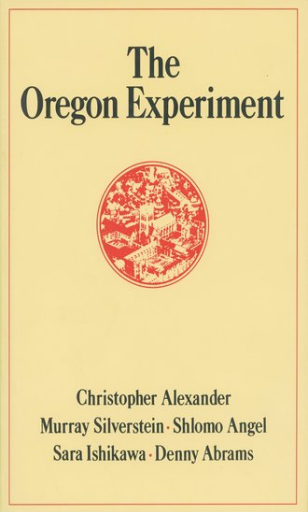
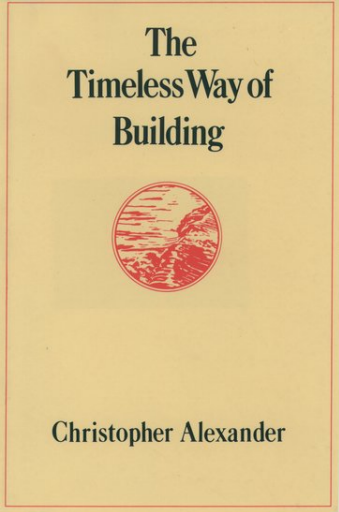
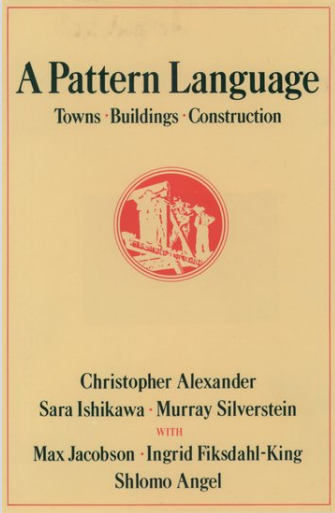
- 2019 PhD Defence
Aalto U. School of Science, Finland
 - Coursework complete
- Feb. 2018
U. Toronto iSchool
 - Taught master's Systems Thinking Systems Design
- 2014-Aug. 2017
Aalto U. School of Science
 - 3 normative theories
 - 3 descriptive theories on 7 cases of *Open Sourcing while Private Sourcing* at IBM 2001-2011
 - Multiparadigm interplay + generative pattern language
- March, Nov. 2017
Tongji U. College of Design&Innovation
 - Taught in Ph.D. Quant Methods
- Jan.-April 2016
Aalto U. Creative Sustainability
 - Rewrote+taught Sys Thinking II
- 2010-2011
Aalto U. Creative Sustainability
 - New masters
- 2006-2008
Helsinki Polytechnic Stadia
 - Funded by Tekes (Finnish Agency for Innovation)
 - Study: business innovation
 - New master's in International Service Business Management
- 1990-1992
U. Toronto Rotman
 - Cofounder Canadian Centre for Marketing Information Technology



Agenda

1. 1964 → 1999 → 2012:
Synthesis of Form → OOPSLA 1996 → Battle (Eishin)
2. 1993 → 2002 → 2006 → 2010:
Hillside Group, IGS Method, AWB, Eclipse
3. 2014 → ...
Wicked Messes, Service Systems Thinking

The writing of 1975-1979 by Alexander was prescriptive; the 2012 is reflections on practice



SUMMARY OF THE PATTERN LANGUAGE

253 patterns on towns, buildings and construction

To look for patterns dealing with particular topics, use the following topic finder: please click on the topic in the following list.... [regions...](#) [town and country boundaries...](#) [urban structures...](#) [communities...](#) [networks...](#) [neighborhood structures...](#) [local centers...](#) [houses...](#) [living rooms](#)

TOWNS

The language begins with patterns that define towns and communities. These patterns can never be designed or built in one fell swoop - but patient piecemeal growth, designed in such a way that every individual act is always helping to create or generate these larger global patterns, will, slowly and surely, over the years, make a community that has these global patterns in it.

First, one all important comment about the region as a whole:

1. INDEPENDENT REGIONS

Within each region work toward those regional policies which will protect the land and mark the limits of the cities:

2. THE DISTRIBUTION OF TOWNS
3. CITY COUNTRY FINGERS
4. AGRICULTURAL VALLEYS
5. LACE OF COUNTRY STREETS
6. COUNTRY TOWNS
7. THE COUNTRYSIDE

Through city policies, encourage the piecemeal formation of those major structures which define the city:

8. MOSAIC OF SUBCULTURES
9. SCATTERED WORK
10. MAGIC OF THE CITY
11. LOCAL TRANSPORT AREAS

159 LIGHT ON TWO SIDES OF EVERY ROOM

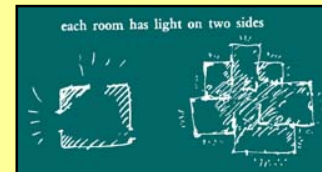
... once the building's major rooms are in position, we have to fix its actual shape; and this we do essentially with the position of the edge. The edge has got its rough position already from the overall form of the building - WINGS OF LIGHT (107), POSITIVE OUTDOOR SPACE (106), LONG THIN HOUSE (109), CASCADE OF ROOFS (116). This pattern now completes the work of WINGS OF LIGHT (107), by placing each individual room exactly where it needs to be to get the light. It forms the exact line of the building edge, according to the position of these individual rooms. The next pattern starts to shape the edge.



When they have a choice, people will always gravitate to those rooms which have light on two sides, and leave the rooms which are lit only from one side unused and empty.

Therefore:

Locate each room so that it has outdoor space outside it on at least two sides, and then place windows in these outdoor walls so that natural light falls into every room from more than one direction.



The language connects patterns to higher and lower orders

→ ↻ ↗ ⓘ www.jacana.plus.com/pattern/P159.htm ☆ 🔊 ⚙️ 🔒 🗄️

LIGHT ON TWO SIDES OF EVERY ROOM ** 159

Buildings: Internal-external connections [Menu](#) [Prev](#) [Next](#)

Problem
When they have a choice, people will always gravitate to those rooms which have light on two sides, and leave the rooms which are lit only from one side unused and empty.

Solution
Locate each room so that it has outdoor space outside it on at least two sides, and then place windows in these outdoor walls so that natural light falls into every room from more than one direction.

Select High Order Pattern and Go to it.	Select Low Order Pattern and Go to it.
106 POSITIVE OUTDOOR SPACE ** ▲	106 POSITIVE OUTDOOR SPACE ** ▲
107 WINGS OF LIGHT ** ▲	180 WINDOW PLACE ** ▲
109 LONG THIN HOUSE * ▲	192 WINDOWS OVERLOOKING LIFE * ▲
	209 ROOF LAYOUT * ▲

→ ↻ ↗ ⓘ www.jacana.plus.com/pattern/P127.htm ☆ 🔊 ⚙️ 🔒 🗄️

INTIMACY GRADIENT ** 127

Buildings: Gradients of space and movement [Menu](#) [Prev](#) [Next](#)

Problem
Unless the spaces in a building are arranged in a sequence which corresponds to their degrees of privateness, the visits made by strangers, friends, guests, clients, family, will always be a little awkward.

Solution
Lay out the spaces of a building so that they create a sequence which begins with the entrance and the most public parts of the building, then leads into the slightly more private areas, and finally to the most private domains.

Select High Order Pattern and Go to it.	Select Low Order Pattern and Go to it.
96 NUMBER OF STORIES * ▲	129 COMMON AREAS AT THE HEART ** ▲
107 WINGS OF LIGHT ** ▲	130 ENTRANCE ROOM ** ▲
110 MAIN ENTRANCE ** ▲	141 A ROOM OF ONE'S OWN ** ▲
	142 SEQUENCE OF SITTING SPACES * ▼

127 Intimacy Gradient** (#1 of 2)

. . . if you know roughly where you intend to place the building wings -- WINGS OF LIGHT (107), and how many stories they will have -- NUMBER OF STORIES (96), and where the MAIN ENTRANCE (110) is, it is time to work out the rough disposition of the major areas on every floor. In every building the relationship between the public areas and private areas is most important.

* * *

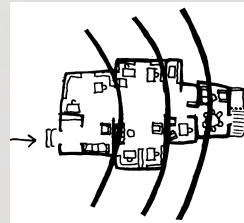
Unless the spaces in a building are arranged in a sequence which corresponds to their degrees of privateness, the visits made by strangers, friends, guests, clients, family, will always be a little awkward.

In any building -- house, office, public building, summer cottage - people need a gradient of settings, which have different degrees of intimacy. A bedroom or boudoir is most intimate; a back sitting room. or study less so; a common area or kitchen more public still; a front porch or entrance room most public of all. When there is a gradient of this kind, people can give each encounter different shades of meaning, by choosing its position on the gradient very carefully. In a building which has its rooms so interlaced that there is no clearly defined gradient of intimacy, it is not possible to choose the spot for any particular encounter so carefully; and it is therefore impossible to give the encounter this dimension of added meaning by the choice of space. This homogeneity of space, where every room has a similar degree of intimacy, rubs out all possible subtlety of social interaction in the building.

We illustrate this general fact by giving an example from Peru - a case which we have studied in detail. [...]

The intimacy gradient is unusually crucial in a Peruvian house. But in some form the pattern seems to exist in almost all cultures. We see it in widely different cultures -- compare the plan of an African compound, a traditional Japanese house, and early American colonial homes -- and it also applies to almost every building type -- compare a house, a small shop, a large office building, and even a church. It is almost an archetypal ordering principle for all man's buildings. All buildings, and all parts of buildings which house well defined human groups, need a definite gradient from "front" to "back," from the most formal spaces at the front to the most intimate spaces at the back.

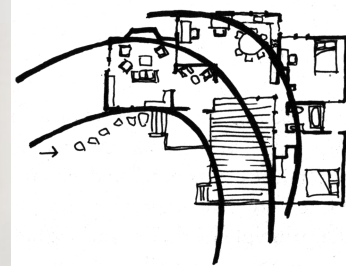
In an office the sequence might be: entry lobby, coffee and reception areas, offices and workspaces, private lounge.



Office intimacy gradient.

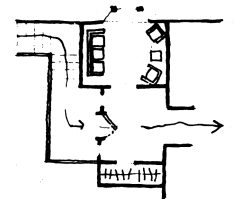
In a small shop the sequence might be: shop entrance, customer milling space, browsing area, sales counter, behind the counter, private place for workers.

In a house: gate, outdoor porch, entrance, sitting wall, common space and kitchen, private garden, bed alcoves.



Intimacy gradient in a house.

And in a more formal house, the sequence might begin with something like the Peruvian sala -- a parlor or sitting room for guests.



Formal version of the front of the gradient.

127 Intimacy Gradient** (#2 of 2)

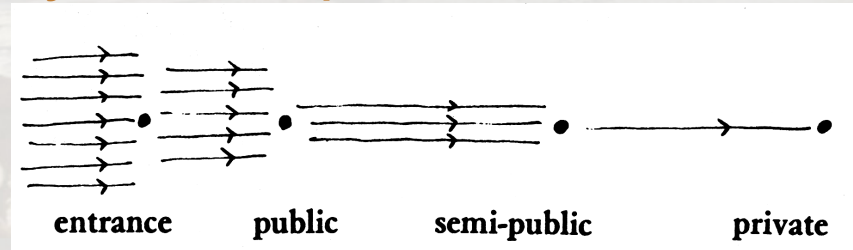
... if you know roughly where you intend to place the building wings -- WINGS OF LIGHT (107), and how many stories they will have -- NUMBER OF STORIES (96), and where the MAIN ENTRANCE (110) is, it is time to work out the rough disposition of the major areas on every floor. In every building the relationship between the public areas and private areas is most important.

* * *

Unless the spaces in a building are arranged in a sequence which corresponds to their degrees of privateness, the visits made by strangers, friends, guests, clients, family, will always be a little awkward.

Therefore:

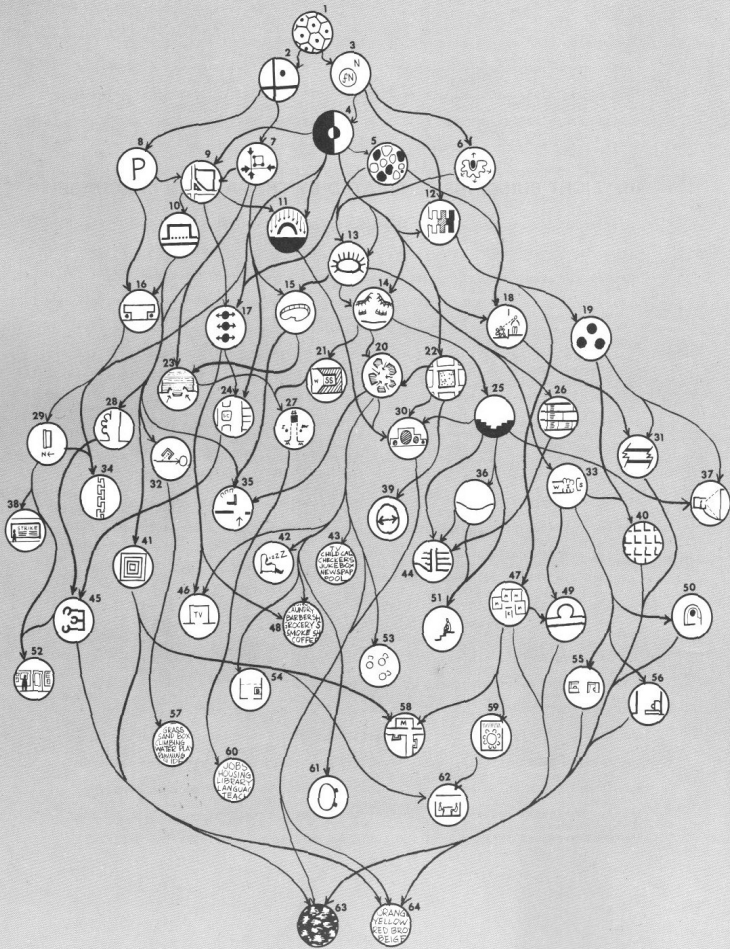
Lay out the spaces of a building so that they create a sequence which begins with the entrance and the most public parts of the building, then leads into the slightly more private areas, and finally to the most private domains.



* * *

At the same time that common areas are to the front, make sure that they are also at the heart and soul of the activity, and that all paths between more private rooms pass tangent to the common ones -- COMMON AREAS AT THE HEART (129). In private houses make the ENTRANCE ROOM (130) the most formal and public place and arrange the most private areas so that each person has a room of his own, where he can retire to be alone A ROOM OF ONE'S OWN (141). Place bathing rooms and toilets half-way between the common areas and the private ones, so that people can reach them comfortably from both BATHING ROOM (144); and place sitting areas at all the different degrees of intimacy, and shape them according to their position in the gradient - SEQUENCE OF SITTING SPACES (142). In offices put RECEPTION WELCOMES YOU (149) at the front of the gradient and HALF-PRIVATE OFFICE (152) at the back. . . .

Pattern language intends to give 3 types of help



1. It gives him the opportunity to use the patterns in the way which pays full respect to the **unique features** of each special building: the local peculiarities of the community, its special needs ...
2. It tells him which patterns to consider **first**, and which ones to consider **later**. Obviously he wants to consider the **biggest ones** ... before he considers the **details**.
3. It tells him which patterns "**go together**" ... so that he knows which ones to think about at the same time, and which ones separately (Alexander et al., 1968, pp. 17–19).

The essential idea of a pattern language is: *a solution to a problem in context*

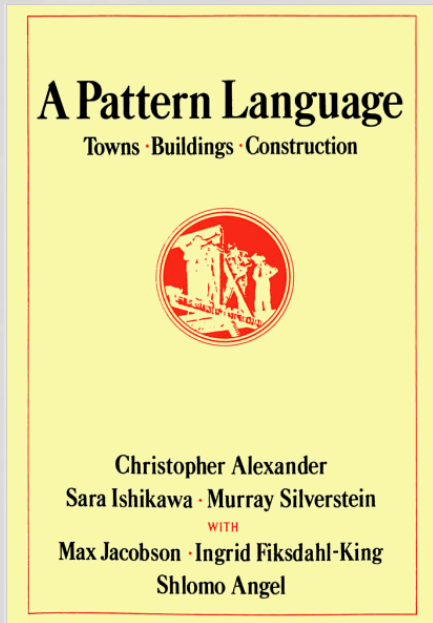
Every time a designer creates a pattern (or, for that matter, entertains any idea about the physical environment), he essentially goes through a three-step process.

He considers a PROBLEM, invents a PATTERN to solve the problem, and makes mental note of the range of CONTEXTS where the pattern will solve the problem. [....]

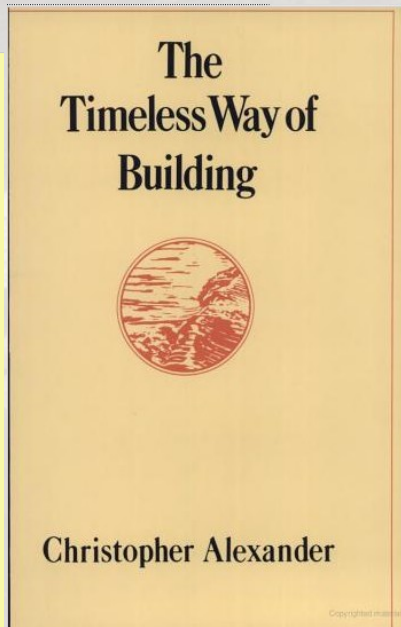
The format says that whenever a certain **CONTEXT** exists, a certain **PROBLEM** will arise; the stated **PATTERN** will solve the **PROBLEM** and there should be provided in the **CONTEXT**.

While it is not claimed that the PATTERN specified is the only solution to the PROBLEM, it is implied that unless the PATTERN or an equivalent is provided, the PROBLEM will go unsolved (Alexander, Ishikawa, & Silverstein, 1967, pp. 1–4).

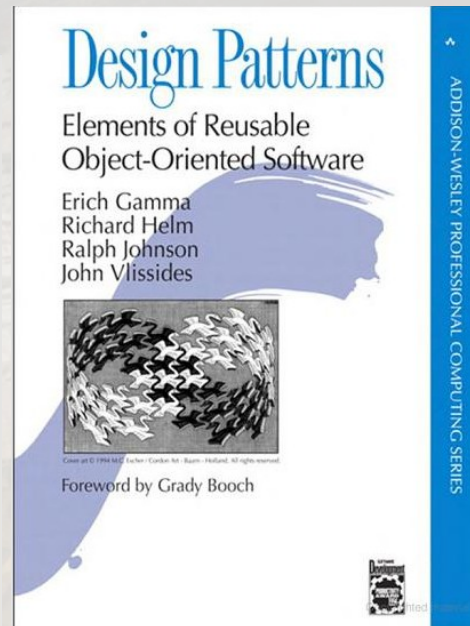
An evolution of pattern languages across domains



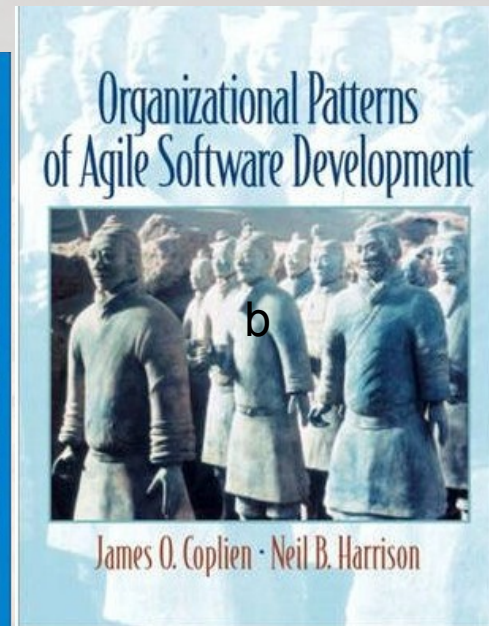
1977



1979



1994



2005

Here is a short and necessarily incomplete definition of a pattern:

A recurring structural configuration that solves a problem in a context, contributing to the wholeness of some whole, or system, that reflects some aesthetic or cultural value.[1]

Pattern Name: A name by which this problem/solution pairing can be referenced

Problem: The specific problem that needs to be solved.

Context

The circumstances in which the problem is being solved imposes constraints on the solution. The context is often described via a "situation" rather than stated explicitly.

Forces

The often contradictory considerations that must be taken into account when choosing a solution to a problem.

Solution: The most appropriate solution to a problem is the one that best resolves the highest priority forces as determined by the particular context.

Resulting Context

The context that we find ourselves in after the pattern has been applied. It can include one or more new problems to solve

Rationale

An explanation of why this solution is most appropriate for the stated problem within this context.

Related Patterns

The kinds of patterns include:

- Other solutions to the same problem,
- More general or (possibly domain) specific variations of the pattern,
- Patterns that solve some of the problems in the resulting context (set by this pattern)

Source: [1] Coplien, James O., and Neil B. Harrison. 2004. *Organizational Patterns of Agile Software Development*. Prentice-Hall, Inc.

<http://books.google.ca/books?id=6K5QAAAAMAAJ> . [2] Gerard Meszaros and Jim Doble, "A Pattern Language for Pattern Writing", *Pattern Languages of Program Design* (1997), <http://hillside.net/index.php/a-pattern-language-for-pattern-writing>

1996/10/08 Christopher Alexander, "Patterns in Architecture", OOPSLA '96

Christopher Alexander's presentation at the 1996 OOPSLA Conference was lightly edited in the 1999 article. Watching the video and reading the text, the divergences are small until 46 minutes into 63-minutes, when the text was significantly rewritten.

The digest maps the published 1999 article to the 1996 presentation on video.

- Alexander, Christopher. 1996. "Patterns in Architecture" presented at *OOPSLA '96*, October 8, San Jose, California. <https://www.youtube.com/watch?v=98LdFA-zfA>.
- Alexander, Christopher. 1999. "The Origins of Pattern Theory: The Future of the Theory, and the Generation of a Living World." *IEEE Software*, 16 (5): 71–82. doi:10.1109/52.795104. <http://dx.doi.org/10.1109/52.795104>.



[03:37] In effect, I'm just going to do three things.



THE ORIGINS OF PATTERN THEORY

THE FUTURE OF THE THEORY, AND THE GENERATION OF A LIVING WORLD

Christopher Alexander

Introduction by James O. Coplien



nce in a great while, a great idea makes it across the boundary of one discipline to take root in another. The adoption of Christopher Alexander's patterns by the software community is one such event. Alexander both commands respect and inspires controversy in his own discipline; he is the author of several books with long-running publication records, the first recipient of the AIA Gold Medal for Research, a member of the Swedish Royal Academy since 1980, a member of the American Academy of Arts and Sciences, recipient of dozens of awards and honors including the Best Building in Japan award in 1985, and the American Association of Collegiate Schools of Architecture Distinguished Professor Award. It is odd that his ideas should have found a home in software, a discipline that deals not with timbers and tiles but with pure thought stuff, and with ephemeral and weightless products called programs. The software community embraced the pattern vision for its relevance to problems that had long plagued software design in general and object-oriented design in particular.

The Quality Without a Name

Alexander's search, culminating in pattern languages, was to find **an objective (rather than a subjective) meaning for beauty, for the aliveness that certain buildings, places, and human activities have**. The objective meaning is the quality without a name, and I believe we cannot come to grips with Alexander in the software community unless we come to grips with this concept. [...]

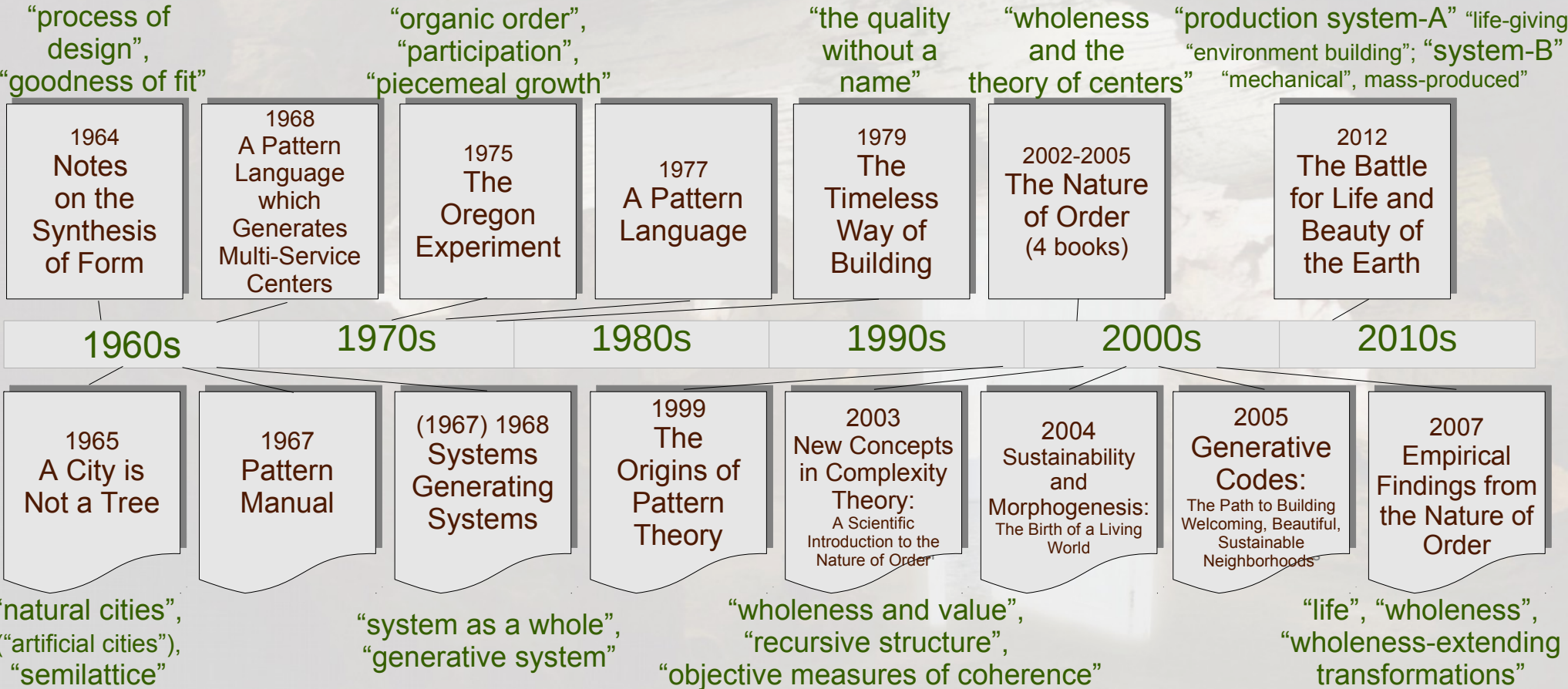
The quality is an objective quality that things like buildings and places can possess that makes them good places or beautiful places. Buildings and towns with this quality are habitable and alive. **The key point to this** -- and the point that really sets Alexander apart from his contemporaries and stirs philosophical debate -- **is that the quality is objective**.

It started in 1964 when he was doing a study for the Bay Area Rapid Transit (BART) system One of the key ideas in this book was that **in a good design there must be an underlying correspondence between the structure of the problem and the structure of the solution** — good design proceeds by writing down the requirements, analyzing their interactions on the basis of potential misfits, producing a hierarchical decomposition of the parts, and piecing together a structure whose structural hierarchy is the exact counterpart of the functional hierarchy established during the analysis of the program. (Alexander 1964)

Alexander was studying the system of forces surrounding a ticket booth, and he and his group had written down 390 requirements for what ought to be happening near it. Some of them pertained to such things as being there to get tickets, being able to get change, being able to move past people waiting in line to get tickets, and not having to wait too long for tickets. What he noticed, though, was that **certain parts of the system were not subject to these requirements and that the system itself could become bogged down because these other forces — forces not subject to control by requirements—acted to come to their own balance within the system**. For example, if one person stopped and another also stopped to talk with the first, congestion could build up that would defeat the mechanisms designed to keep traffic flow smooth. Of course there was a requirement that there not be congestion, but **there was nothing the designers could do to prevent this by means of a designed mechanism**.

Alexander proposes some words to describe the quality without a name, but even though he feels they point the reader in a direction that helps comprehension, **these words ultimately confuse. The words are alive, whole, comfortable, free, exact, egoless, and eternal**. I'll go through all of them to try to explain the quality without a name.

Over 50 years, Christopher Alexander and coauthors evolved concepts and language in built environments



Agenda

1. 1964 → 1999 → 2012:
Synthesis of Form → OOPSLA 1996 → Battle (Eishin)
2. 1993 → 2002 → 2006 → 2010:
Hillside Group, IGS Method, AWB, Eclipse
3. 2014 → ...
Wicked Messes, Service Systems Thinking



[Front Page](#)

This [ContentCreationWiki](#) is focused on [PeopleProjectsAndPatterns](#) in [SoftwareDevelopment](#).

The idea of a "Wiki" may seem odd at first, but dive in, explore its links and it will soon seem familiar. "Wiki" is a composition system; it's a discussion medium; it's a repository; it's a mail system; it's a tool for collaboration. We don't know quite what it is, but we do know it's a fun way to communicate asynchronously across the network.

To find a page on any specific topic, go to [FindPage](#). To see an auto-generated list of pages which have changed recently, try [RecentChanges](#). If you want a short list of randomly-selected pages, try [RandomPages](#). [CategoryCategory](#) is the top level of page categorization; you can use it to delve deeper into the site.

Edit pages by using the [EditText](#) link at the bottom of the page you wish to edit. Don't worry too much about messing up, as the original text is backed up and can be easily restored (meaning, everyone can see the changes made, and will be able to correct mistakes, erase, and so on, if necessary).

The [TextFormattingRules](#) are quite simple, and the [TipsForBeginners](#) will help you learn to apply them gracefully. You'll probably want to start by editing pages that already exist. The [WikiWikiSandbox](#) is set aside for



[Design 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. --
[ChristopherAlexander](#)

A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples. The solution is a general arrangement of objects and classes that solve the problem. The solution is customized and implemented to solve the problem in a particular context. - [DesignPatternsBook](#)

Some topics that categorize [DesignPatterns](#) into the [GangOfFour](#) categories:

Given that patterns could be applied to many different disciplines, I would suggest that we talk about [SoftwareDesignPatterns](#), to differentiate from [ArchitecturalDesignPatterns](#) or other kinds. Then the question is, are there any design patterns that work across specific disciplines? I doubt it, although there may be some "meta" patterns...



Portland Pattern Repository

We're writing about computer programs in a new stylistic form called [pattern languages](#). The form has many internal references which map well to hypertext links. We've added links to published (or soon to be published) documents. Short summaries appear in the...

- [Pattern Language Catalog](#)

We've also created a space for exploring the not-quite-yet patterns we all carry around in our heads...

- [People, Projects & Patterns](#)

The Hillside Group's [Patterns Home Page](#) lists other pattern resources including papers, books, conferences.

New [survey results](#) are in. This form tallies survey responses as they are made. Have a look to see what people like about the repository. Then add your two-cents worth.



MAIN MENU

- ▶ Hilltop
- ▶ Books
- ▶ Contact
- ▶ Conferences
- ▶ **Patterns**
 - Books
 - Patterns Catalog
 - About Patterns
 - TPLoP
 - Education
 - Mailing Lists
 - Writing
 - Tools
 - Links
- ▶ Vision
- ▶ Wiki

Home You are here: [Home](#) > [Patterns](#)

DESIGN PATTERNS LIBRARY

Welcome to the patterns home page. It is a source for information about all aspects of software [pattern languages](#). If you are new to patterns, James Coplien and Richard Gabriel have created a succinct [patte](#)

Patterns and Pattern Languages are ways to describe best practices, good designs, and capture experience in a way that it is possible for others to reuse this experience. The Hillside Group takes pleasure in sponsoring many different [PLoP conferences](#) that are provided for the betterment of the pattern community.

Fundamental to any science or engineering discipline is a common vocabulary for expressing its concepts, and a language for relating them. The goal of patterns within the software community is to create a body of literature where software developers resolve recurring problems encountered throughout all of software development. Patterns help create a shared language for communicating insight and experience about these problems and their solutions. Formally codifying these solutions and their relationships lets us successfully capture the body of knowledge which defines our understanding of good architectures that meet the needs of their users. Forming a pattern language for conveying the structures and mechanisms of our architecture lets us intelligibly reason about them. The primary focus is not so much on technology as on creating a culture to document and support sound engineering architecture and

- [Brad Appleton](#)



1997: Seiad Foak, Watson Research with Consumer-Driven Supply Chain

Project Goals...

The First-of-a-Kind projects bridge from some current directions to next-generation CDSC directions

Information-Intensive Integrated Supply Chain Reference Model

Marketing Microworld

- ▶ Next-generation "integrated outcome simulation" envisioned
- ▶ Towards "Managing by Wire"

System Envisioning Tool

- ▶ Prototype for next-generation dynamic analysis models
- ▶ Potential IBM Software Solutions product in 1999

Seiad Technology Project

Marketing Microworld

- ▶ Next-generation "integrated outcome simulation" envisioned
- ▶ Towards "Managing by Wire"

System Envisioning Tool

- ▶ Prototype for next-generation dynamic analysis models
- ▶ Potential IBM Software Solutions product in 1999



Agenda

The Projects, & Goals

Project Descriptions

Work Plan

Methods

Research Foundations

Analysis Patterns -- Decision Processes

Object Modeling -- Supply Chain

The Seiad Vision

- ▶ "Managing by Wire"
- ▶ Seiad Manifesto / "Stake in the Ground"

Executable Business Models

- ▶ Application orientation vs. mirror worlds
- ▶ Demonstration

Enterprise Builder

- ▶ Enterprise scale through visual builders
- ▶ Demonstration

Analytical Marketing Models

- ▶ Long-term research agenda

Linkages to ESS Project

- ▶ Initial alignment with Enterprise Support Systems -- cross-industry



1998: OOPSLA System Envisioning Workshop



Conference Chair:
Bjorn Freeman-Benson,
[Object Technology International](#)

Program Chair:
[Craig Chambers](#),
[University of Washington](#)

[Conference Committee](#)

[Program Committee](#)

[OOPSLA'98 Office](#)

[Mid-Year Workshops](#)

[Suggested Readings](#)

[ISMM '98](#)

[\[Text-only Home\]](#)

System envisioning (OOPSLA 1998 workshop summary)

Reported by: Charles E. Matthews and Ralph Hodgson

Workshop Organizers: Ralph Hodgson, Tom Bridge, Charles E. Matthews, Robert Coyne, Bruce Anderson, Deborah Leishman, Doug McDavid, Carl Ballard

Editorial note by David Ing: This report is republished on coevolving.com with the permission of Ralph Hodgson received 2006/02/16. The original article is not available online, but the reference is provided as <http://doi.acm.org/10.1145/346852.346964> Some addresses at the end of the article have been corrected.

Overview

Systems are conceived out of an understanding and conceptualizing of a problem space. System Envisioning is about how we create possibilities for what a system might and should do and seeks to answer:

- How do we formulate and choose among alternative concepts of a system?
- What considerations affect the trade-offs and the interrelationships between requirements, specification, and design?
- How are these aspects of system development affected by the political, social, and cultural issues within an organization?

Motivations

This workshop was motivated by an interest in sharing experiences on the relationships between problem domain understanding and creative thinking on formulating systems concepts. We were interested in how different types of thinking and action are involved in developing the conceptual architecture of a system. Particularly, we were concerned with requirements elicitation and generation, organizational design, systems thinking, holonics and cybernetics, object thinking, creativity and imagineering, metaphorical exploration, synectics and analogical reasoning, human communications and dialog-based interaction.

Goals and Objectives

We wanted to identify motivational interests and to share experiences on how system envisioning has happened and can happen in system development projects – including experiences related to the effectiveness of tools used within the specification and development process. We wanted to share stories on “leaving behind” an “as-is” system to:

- re-conceptualize the possibilities of a software system;
- identify interventions and techniques for imagining and sharing the possibilities of a “could-be” or “to-be” system;
- share ideas and experiences on the role of metaphors and analogies in system conceptualization;
- explore the relationships between systems thinking and object thinking;

Putting It All Together

Towards a Pattern Language for Interaction Design: A CHI 97 Workshop

Elisabeth Boyle, Rachel Bellamy, George Casaday, Thomas Erickson, Sally Fincher, Beki Grinter, Ben Gross, Diane Lehder, Hans Marmolin, Brian Moore, Colin Potts, Grant Skousen, John Thomas

Pattern languages are representations that have been used in architecture and urban design for about twenty years. They focus on the interaction between physical form and social behavior, and express design solutions in an understandable and generalizable form. But pattern languages are not simply sets of patterns intended to be universally applied; instead, they are actually meta-languages which, when used in a particular situation, generate situated design languages. This report describes a CHI 97 workshop which explored the utility of pattern languages for interaction design. We discuss the workshop's rationale, the structure and process of the workshop, and some of the workshop's results. In particular, we describe some patterns developed as part of the workshop, and our consequent reflections on the use of patterns and pattern languages as lingua franca for interaction design. This report concludes with a bibliography on pattern languages and related matters that spans architecture, software design, and organizational design.

Introduction

The Challenge: Complexity and Diversity

Interaction design is becoming an increasingly complex and diverse activity. It is becoming more complex in that communications and computational technologies, as they become cheaper and smaller, are being integrated into more devices and embedded in more environments. This, in turn, makes interaction design relevant for an increasing number of new application domains. And even as the space of design possibilities increases, workplace studies are making us aware of the complexity of the socio-technical systems within which we are working to integrate new technologies. How are we to manage this complexity?

Interaction design is becoming more diverse in that a wider range of people are becoming involved in it. Within CHI, it is well accepted that anthropologists, psychologists, and visual designers, as well as engineers and computer scientists, have roles to play in systems design. As computing systems shrink in size, industrial and product designers need to work hand in hand with systems designers. The advent of virtual spaces create roles for architects and interior designers. The commercialization of video and multimedia technologies create roles for musicians, film producers, et al. While the multidisciplinary nature of interaction design brings much richness, it is also challenging because no common perspective, set of practices, or theoretical orientation can be assumed.

Another factor driving the diversification of interaction design is customization. As systems become increasingly customizable, more and more design—in the sense of front end creation, application programming, and software configuration—is being done in-house. Sometimes this means that traditional MIS departments are playing a role; sometimes it means that external consultants are involved; sometimes it means that end users themselves participate. In many cases these participants lack the time, resources, or inclination to engage in research on the needs and practices of their users. And, in many cases, these participants lack formal training in design, and hence any common perspective or language.

A Possible Solution: Pattern Languages

So, we have a rapidly expanding game: more players and more technology projected onto workplaces which we are learning more and more about. This increasing complexity and diversity can

be source of richness, or of chaos. Thus, we need to explore ways of dealing with the increasing complexity and diversity of the interaction design field. This workshop explored one approach to putting it all together through a common language. Our model is the work of Christopher Alexander and his colleagues who over the last few decades have looked at what works and what doesn't work in architecture and urban design. The basic approach is to closely examine particular cases, attempt to identify recurring patterns and integrate them into a language of relatively concrete patterns. Their work is codified in the book *A Pattern Language* [1]; each pattern is described; examples are given; empirical data supporting the pattern are referenced; and the relationships to other patterns are defined. The way of using the patterns, that is, the process of design, is described in a companion volume, *The Timeless Way of Building* [2].

Let's take a brief look at Alexander's Pattern Language. The language consists of a network of over 250 patterns. The patterns cover a range of scales, from a pattern for the distribution of towns and cities to patterns for walls and room sizes. The patterns are loosely connected across scales: any given pattern typically points to smaller scale patterns which can support it, and larger scale patterns in which it may participate. For example, a pattern called 'Identifiable Neighborhood' (aimed, obviously enough, at creating neighborhoods with their own particular sense of place) will involve a number of lower level patterns. Possibilities include 'Street Café', 'Individually Owned Shops', 'Corner Grocery', 'Beer Hall', and so on. At the same time, 'Identifiable Neighborhood' partici-



Lingua Francas for Design: Sacred Places and Pattern Languages

Thomas Erickson

IBM T. J. Watson Research Center
P.O. Box 704 Yorktown Heights, NY 10598 USA
(01) 914 784-7577

snowfall@acm.org

ABSTRACT

A central challenge in interaction design has to do with its diversity. Designers, engineers, managers, marketers, researchers and users all have important contributions to make to the design process. But at the same time they lack shared concepts, experiences and perspectives. How is the process of design—which requires communication, negotiation and compromise—to effectively proceed in the absence of a common ground? I argue that an important role for the interaction designer is to help stakeholders in the design process to construct a *lingua franca*. To explore this issue, which has received remarkably little attention in HCI, I turn to work in urban design and architecture. I begin by discussing a case study in community design, reported by Hester [10], that demonstrates the power of a *lingua franca* for a particular design project. I then describe the concept of pattern languages and discuss how they might be adapted to the needs of interaction design in general, and used, in particular, as meta-languages for generating *lingua francas* for particular design projects.

Keywords

Design methods, patterns, pattern language, interaction design, interdisciplinary design, architecture, urban design.

1. INTRODUCTION

The fundamental premise of this paper is that the design of interactive systems is, at its heart, a communicative process. Successful design requires communication—presentation, discussion, disagreement, negotiation, compromise, and so on—among a diverse array of people.

If we take this premise seriously, it raises the question of how we go about supporting the communicative aspect of design. And, indeed, there has been a lot of work to this end. The basic

repertoire of designers' tools—storytelling, scenario-making, prototype building, user testing, etc.—are all methods which aim to improve communication. However, often these tools are 'owned' by the designers—that is, they require the designers' expertise to deploy, administer, operate or interpret. Workers in the participatory design tradition (e.g. [13, 15]) have explored various approaches to making these tools more open, but more remains to be done.

The fundamental goal of this paper is to describe and explore one approach to making communication in design a more egalitarian process: the notion of creating a *lingua franca*—a common language—for a design project. The idea is that a *lingua franca* is accessible to all stakeholders, particularly those who are traditionally marginalized in the design process: the users. A secondary goal is to encourage readers to join in exploring ways of creating design *lingua francas*. That is, I strive to raise important questions, share provocative examples, point to common resources, and suggest productive directions for research and design practice; however this is a new area of investigation for HCI—those looking for solid, empirically validated answers will not find them here.

The paper begins by making the case for a *lingua franca*, arguing that the increasing complexity and diversity of the design process creates a need for common languages. Next, drawing from the urban design literature, it describes the example of the redevelopment of the town of Manteo, North Carolina [10]. Manteo provides an illustration of the possibility—and the power—of creating a *lingua franca* as part of a design project. The next two sections turn to the topic of pattern languages. Originally developed in architecture by Alexander and his colleagues [1, 2], pattern languages have been taken up by the object oriented programming community [6, 11, 12, 16, 17, 19] and are now seeing increasing interest from the HCI community [3, 5, 7, 8, 18]. The sections describe Alexandrian pattern languages, discuss why they are well suited to the generation of *lingua francas*, and explore ways in which they might be used to support interaction design. The paper ends with a discussion of questions and issues raised by the paper's reviewers, and a summary.

2. THE NEED FOR A LINGUA FRANCA

The interactive systems design space is growing rapidly. From the technical vantage point it is evident that microprocessors are continuing to shrink in terms of their cost, power, and space

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DIS '99, Brooklyn, New York.
Copyright 2000 ACM 1-58113-219-0/00/0008...\$5.00.

Configurable DEVELOPMENT PROCESSES

Keeping the focus on what is being produced.

JOHN CAMERON

THE DIVERSITY OF IT PROJECTS FRUSTRATES ANY DIRECT ATTEMPT TO SYSTEMATIZE THE PROCESSES USED FOR THEIR DEVELOPMENT. ONE SIZE JUST WON'T FIT ALL. EVEN THREE OR FOUR SIZES AREN'T ENOUGH BECAUSE THE SET OF PROJECTS DOESN'T NEATLY DIVIDE INTO THREE OR FOUR SIMPLE CATEGORIES. A MORE FLEXIBLE AND CONFIGURABLE APPROACH TO PROCESS GUIDANCE IS NEEDED, A WAY OF TAILORING THE PROCESS TO THE NEEDS OF EACH PARTICULAR PROJECT.

To make processes configurable there must be some concept of modularity. It must be possible to select different subsets of the available modules and put them together in a coherent way. The scheme proposed here is very simple. The main focus is on the tangible things produced. They are identified (at a certain level of granularity) as "work products" and a descriptive module created for each distinct type. The modules, called Work Product Descriptions (WPDs), describe what the work product is, why and when it is needed, and how it is produced. The WPDs comprise an important subset of the configurable process framework. The process is configured to a particular situation by deciding which work products need to be produced and then making choices about sequencing and phasing.

Work products cover the full range of project work including project management, business process design, organizational change, requirements, usability, architecture, design, construction, and testing. Figure 1, for example, shows work products associated with the application development part of the framework.

The dynamic stability model [4] provides a management consultant's perspective on this approach to configuration. This model classifies industrial production processes into invention (meaning each product is uniquely designed and built), mass production, continuous improvement, and mass cus-

tomization. To achieve the generally desirable goal of mass customization, in which product and process are both customized to the customer's needs, it is necessary to have modular processes and a means of configuring them. Similarly, the sense-and-respond model of business organization [1], whose goal is responsive, adaptive enterprises, also relies on modular descriptions of capabilities.

Experience at IBM

The work product approach was first developed and used at IBM by the Object-Oriented Technology Center, a group since disbanded, but whose mission from 1994-96 was to support internal OO projects. One of the main reasons for their emphasis on WPDs was the difficulty they found in reaching consensus on the process aspects of development. They found it easier to agree on the artifacts that have to be produced; their work is described in [3].

Since 1996 a number of other IBM working groups have adopted the approach. The scope has been substantially extended, for example to cover project management, various consulting methodologies, and a wide range of specialist technical disciplines. Over 300 WPDs are in use, most of them shared by many groups. The approach has been standard in most of IBM Global Services since September 2000.

Figure 1. List of 96 WPDs used in IBM custom application development (v1.1).

Acceptance Test Plan	Configuration Management Procedures	IT Readiness Assessment and Issues	System Context Diagram
Analysis Class Descriptions	Cost-Benefit Impact Analysis	Logical Data Model	System Management Plan
Analysis Class Diagram	Current IT Infrastructure	Nonfunctional Requirements	System Test Plan
Analysis Guidelines	Current Solution Evaluation	Object/Access Table	Technical Prototype
Analysis Interaction Diagram	Customer View and Requirements	Operational Model	Test Case
Analysis State Chart Diagram	Decision Framework	Organization Change Readiness Assessment and Issues	Test Results
Application Program Interface	Deployment Plan	Package Technical Criteria	Training and User Support Approach
Architectural Decisions	Deployment Unit	Physical Database Design	Transaction Descriptions
Architectural Template	Deployment Unit Hierarchy	Physical Packaging	Usability Design and Evaluation Plan
Architectural Overview Diagram	Design Class Descriptions	Process Model (data flow diagrams)	Usability Requirements
As-Is Organization Assessment	Design Class Diagram	Process/Usage Usage Matrix	Usability Test Plan
As-Is Organization Description	Design Guidelines	Program Module Invocation Model	Usability Test Report
As-Is Process Definition and Assessment	Design Interaction Diagram	Program Module Specification	Use Case Model
Build procedures	Design State Chart Diagram	Project Estimates	Use Case Realization Report
Business Context Diagram	Early Usability Evaluation	Project Goals	User Interface Architecture
Business Event List	Education and Training Plan	Project Tracking Best Practices	User Interface Conceptual Model
Business Object Model	End-User Training Materials	Project Workbook Outline	User Interface Design Guidelines
Business Process Model	End-User Training Specifications	Reference Architecture Fit/Gap Analysis	User Interface Design Specification
Business Rule Catalog	Envisioned To-Be Business Goals	Release Plan	User Interface Prototype
Code/Class Access List	Executables	Request for Information	User Profiles
Change Cases	Glossary	Request for Vendor Proposal and Response	User Support Materials
Classified Business Terms	Incremental Goals	Service Level Characteristics Analysis	Vendor Qualification
Coding Guidelines	Information Technology Standards	Software Distribution Plan	Visual Assessment
Component Model	IT Organization Skills Gap Analysis	Source Code	Visual Resources

ment Approach section of the WPD is not sufficient. They can differentiate the use of the same WPD in different contexts.

Within IBM the term "engagement model" is used for all the material needed to describe a certain class of project. An engagement model consists of a set of WPDs, a WBS, a set of role descriptions, and a set of techniques. The management of the process framework is quite complicated. Engagement models and a few of the specialist elements they contain are managed by the groups that do the projects they describe. Other groups manage the WPDs, roles, and other reusable components.

THE WORK PRODUCT APPROACH TO CONFIGURABLE PROCESSES IS AN ATTEMPT TO STRUCTURE AND MANAGE THE KNOWLEDGE IN A VERY COMPLEX DOMAIN, KNOWLEDGE ABOUT HOW TO DO IT PROJECTS.

valuable part of any method. So, more is needed than just WPDs.

The Rest of the Process Framework

The process framework scheme used by IBM has four main components:

- Work Product Descriptions, classified by subject matter, with associated dependency diagrams, as described here.
- Work Breakdown Structures (WBS) describe the temporal structure of a project. A WBS is a skeletal plan, which divides the project into a hierarchical structure of major and minor checkpoints each with exit criteria and a description of the work needed to reach the checkpoint.
- Roles describe sets of skills. They are associated with WPDs and with elements in the WBS.
- Techniques are used for detailed guidance on building a work product or group of work products, when the terse summary in the Develop-

Configuring the Process Framework

Configuration plays a central role in methods based on WPDs. This represents a psychological shift in the role of method. All too often, deviation from a standard methodology is seen as an imperfection, as an unwelcome compromise (despite the fact it always happens!). This attitude is sometimes encouraged by methodologists who, as a group, are not noted for their flexibility. Instead, adapting to particular circumstances should be the norm, and should be an integral part of any method and of the way it is taught.

The usual context for configuration is a project. As the project starts key members of the project team configure the method to their needs and circumstances. The early and central question is, "What work products are needed on this project?" not just, of course, what is to be delivered, but also what is to be produced along the way. Tailoring or configuration work is done early during the proposal phase and revised when the project starts. If there is a well-established matching engagement model, the simplest approach is to amend the associated list of WPDs. Work products are usually selected or deselected in groups. Dependency diagrams help people visualize the impact of their decisions.

Figure 3 shows the form of a spreadsheet that can be used to record the results of the configuration. The spreadsheet starts from a standard list of WPDs, either the full list or the WPDs associated with an engagement model. Some groups also use a standard

Architectural thinking and modeling with the Architects' Workbench

S. Abrams
B. Bloom
P. Keyser
D. Kimelman
E. Nelson
W. Neuberger
T. Roth
I. Simmonds
S. Tang
J. Vlissides

Collecting and organizing all of the architectural information for a system is a challenge faced by information technology (IT) architects. Transforming that information into models of a viable architecture and keeping associated work products consistent and up to date is an even greater challenge. Despite this, model-centric architectural methods are not as widely adopted or as closely followed as they could be, partly due to a lack of appropriate tools. The Architects' Workbench (AWB) is a prototype tool that addresses these problems and supports the creative process of architectural thinking and modeling. This paper presents key AWB innovations and discusses how their design was motivated by insights into architectural work and feedback from IT architects. We describe the design of AWB itself as a metamodel-driven and method-based tool, and we report on experience from the use of AWB in production environments.

INTRODUCTION

IT architects are faced with a formidable information management challenge as they design systems to address customer needs. For example, in designing a new Web application that will serve as the single point from which a customer of a large financial institution can access all of his or her financial information, it is often necessary to integrate hundreds of legacy systems dealing with various financial instruments, dozens of databases storing account information, and myriad rules and constraints. The challenge for the IT architect is to organize and analyze initial descriptions of customer needs as well as the existing IT environment, and to design an appropriate solution. This activity often involves progressively formalizing informa-

tion and building up architectural models. The resulting models and design rationale must then be incorporated into a number of overlapping work product documents for a variety of stakeholders, and these documents must be kept up to date and consistent as the system evolves.

IT architecture

To better understand this challenge, we consider IT architecture in greater detail. Many definitions of IT

© Copyright 2006 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information service systems. Permission to republish any other portion of the paper must be obtained from the Editor. 0018-8670/06/45.00 © 2006 IBM

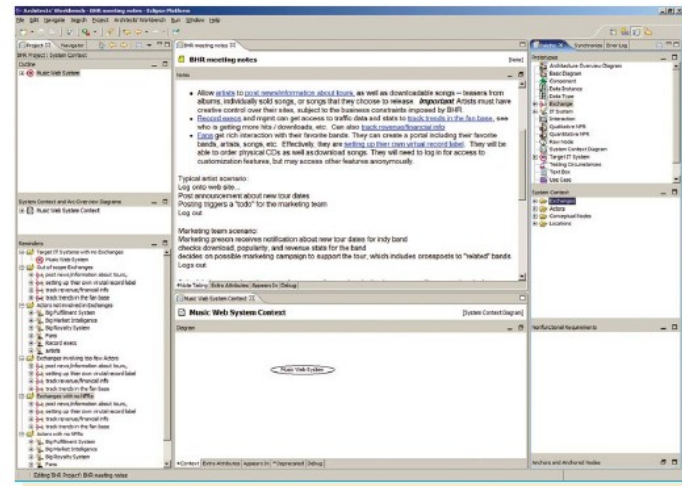


Figure 1
AWB configured for system context, showing reminders

tion and building up architectural models. The resulting models and design rationale must then be incorporated into a number of overlapping work product documents for a variety of stakeholders, and these documents must be kept up to date and consistent as the system evolves.

AWB provides many ways of viewing and interacting with models. At any time, the architect can begin to visualize relationships among elements by using the basic diagram. As understanding of the solution evolves, the user can refactor the model, refining the elements involved and specifying the relationships among them and elaborating the details of the model in customized diagram editors (such as those for system context, component interactions, and operational modeling), while maintaining connections among the model elements and traceability back to less-structured information. The user can move between textual, tree-based, tabular, and graphical representations of the model and can generate work products to check the state of the architectural documents. The documents can

be edited in WYSIWYG ("what you see is what you get") views, and the changes are immediately reflected back to the underlying model. As the user switches among the various activities needed to complete the architecture, AWB responds with task-appropriate guidance and presentations of the model.

We have briefly outlined many of AWB's capabilities. In the following sections, we will discuss in more depth some of its key features, focusing on how they address the problems described previously.

AWB FEATURES



Several principles guided the design of many AWB features. We wanted AWB, where possible, to support multiple projections (or views) of a single, integrated model, rather than a number of distinct models that users would have to keep consistent manually. We wanted to support the fundamental thought processes used in the practice of IT

Secure | <https://web.archive.org/web/20121203213741/http://epf.eclipse.org/wikis/openup/>       

 OpenUP [Glossary](#) | [Feedback](#) | [About](#)  [Print](#)

Where am I |  Tree Sets | 

Team

-  [Introduction to OpenUP](#)
-  [Getting Started](#)
-  [Understanding OpenUP](#)
-  [Basic Process Concepts](#)
-  [Practice](#)
-  [Resources for contributing to the Eclipse Process Framework](#)
-  [Resources for Customizing Methods](#)
-  [Delivery Processes](#)
-  [Practices](#)
-  [Roles](#)
-  [Work Products](#)
-  [Tasks](#)
-  [Guidance](#)
-  [Tools](#)
-  [Release Info](#)


[Getting Started](#) > [Understanding OpenUP](#)

Understanding OpenUP



OpenUP is a lean Unified Process that applies iterative and incremental approaches within a structured lifecycle. OpenUP embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. It is a tools-agnostic, low-ceremony process that can be extended to address a broad variety of project types.

 [Expand All Sections](#)

 [Collapse All Sections](#)

Relationships

Contents

- [OpenUP Roadmap](#)
- [Who Should Use OpenUP](#)
- [Core Principles](#)
- [Minimal, Complete, and Extensible](#)

 [Back to top](#)

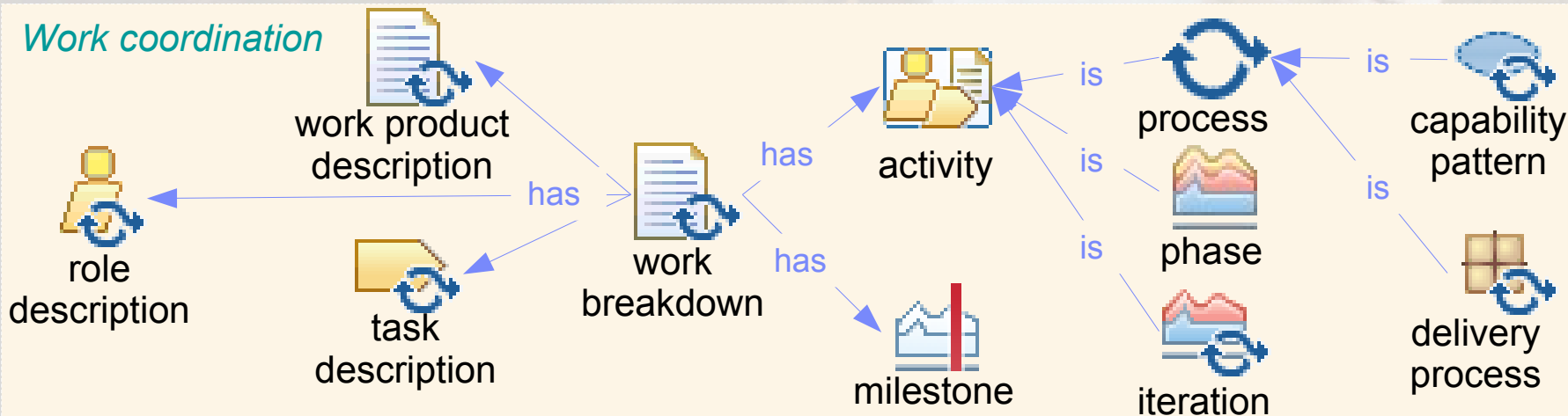
This program and the accompanying materials are made available under the [Eclipse Public License V1.0](#), which accompanies this distribution.

EPF Copyright.

Work content



Work coordination



2018 SIGCHI AWARDS

SIGCHI LIFETIME SERVICE AWARD

JOHN C. THOMAS

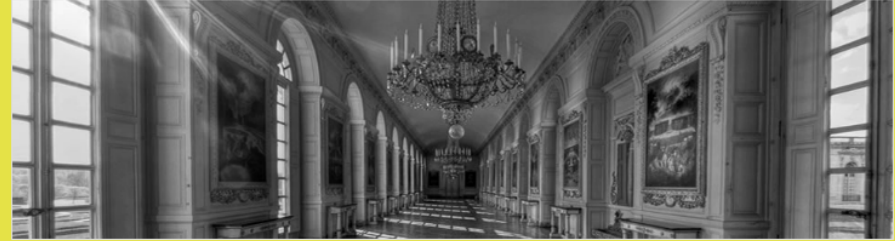


John C. Thomas has been involved in service to SIGCHI since the original 1982 conference in Gaithersburg, where he served on the organizing committee and as publications co-chair. John has served in numerous major roles in the CHI and other SIGCHI conferences, including co-chairing the CHI conference in 1991, and serving on the SIGCHI EC, as a VP at Large and as Adjunct Chair for Mentoring (2009-2015), as well as on the EC's Conference Management Committee from 1992-1995. He regularly serves in mentoring roles, including in doctoral consortia. John has also served on conference committees for many SIGCHI-affiliated conferences, including CSCW, Creativity and Computing, GROUP, Interact, and the ACM Conference on Universal Usability. He has served on numerous technical committees, including papers, workshops, panels, posters, case studies and the Interactive Experience. He also served as mentoring chair for CHI for two years and chaired mentoring of reviews for CSCW.

His track record of championing HCI issues in industry is extensive. For example, a couple years after joining IBM Research, and leading research efforts in HCI, he explained to top Research management about the importance of fostering the study of human computer interaction for the continued growth of the computer industry. In addition to research projects on natural language, query languages, speech synthesis, and the psychology of design at IBM Research, he also spent two years in the IBM Office of the Chief Scientist extensively selling the importance of human factors in computing systems within IBM resulting in doubling the number of relevant professionals, changing the development process, instituting scholarship and grants in the field of HCI, and visiting all the world-wide IBM development labs to evaluate, improve, or institute usability labs. In order to accomplish this, he found common cause with allies in other IBM functions including Marketing, Sales, Product Assurance, and Design.

After leaving IBM, he managed the Artificial Intelligence Lab at NYNEX where he convinced management to add a group in Human Computer Interaction and institute a partnership with the University of Colorado HCI group. In his role as Executive Director, he was partly responsible for the program for two major annual technical conferences in telecommunications, the Eastern Communications Forum and the National Communications Forum. Here, he began tracks in usability that introduced hundreds of engineers to the

petersironwood ~ Finding, formulating
and solving life's frustrations.



02 Friday Mar
2018

Pattern Language Overview

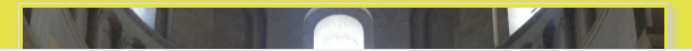
POSTED BY PETERSIRONWOOD IN MANAGEMENT, PSYCHOLOGY, UNCATEGORIZED

≈ 2 COMMENTS

Pattern Language Overview

Prolog/Acknowledgement:

An old story recounts a person walking down a path and noticing two workers laying stones and cementing them into place. The walker noticed that one of the workers walked with a bounce in their step and a whistle on their lips. The other worker, however, trudged from stone pile to wall with a scowl. The walker imagined that perhaps the disgruntled worker was being paid less or was ill or had suffered a recent tragedy. Because the walker was familiar with the "Iroquois Rule of Six" however, they knew that it would be better to test their hypotheses than make assumptions about the reasons. He asked the disgruntled worker what they were doing. "Isn't it obvious? I have to take these stones from the pile over there and lay them in that wall over there and cement them in place." When asked the same question, the worker with the sunny disposition answered, "Isn't it obvious? I'm building a cathedral!"



Tags

Business, collaboratoin,
competition,
cooperation, innovation,
pattern language,
teamwork

Agenda

1. 1964 → 1999 → 2012:
Synthesis of Form → OOPSLA 1996 → Battle (Eishin)
2. 1993 → 2002 → 2006 → 2010:
Hillside Group, IGS Method, AWB, Eclipse
3. 2014 → ...
Wicked Messes, Service Systems Thinking

2016/10/28 Pattern Manual for Service Systems Thinking

Submitted by davidng on Thu, 09/29/2016 - 21:00

Authors

David Ing

Abstract

What is properly required to take the learning on generative pattern languages from the built environment and software development communities, to a world of service system thinking?

This position paper winds back to early days of Center for Environmental Studies, and presents an alternative view on the 1968 Multi-Service Center work, informed by 21st century developments in service systems science. The conventional format for a pattern language has settled into a three-part rule of relations between context, problem and solution. An alternative format of (i) voices on issues (who + what), (ii) affording value(s) (how + why), and (iii) spatio-temporal frames (where + when) is proposed, with a straw man example.

Methods from the 1985 Eishin campus project, published in 2012, are compared against practices that have become common in agile development.

The conceptual shifts from built environment to service systems thinking are expressed as (i) amplifications, (ii) rephilosophizations, and (iii) reinterpretations. The generation and legitimization of pattern languages is considered across a community, with a shift from publishing in books on paper to collaborating with online technologies such as wiki.

At the 2014 PLoP and the 2015 PURPLSOC conferences, the idea of extending the pattern language for environment structure into a new domain of service systems thinking was introduced. In 2016, this idea has been further developed as a baseline for further discussion

Citation

David Ing, "Pattern Manual for Service Systems Thinking: A proposal for discussion", *Proceedings of the 2016 International PUARL Conference*.

Content

- [\[view/download article as DOCX\]](#) (1.4 MB, originally edited in LibreOffice 5)
- [\[view/download article as PDF\]](#) (2.2 MB)

Coevolving Innovations

... In Business Organizations and Information Technologies

Commons front page ...

Home

Publications

Submitted by davidng on Thu, 10/27/2016 - 22:31

Publication Date	Publication Title	Author(s)	Form
October 2016	"Pattern Manual for Service Systems Thinking: A proposal for discussion" [view abstract and article]	David Ing	article in review for the 2016 International Conference
October 2016	"Curriculum Making for Trito Learning: Wayfaring along a meshwork of systems thinking" [view abstract and presentation slides]	David Ing	presentation at RSD5 Relating Systems Thinking Design
	"Service Systems and the Svstems Scienes"		presentation at Wuhan

Index	∨	×
1. Introduction		2
2. Extending an Alexandrian example to services illustrates differences	∨	3
2.1. Alexandrian format follows context, system of forces, configuration		3
2.2. Try services as voices on issues, affording value(s), spatio-temporal frames		7
2.3. The Alexandrian and proposed formats can be mapped for comparison		11
3. Methods associated with pattern language have clarified since 1973	∨	16
3.1. Alexandrian methods include pattern language, budget and reality of the land		16
3.2. Try services with user stories, scoping, reviewing iteratively		18
3.3. The challenge of System-B to System-A is similar to waterfall to agile		19
4. A new format amplifies, rephilosophizes and reinterprets prior doxa	∨	20
4.1. Amplifications include shared meaning, systems thinking, method + process		20
4.1.1. Shared meaning on the situated		20
4.1.2. Systems thinking and complexity		21
4.1.3. Method content + development process		23
4.2. Rephilosophizations include alternative stable states, journeying and meshwork		25
4.2.1. From structuralism to alternative stable states		25
4.2.2. From dwelling to journeying		28
4.2.3. From semi-lattice to meshwork		31
4.3. Reinterpretations include issue-seeking, interactive value and wayfaring		33
4.3.1. From problem-solving to issue-seeking		33
4.3.2. From quality-wholeness to interactive value		36
4.3.3. From anti-patterns to wayfaring		38
5. Pattern languages are generated and legitimized in communities		40
6. References		41
7. About the author:		47

Pattern Manual for Service Systems Thinking: A proposal for discussion

David Ing, Aalto University and the International Society for the Systems Sciences, coevolving@gmail.com

Abstract:

What is properly required to take the learning on generative pattern languages from the built environment and software development communities, to a world of service system thinking?

This position paper winds back to early days of Center for Environmental Studies, and presents an alternative view on the 1968 Multi-Service Center work, informed by 21st century developments in service systems science. The conventional format for a pattern language has settled into a three-part rule of relations between context, problem and solution. An alternative format of (i) voices on issues (who + what), (ii) affording value(s) (how + why), and (iii) spatio-temporal frames (where + when) is proposed, with a straw man example.

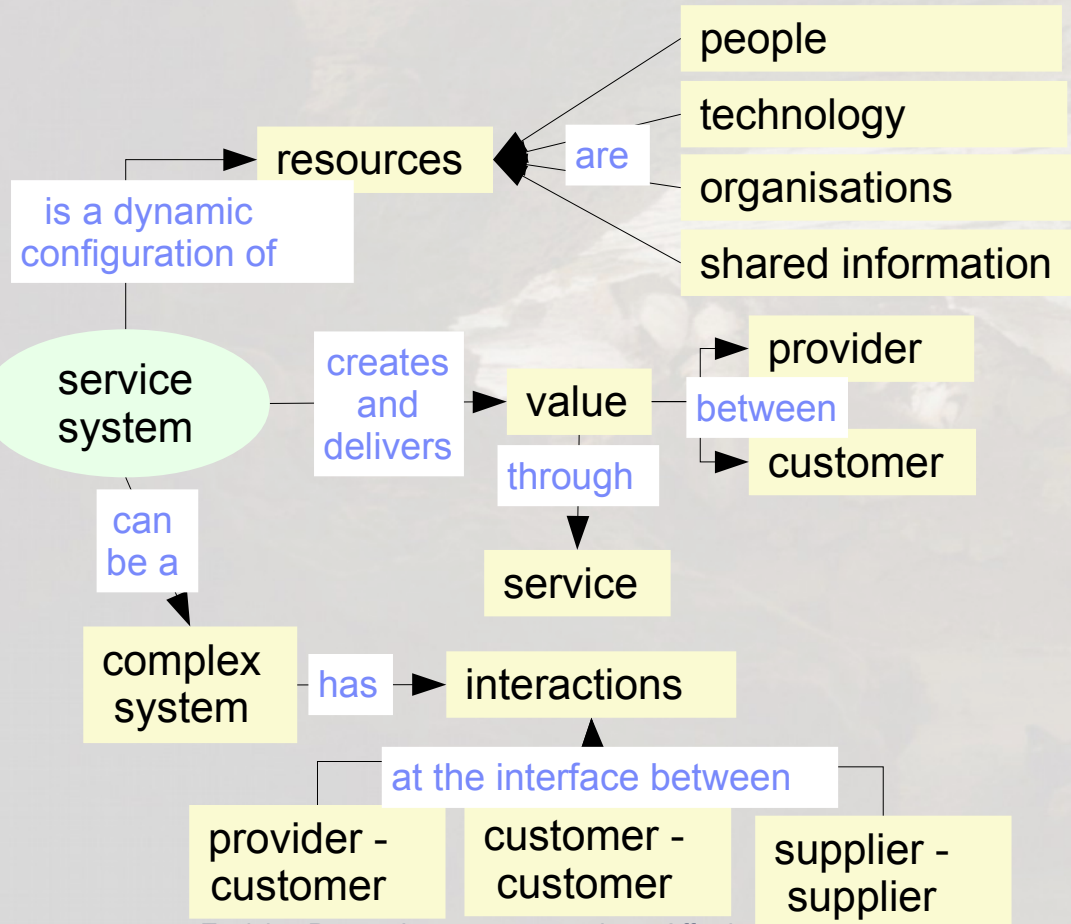
Methods from the 1985 Eishin campus project, published in 2012, are compared against practices that have become common in agile development.

The conceptual shifts from built environment to service systems thinking are expressed as (i) amplifications, (ii) rephilosophizations, and (iii) reinterpretations. The generation and legitimization of pattern languages is considered across a community, with a shift from publishing in books on paper to collaborating with online technologies such as wiki.

At the 2014 PLoP and the 2015 PURPLSOC conferences, the idea of extending the pattern language for environment structure into a new domain of service systems thinking was introduced. In 2016, this idea has been further developed as a baseline for further discussion.

Keywords: *service systems; systems thinking; issue-seeking; interactive value; wayfaring*

After 2007, service systems have been recognized as the largest part of developed economies globally



A **service system** can be defined as a **dynamic configuration of resources** (people, technology, organisations and shared information) that **creates and delivers value** between the provider and the customer through service.

In many cases, a service system is a **complex system** in that configurations of resources interact in a non-linear way.

Primary interactions take place at the interface between the provider and the customer.

However, with the advent of ICT, customer-to-customer and supplier-to-supplier **interactions** have also become prevalent.

These complex interactions create a system whose behaviour is difficult to explain and predict.

(IfM and IBM, 2008, p. 6)



Coevolving Innovations

... in Business Organizations and Information Technologies

Christopher Alexander, Horst Rittel, C. West Churchman

At U.C. Berkeley in the 1960s, [Christopher Alexander](#), [Horst Rittel](#) and [C. West Churchman](#) could have had lunch together. While disciplinary thinking might lead novices to focus only on each of [pattern language](#), [wicked problems](#) and [the systems approach](#), there are ties (as well as domain-specific distinctions) between the schools.



Circa 1968-1970: Christopher Alexander, Horst Rittel, West Churchman

Recent Posts

- [Christopher Alexander, Horst Rittel, C. West Churchman](#)
- [Open Innovation Learning and Open Data](#)
- [Learning data science, hands-on](#)
- [Innovation Learning and Open Sourcing: IoT + Cloud + Cognitive](#)
- [Acts of representation with systems thinking \(OCADU 2017/03\)](#)
- [Service Systems Thinking, with Generative Pattern Language \(Metropolia 2016/12\)](#)

Subscribe via e-mail

Subscribe

Tweets by @daviding



David Ing
@daviding

Anshansicun: Whimsically residential area,... [bit.ly/2jU](#)



At Berkeley: Churchman, Rittel and Alexander taught in 1960-1970s

C. West Churchman (1913-2004)

- 1957 joined Berkeley, graduate programs in OR at School of Business Administration
- 1964-1970 Associate Director and Research Philosopher, Space Sciences Laboratory
- 1981-1994 retired, taught Peace & Conflict Studies

Horst Rittel (1930-1990)

- 1963 Berkeley College of Environmental Design
- 1974 both Berkeley and University of Stuttgart

Christopher Alexander (1936 -)

- 1963 Berkeley College of Environmental Design
- 1967 cofounder Center for Environmental Structure
- 1998 retired from university

Both Alexander and Rittel were part of what at the time was called the 'design methods' movement in architecture, worked and taught in the same building, and did talk and were seen walking off to have lunch together. Churchman was teaching in the Business School a few minutes down on the way to the center of campus.

- *Thor Mann*
(posted April 17, 2017)

“Systems Generating Systems”, Alexander (1968)

1. There are two ideas hidden in the word system: the **idea of a system as a whole** and the **idea of a generating system**.
2. A **system as a whole** is not an object but a way of looking at an object. It focuses on some holistic property which can only be understood as a product of interaction among parts.
3. A **generating system** is not a view of a single thing. It is a kit of parts, with rules about the way these parts may be combined.
4. Almost every ‘system as a whole’ is generated by a ‘generating system’. If we wish to make things which function as ‘wholes’ we shall have to **invent generating systems to create them**. [...]

In a properly functioning building, the **building and the people in it together form a whole**: a social, human whole. The building systems which have so far been created **do not** in this sense **generate wholes at all**. (Alexander, 1968, p. 605)

“Dilemmas in a General Theory of Planning”, (Rittel + Weber, 1973)

The kinds of problems that planners deal with -- societal problems – are inherently different from the problems that scientists and perhaps some classes of engineers deal with.

Planning problems are inherently wicked.

The problems that scientists and engineers have usually focused upon are mostly "**tame**" or "**benign**" ones.

As an example, consider a problem of mathematics, such as solving an equation; or the task of an organic chemist in analyzing the structure of some unknown compound; or that of the chessplayer attempting to accomplish checkmate in five moves.

For each the mission is clear.

It is clear, in turn, whether or not the **problems** have been solved.

Wicked problems, in contrast, have neither of these clarifying traits; and they include nearly all public policy issues – whether the **question** concerns the location of a freeway, the adjustment of a tax rate, the modification of school curricula, or the confrontation of crime.

There are at least **ten distinguishing properties** of planning-type problems, i.e. wicked ones ... We use the term “wicked” in a meaning akin to that of “malignant” (in contrast to “benign”) or “vicious” (like a circle) or “tricky” (like a leprechaun) or “aggressive” (like a lion, in contrast to the docility of a lamb).

Horst WJ Rittel, and Melvin M. Webber. 1973. “Dilemmas in a General Theory of Planning.” *Policy Sciences* 4 (2): 155–169. <https://doi.org/10.1007/BF01405730>.

Ten distinguishing properties of planning-type (wicked) problems (#1 - #5)

	Tame (benign) problems	Wicked (malignant) problems
1.	An exhaustive formulation can be stated containing all the information needed for understanding and solving the problem	There is no definitive formulation of a wicked problem.
2.	There are criteria that tell when <i>the</i> or a solution has been found .	Wicked problems have no stopping rule .
3.	There are conventionalized criteria for objectively deciding whether the offered solution is correct or false.	Solutions to wicked problems are not true-or-false, but good or bad .
4.	One can determine on the spot how good a solution-attempt has been.	There is no immediate and no ultimate test of a solution to a wicked problem
5.	The problem-solver can try various experimental runs without penalty.	Every solution to a wicked problem is a " one-shot operation "; because there is no opportunity to learn by trial and error, every attempt counts significantly.

Ten distinguishing properties of planning-type (wicked) problems (#6 - #10)

	<i>Tame (benign) problems</i>	<i>Wicked (malignant) problems</i>
6.	There are criteria which enable proof that all solutions have been identified and considered .	Wicked problems do not have an enumerable (or an exhaustively describable) set of potential solutions , nor is there a well-described.
7.	There might be important classes to know which type of solution to apply.	Every wicked problem is essentially unique .
8.	Small steps lead to overall improvement, through incrementalism .	Every wicked problem can be considered to be a symptom of another problem .
9.	Rules or procedures can determine the “correct” explanation or combination of them.	The existence of a discrepancy representing a wicked problem can be explained in numerous ways . The choice of explanation determines the nature of the problem's resolution.
10.	Science does not blame for postulating hypotheses that are later refuted .	The social planner has no right to be wrong (i.e., planners are liable for the consequences of the actions they generate)

“The Systems Approach and Its Enemies”, (Churchman, 1979)

Common to all **these enemies** is that **none of them accepts the reality of the "whole system"**: we do not exist in such a system. Furthermore, in the case of **morality, religion, and aesthetics**, at least a part of our reality as human is not "in" any system, and yet it plays a central role in our lives.

To me these enemies provide a powerful way of learning about the systems approach, precisely because they **enable the rational mind to step outside itself and to observe itself** (from the vantage point of the enemies). [....]

We must **face the reality** that the enemies offer: what's really happening in the human world is politics, or morality, or religion, or aesthetics. This confrontation with reality is totally different from the rational approach, because **the reality of the enemies cannot be conceptualized, approximated, or measured** (Churchman, 1979, pp. 24–53).

A *mess* (or *problématique*) is a system of problems

The **optimal solution** of a model is not an optimal solution of a problem unless the model is a **perfect representation** of the problem. Therefore, in testing a model and evaluating solutions derived from it, the model itself should not be used to determine the relevant comparative performance measures.

All **models** are **simplifications of reality**. If this were not the case, their **usefulness** would be diminished. Therefore, it is critical to determine how well they represent reality.

... what the French call a ***problématique*** and I call a ***mess*** ... is a **complex and highly dynamic system of interacting problems.**

Problems are elements abstracted from messes; therefore, problems are to messes what atoms are to planets. There is an important systems principle, familiar to all of you, that applies to messes and problems: that **the sum of the optimal solutions to each component problem considered separately is not an optimal solution to the mess**. This follows from the fact that the behavior of the mess depends more on how the solutions to its component problems interact than on how they act independently of each other.

The treatment of messes requires **more than problem solving**; it requires planning. Planning should consist of the design of a desirable future and invention or selection of ways of getting there. Therefore, it is **more** a matter of **synthesis**, of **design** and **invention** than it is of analysis, of programming and budgeting.

Ackoff, Russell L. 1977. "Optimization + Objectivity = Optout." *European Journal of Operational Research* 1 (1): 1–7.
[https://doi.org/10.1016/S0377-2217\(77\)81003-5](https://doi.org/10.1016/S0377-2217(77)81003-5).

Pattern language is *not* for wicked problems!

coevolving.com/blogs/index.php/archive/exploring-the-context-of-pattern-languages/

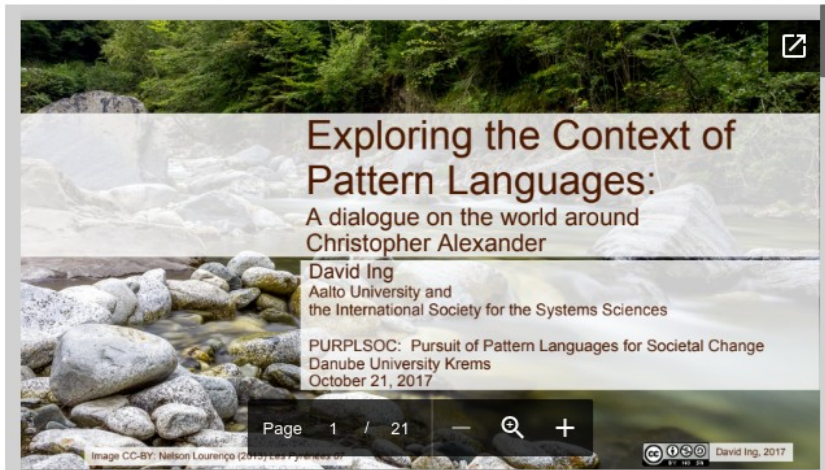
Exploring the Context of Pattern Languages

Pattern language is not for wicked problems, said [Max Jacobson](#), coauthor with [Christopher Alexander](#) of the 1977 *A Pattern Language: Towns, Building, Construction*. In addition, the conventional definition of an Alexandrian pattern as “a solution to a problem in context” when applied to social change might better use the term “intervention”, rather than “solution”.

These are two of the major ideas that emerged at [Purplsoc 2017](#) conference last October. A 90-minute workshop was run in parallel with other breakouts.

For about the first hour, vocal participants included Max Jacobson (who had given a plenary talk on “A Building is not a Turkish Carpet”), [Christian Kohls](#) (who gave a plenary talk on “Patterns for Creative Space”) and [Peter Baumgartner](#) (one of the Purplsoc chairs).

As an impetus to discussion, we stepped through slides that had been posted on the [Coevolving Commons](#).



For people who would like the next-best experience to being there, the slides have now been matched up with the digital audio recording, for viewing as a [web video](#).



Complicated systems are rare; complex systems are the norm

The following is possibly the golden rule for distinguishing 'complex' from 'complicated' problems and systems.

Complicated problems

originate from **causes** that can be **individually distinguished**; they can be **addressed piece-by-piece**; for **each input** to the system there is a **proportionate output**; the relevant systems can be **controlled** and the problems they present admit **permanent solutions**.

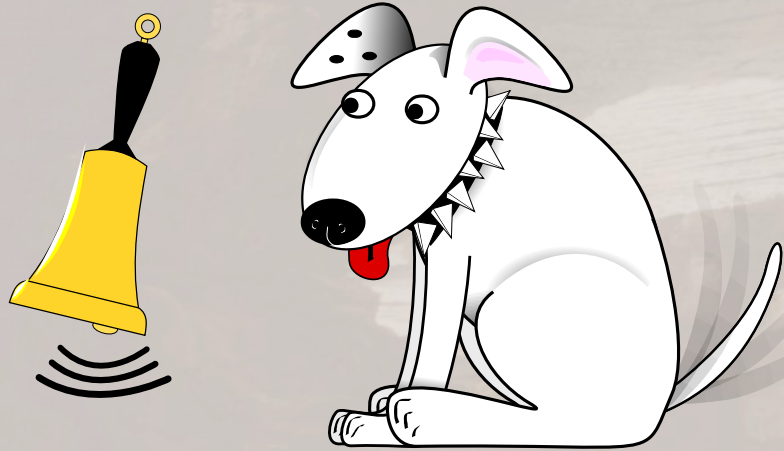
... complex problems and systems

result from networks of **multiple interacting causes** that cannot be individually distinguished; must be **addressed as entire systems**, that is they cannot be addressed in a piecemeal way; they are such that **small inputs** may result in **disproportionate effects**; the problems they present **cannot be solved once and for ever**, but require to be systematically managed and typically **any intervention merges into new problems** as a result of the interventions dealing with them; and the **relevant systems cannot be controlled ...**

... decision-makers ask their consultants ... to **treat complex problems as if they were complicated** ones. Complexity and the nature of contemporary science show that the claim to 'solve' (complex) problems is often ungrounded. **'Learning to dance'** with a complex system is definitely **different from 'solving' the problems** arising from it.

Ask Not What's Inside Your Head, but What Your Head's Inside of

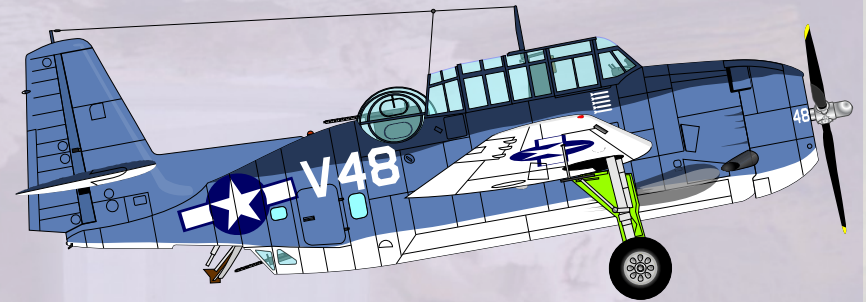
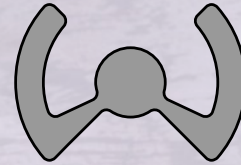
Stimulus – Response (Behavioral Psychology)



[In the 1950] psychophysics of perception ... "givens" in the light to the eye could not support perceptual phenomena, but only elementary experiences such as sensations. [...] Succinctly put, the psycho-physical program was ... traditional in considering perception to be a set of responses to presented stimuli (albeit "higher order" stimuli).

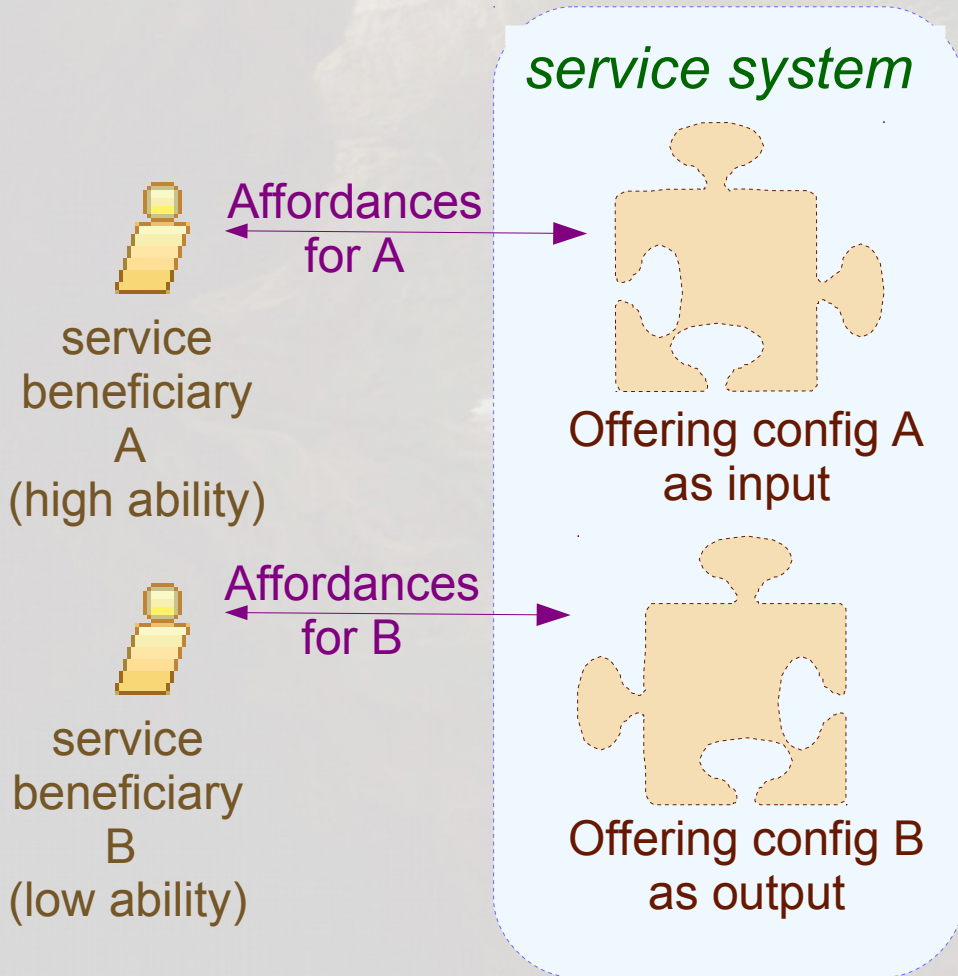
William M. Mace 1977. "James J. Gibson's Strategy for Perceiving: Ask Not What's inside Your Head, but What Your Head's inside of." In *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, edited by Robert Shaw and John Bransford, 43–65.

Ecological Approach to Perception



Over the last 10-15 years [James J. Gibson] has tried to develop enough theory ... to demonstrate that direct perception is indeed plausible even if hordes of difficult details remain to be worked out. The ... analysis of the optic array, stimulus organization, and the functional organization of perceptual systems are what Gibson offers points to as radical features

Affordances are relational in an ecological perception



The term **affordance** refers to whatever it is about **the environment** that **contributes** to the kind of **interaction** that occurs. [...]

An affordance relates attributes of something in the environment to an interactive activity by an agent who has some ability, and an ability relates attributes of an agent to an interactive activity with something in the environment that has some affordance.

The relativity of affordances and abilities is fundamental. Neither an affordance nor an ability is specifiable in the absence of specifying the other.

James G. Greeno 1994. "Gibson's Affordances."
Psychological Review 101 (2): 336–342.

Lifelines co-respond with habit, agencing, and attentionality



Habit, rather than volition:

I become my walking, and that my walking walks me. I am there, inside of it, animated by its rhythm. And with every step I am not so much changed as modified, in the sense not of transition from one state to another but of perpetual renewal. [p. 16]

Ingold, Tim. 2017. "On Human Correspondence." *Journal of the Royal Anthropological Institute* 23 (1):9–27. <https://doi.org/10.1111/1467-9655.12541>.

Images from Flickr: "Sandy walks on sunny evenings" CC-BY 2010 Satish Krishnamurthy; "Jump Together" CC-BY 2011 Stephanie Evanoff; "IMG 2012" CC-BY 2013 Ondrej Tachovsky

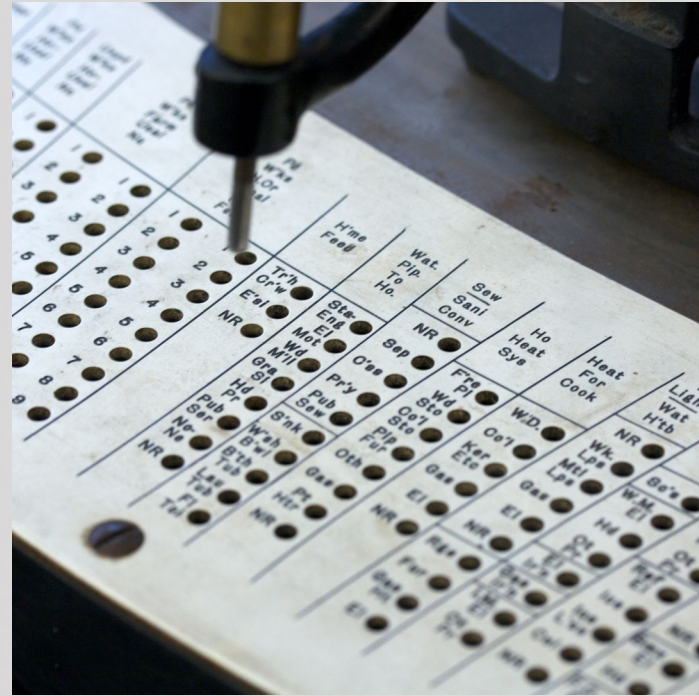
Agencing, rather than agency:

Interaction goes back and forth as agents, facing each other on opposite banks of the river, trade messages, missiles, and merchandise. But to *correspond*, in my terms, is to join with the swimmer in the midstream. It is a matter not of taking sides but of going along. [p. 18]

Attentionality, rather than intentionality:

Walking calls for the pedestrian's continual responsiveness to the terrain, the path, and the elements. To respond, he must attend to these things as he goes along, joining or participating with them in his own movements. [p. 19]

Infrastructure advances: Tabulating → Automating → Co-responding



The Tabulating Era (1900s-1940s)

- Single purpose mechanical systems



The Programming Era (1900s-today)

- If / then logic and loops, instructions coded in software



The Cognitive Era (2011 →)

- Man-machine symbiosis in cooperative interactions (Licklider)

Kelly, John E. 2015. "Computing, Cognition and the Future of Knowing." *IBM Research and Solutions Portfolio*. Somers, NY: IBM. https://www.research.ibm.com/software/IBMResearch/multimedia/Computing_Cognition_WhitePaper.pdf.

Images from Flickr: "Hollerith Census Machine pantograph" CC-BY 2008 Marcin Wichary; "Tarek was pleased with the prevalence of Bank of Montreal ATMs" CC-BY 2008 Marchin Wichary; "Pokemon Go" CC-BY 2016 Paintimpact

Alexandrian format mapped to proposed service systems thinking

Format for service systems thinking

(i) Pattern label	An interaction phrased as a present participle
(ii) Voices on issues (who and what)	Archetypal roles of stakeholders, with concerns and interests posed as questions
(iii) Affording value(s) (how and why)	Objects and/or events that enable modes of practised capacities for independent or mutual action
(iv) Spatio-temporal frames (where and when)	Occasions at which dwelling in issues and affordances are salient and at hand
(v) Containing systems (slower and larger)	Constraining conditions in which the pattern operates, potentially where multi-issue messes are dissolved
(vi) Contained systems (faster and smaller)	Opportunistic conditions which the pattern contains, potentially allowing ad hoc resolving of a specific issue at hand

Try who+what, how+why, where+when, containing, contained

(i) Pattern label	Tapping into the grapevine ◇◇◇	Signing in for services ◇◇◇	Minding children ◇◇◇
(ii) Voices on issues (who and what)	(a) For a client, what jobs and training are available? (b) For a neighbour, in what ways can we share and update community news?	(a) For a client, what services are available to me, now and on appointment? (b) For a parent, what do I do with my kids while I'm busy? (c) For a child, what can I do while my parent is at the MSC?	
(iii) Affording value(s) (how and why)	Displaying up-to-date news and local information, so that individuals can know ways to independently act. Adding, revising and moderating community contributions so that individual and authoritative viewpoints are balanced.	Matching client needs with MSC resources, so that holistic treatments are received. Triaging and scheduling so that urgent cases are prioritized, and wait times are tolerable	Leaving a child at a supervised play area so that whereabouts are known. Availing distractions for toddlers through teens, so that coming with parents is less of a chore
(iv) Spatio-temporal frames (where and when)	Access to information onsite MSC for clients who don't have devices, and on the open Internet for the public ◇◇◇	On demand lookups of trending and prior MSC busy and slow periods transparently visible onsite and on the Internet, enabling clients to adjust and/or rebook ◇◇◇	Facilities and programs are known both to children and parents in advance of appointments ◇◇◇
(v) Containing systems (slower and larger)	For municipal, regional and national agencies, are community health and social services in their jurisdictions well provide?		For extended family, schools and community workers, what personal responsibilities inhibit service engagement?
vi) Contained systems (faster and smaller)	For neighbours in mutual support, friends and family ties, who should know about news?	For friends or assistants speaking on behalf or interpreting for a client, is the situation understood?	For other parents at the MSC at the same time, would you look after my kids like I look after yours?

Minding children: who+what, how+why, where+when, containing, contained

(i) Pattern label	Minding children
	◇ ◇ ◇
(ii) Voices on issues (who and what)	(a) For a client, what services are available to me, now and on appointment? (b) For a parent, what do I do with my kids while I'm busy? (c) For a child, what can I do while my parent is at the MSC?
(iii) Affording value(s) (how and why)	Leaving a child at a supervised play area so that whereabouts are known. Availing distractions for toddlers through teens, so that coming with parents is less of a chore
(iv) Spatio-temporal frames (where and when)	Facilities and programs are known both to children and parents in advance of appointments
	◇ ◇ ◇
(v) Containing systems (slower and larger)	For extended family, schools and community workers, what personal responsibilities inhibit service engagement?
(vi) Contained systems (faster and smaller)	For other parents at the MSC at the same time, would you look after my kids like I look after yours?

newer, older

Welcome Visitors

Welcome to the [Smallest Federated Wiki](#). This page was first drafted Sunday, June 26th, 2011. The pages on this particular site have been edited to describe how to get things done on many of the federated sites.

Featured Sites

[sites.fed.wiki.org](#)

A catalog of federated wiki sites with domain names for page titles and brief descriptions tuned to look good in search results. Know your federation.

Topic Based Subsets

We pick topics that have been of lasting interest and subset them into their own federated wiki sites. We've built this feature into c2 wiki's [Subset Wiki](#) bridge and only use it here. [github](#)

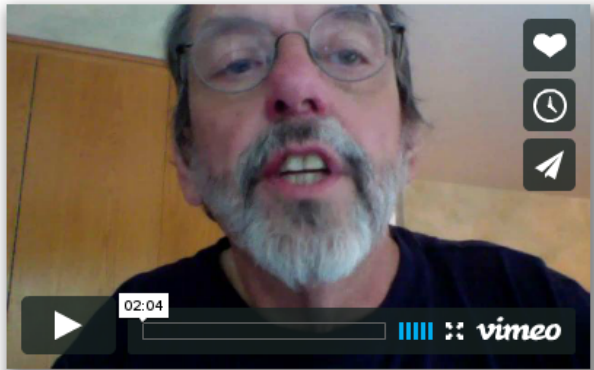
Learn More

Read a little bit of [How To Wiki](#). Then move on to our [Sandbox](#) and give your new knowledge a workout. Still confused? Look for answers in our [Frequently Asked Questions](#), updates in [Recent Changes](#).

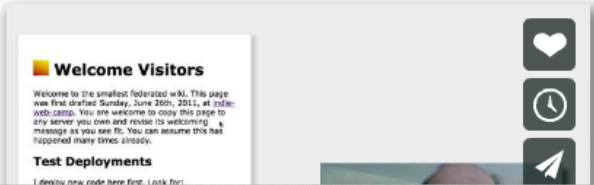
Grid of icons for editing and navigation, including arrows, plus signs, and a cursor icon.

Smallest Federated Wiki

Our new wiki innovates three ways. It shares through federation, composes by refactoring and wraps data with visualization. Follow our open development on GitHub or just watch our work in progress videos here.



We introduce the parts of a Federated Wiki page. The "story" is a collection of paragraphs and paragraph like items. The "journal" collects story edits. Should you take my page and edit it as yours, I can see what you've done and may decide to take your edits as my own.



fedwiki / wiki

<> Code Issues 29 Pull requests 1 Projects 0

Federated Wiki - node server as npm package <https://npmjs.org/package/federated-wiki>

ReadMe.md

Federated Wiki (Node.js server)

Here we have a new wiki repository, and package, which only exports a `wiki-client`, and plug-ins) and start the server.

Using Federated Wiki

Learn [how to wiki](#) by reading [fed.wiki.org](#)

Running your own Server

The quickest way to set up wiki on your local machine is to install

```
$ npm install -g wiki
$ wiki
```

Visit localhost:3000 to see your wiki. If you choose a host visible to the internet, you may want to use a tunneling service.

Updating the Server Software

From time to time some of the packages that makeup the wiki server will be updated. To update any of the wiki packages, run:

Generative Pattern Language

While the label "pattern language" has been appropriated for a variety of contexts, the label of "generative pattern language" can be used for the "purer" thinking originating from the Center for Environmental Structure at U.C. Berkeley.

Christopher Alexander and his colleagues have a significant body of artifacts since the formation of the CES in 1967.

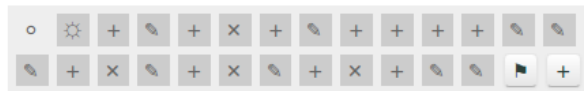
Pattern Manual (1967) is a charter for the CES.

A Pattern Language Which Generates Multi-Service Centers (1968) demonstrates how a pattern language could become instantiated differently for a variety of sites and circumstances.

"*Systems Generating Systems (1968)*" articulates the ties between a pattern language and systems thinking.

The Battle for Life and Beauty of the Earth (2012) is a history of a development project for the Eishin campus in Japan, demonstrating the CES vision from start to finish.

The variety of [Current Applications of Pattern Languages](#) often don't reflect the full vision of generativity.



A Pattern Language Which Generates Multi-Service Centers (1968)

Christopher Alexander, Sara Ishikawa, and Murray Silverstein. 1968. *A Pattern Language Which Generates Multi-Service Centers*. Center for Environmental Structure. [preview on Google Books](#)

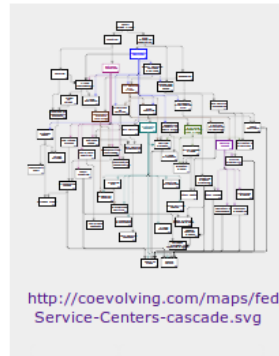
[Introduction \(Alexander et. al. 1968\)](#)

[I. Summaries of 64 Patterns \(Alexander et al. 1968\)](#)

[II. The Idea of a Pattern Language \(Alexander et al. 1968\)](#)

[III. Eight Buildings Generated by the Pattern Language \(Alexander et al. 1968\)](#)

[IV. The Language \(Alexander et. al 1968\)](#)



This page is part of [Historic Works on Generative Pattern Languages](#)



Summaries of 64 Patterns (Alexander et al. 1968)

Each pattern prescribes some feature of a multi-service center building. It describes a relationship which is required to solve a problem which will occur in that building. The summary does not describe this problem; it describes only the pattern [...] [p. 5]

1. [Small Target Areas \(1968\)](#): The multi-service center services a target area with population $34,000 \pm 20\%$.

2. [Location \(1968\)](#): Service centers are located within two blocks of a major intersection.

3. [Size Based on Population \(1968\)](#): The total area of an MSC which services a target area of population N , is $.9N$ square feet.

4. [Community Territory \(1968\)](#): The service center is divided into two zones, services and community territory; community territory includes space for community projects and a public area.

5. [Small Services without Red Tape \(1968\)](#): Each service has a staff size greater than 12; each service is physically cohesive and autonomous; services are loosely organized with respect to each other.

6. [Expansion \(1968\)](#): The number of services can grow and the size of any one service can grow; the relationship of all services to community territory does not change.

7. [Entrance Locations \(1968\)](#): The building's



- ABOUT
- SPECIFICATIONS
- SOFTWARE SUPPORT
- PUBLICATIONS
- EVENTS
- FAQ
- CONTACT

- Learn to Use SBGN
- Symbol Highlights
- Published Maps
- Examples
- Templates
- Contribute
- Competition
- SBGN Development

Specifications

There are three orthogonal and complementary visual languages defined in SBGN: Process Description (PD), Entity Relationship (ER), and Activity Flow (AF). They can be seen as three alternative projections of the underlying, more complex biological information.

Process Description language

The SBGN *Process Description* (PD) language shows the temporal courses of biochemical interactions in a network. It can be used to show all the molecular interactions taking place in a network of biochemical entities, with the same entity appearing multiple times in the same diagram.

Current Specification:

The [SBGN PD language Level 1 Version 1.3](#) – Available at *Journal of Integrative Bioinformatics*, 2015 Sep 4;12(2):263 (doi: 10.2390/biecoll-jib-2015-263).

For end-users more than software developers [a user manual is also available](#).

Entity Relationship language

The SBGN *Entity Relationship* (ER) language allows you to see all the relationships in which a given entity participates, regardless of the temporal aspects. Relationships can be seen as rules describing the influences of entities nodes on other relationships.

Current Specification:

The [SBGN ER language Level 1 Version 2.0](#) – Available at *Journal of Integrative Bioinformatics*, 2015 Sep 4;12(2):264 (doi:10.2390/biecoll-jib-2015-264)

Activity Flow language

The SBGN *Activity Flow* (AF) language depicts the flow of information between biochemical entities in a network. It omits information about the state transitions of entities and is particularly convenient for representing the effects of perturbations, whether genetic or environmental in nature.

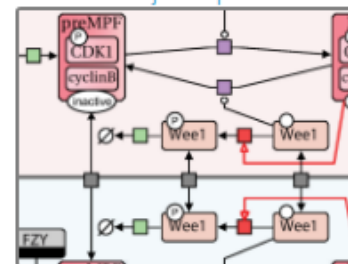
Current Specification:

The [SBGN AF language Level 1 Version 1.2](#) – Available at *Journal of Integrative Bioinformatics*, 2015 Sep 4;12(2):265. (doi:10.2390/biecoll-jib-2015-265).

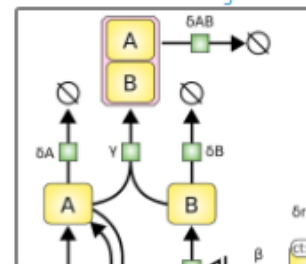
Published maps

This page collects examples of SBGN-compliant diagrams from

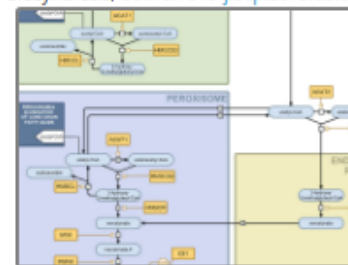
PD map of the *Drosophila* cell cycle,
doi:10.1371/journal.pcbi.1005740



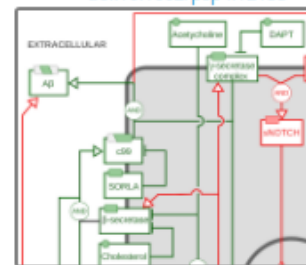
PD map of two-gene system beha
doi:10.1038/nrg3885



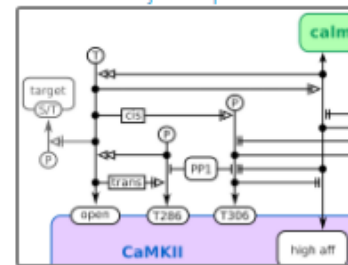
PD map of the mammalian cholesterol
biosynthesis, doi:10.1016/j.bcp.2013.03.021



AF map of protein precursor proces
doi:10.1002/psp.4.12155



ER map of CaMKII regulation by calmodulin,
doi:10.1371/journal.pone.0029406





Contents lists available at ScienceDirect

Ecological Modelling

journal homepage: www.elsevier.com/locate/ecolmodel



Holons, creons, genons, envions, in hierarchy theory: Where we have gone

Timothy Allen^{a,*}, Mario Giampietro^b

^a Department of Botany, University of Wisconsin, 480 Lincoln Drive, Madison, WI 53706, United States

^b Universitat Autònoma de Barcelona (UAB), Institute of Environmental Science and Technology (ICTA), Edifici Q (ETSE) Escola Tècnica Superior d'Enginyeria – ICTA, Campus de Bellaterra, 08193 Cerdanyola del Valles, Barcelona, Spain



ARTICLE INFO

Article history:

Received 28 October 2013
Received in revised form 24 June 2014
Accepted 25 June 2014
Available online 16 July 2014

Keywords:

Complexity
Hierarchy theory
Holon
Network ecology
Narrative
Systems analysis

ABSTRACT

This paper compares and contrasts hierarchy theory and network theory, with the purpose of instructing practitioners in both fields, particularly network theorist, as to how each might relate, and translate to the other. Hierarchy theory and network theory are distinctive but twins. Network theory works its way upscale, incrementally, while hierarchy theory reaches upscale, happy to redefine situations at each new level. Both theories are distinguished from most others in their use of holons. Holons are the vehicle used in this paper to tie network and hierarchy theory together, and show how working in tandem they can advance complexity theory in biology in general. Holons are dual structures that embody contradiction in simultaneous wholeness and partness. Patten defines holons in terms of how they function, and in this way he translates across levels with explicit steps. He does this by specifying the input envions (environment) to feed creons, the input points of holons. The output envion is fed by the holon's genon, the points of output. These steps limit the rescaling of network theory, but allow quantification all the way. Hierarchy theory is not so limited in rescaling, but it pays the price of limiting quantification across levels. Hierarchy theory reaches further upscale with set theoretic devices that make it robust across many levels. It is explicit about the categories. Networks are internally consistent and so present models, the dualities of holons notwithstanding. When inconsistency looms, hierarchy theory moves to narratives, which do not have to be consistent, as models must. In a new elaboration of holon here, hierarchy theory identifies an energy/matter half separate from a coded information half. There are three processes: creating, becoming something else, and narrating to the world; all three progress at their own rates, associated with different causalities. It all maps onto taxon, creon, genon, and envions, emphasizing the larger unity of network and hierarchy theory. Biological and ecological sub-disciplines map onto different parts of the holon. There is also a new theory of how observer decisions are critical in holons. The move between levels that characterizes complexity causes complex systems to become undefinable. With regard to that issue hierarchy theory offers the robustness of narrative form, while network theory hangs on to definitions as long as it can. As hierarchy theory moves upscale, fixed parameters become variables and lose their constancy. In this way structures melt into behavior of some yet higher level structure. Hierarchy theory considers melting structure as being no problem, while network theory ignores the fact that just beyond its purview, structures do indeed melt. So we need hierarchy theory and network theory in tandem to make network theory bolder, and hierarchy theory more tractably quantitative.

© 2014 Elsevier B.V. All rights reserved.

This paper compares and contrasts hierarchy theory and network theory, as devices for pressing the issue of complex systems. Our purpose is to instruct practitioners in both fields. We also intend to show practitioners from both fields the utility of hierarchy and network approaches as duals for addressing complexity.

Particularly network theorists might benefit from how the two theories relate and translate to the other. The danger is for network theorists to fail to take hierarchy theory seriously, dismissing it as a qualitative preliminary, before the real work of quantification and algebraic calculation. United, or at least juxtaposed, hierarchy theory and network theory can make unique contributions to complexity science. This paper invokes a large number of devices, so we need a thread so the reader does not get lost. Accordingly the vehicle for moving this paper forward is the holon, which will be defined

* Corresponding author. Tel.: +1 6086985729; fax: +1 6082627509.
E-mail address: tallen@wisc.edu (T. Allen).

<http://dx.doi.org/10.1016/j.ecolmodel.2014.06.017>
0304-3800/© 2014 Elsevier B.V. All rights reserved.



Unraveling the Complexity of the Jevons Paradox: The Link Between Innovation, Efficiency, and Sustainability

Mario Giampietro^{1,2*} and Kozo Mayumi³

¹ Institut de Ciència i Tecnologia Ambiental, Universitat Autònoma de Barcelona, Bellaterra, Spain, ² Institut Català de Recerca i Estudis Avançats, Barcelona, Spain, ³ Faculty of Integrated Arts and Sciences, Tokushima University, Tokushima, Japan

OPEN ACCESS

Edited by:

Franco Ruzzenenti,
University of Groningen, Netherlands

Reviewed by:

Grégoire Wallenborn,
Université libre de Bruxelles, Belgium
Marco Raugel,
Oxford Brookes University,
United Kingdom

*Correspondence:

Mario Giampietro
mario.giampietro@uab.cat

Specialty section:

This article was submitted to
Energy Systems and Policy,
a section of the journal
Frontiers in Energy Research

Received: 10 January 2018

Accepted: 20 March 2018

Published: 04 April 2018

Citation:

Giampietro M and Mayumi K (2018)
Unraveling the Complexity of the
Jevons Paradox: The Link Between
Innovation, Efficiency, and
Sustainability. *Front. Energy Res.* 6:26.
doi: 10.3389/fenrg.2018.00026

The term "Jevons Paradox" flags the need to consider the different hierarchical scales at which a system under analysis changes its identity in response to an innovation. Accordingly, an analysis of the implications of the Jevons Paradox must abandon the realm of reductionism and deal with the complexity inherent in the issue of sustainability: when studying evolution and real change how can we define "what has to be sustained" in a system that continuously becomes something else? In an attempt to address this question this paper presents three theoretical concepts foreign to conventional scientific analysis: (i) complex adaptive systems—to address the peculiar characteristics of learning and self-producing systems; (ii) holons and holararchy—to explain the implications of the ambiguity found when observing the relation between functional and structural elements across different scales (steady-state vs. evolution); and (iii) Holling's adaptive cycle—to illustrate the existence of different phases in the evolutionary trajectory of a complex adaptive system interacting with its context in which either external or internal constraints can become limiting. These concepts are used to explain systemic drivers of the Jevons Paradox. Looking at society's thermodynamic foundations, sustainability is based on a dynamic balance of two contrasting principles regulating the evolution of complex adaptive systems: the minimum entropy production and the maximum energy flux. The co-existence of these two principles explains why in different situations innovation has to play a different role in the "sustainable development" of society: (i) when society is not subject to external biophysical constraints improvements in efficiency serve to increase the final consumption of society and expand its diversity of functions and structures; (ii) when the expansion of society is limited by external constraints improvements in efficiency should be used to avoid as much as possible the loss of the existing diversity. It is concluded that sustainability cannot be achieved by technological innovations alone, but requires a continuous process of institutional and behavioral adjustment.

Keywords: Jevons paradox, energy efficiency, innovation, complex adaptive system, holon, rebound effect, MuSIASEM, complexity theory

Get the book

Open Innovation Learning: Theory-building on open sourcing while private sourcing, [CC-BY-SA](#) 2017, 2018 [David Ing](#); preface by [Jim Spohrer](#).



Open Access
ePub3 eBook (2018),
ISBN 978-1-7751672-2-8
free of charge
on [Kobo](#),
on the [Internet Archive](#), and
at [Smashwords](#).



E-reader (Kindle)
mobi eBook (2018),
ISBN 978-1-7751672-3-5
ASIN B079RMN1T7
in [Canada](#) for \$0.99CAD,
in the [USA](#) for \$0.79USD,
in the [UK](#) for £0.99,



Open Access
A4 PDF eBook (2017),
ISBN 978-1-7751672-0-4,
DOI:[10.20850/9781775167204](#)
at the [Glasstree Shop](#),
free of charge [with log in](#)



Perfect bound
book (2017) in color,
ISBN 978-1-7751672-1-1,
DOI:[10.20850/9781775167211](#)
at the [Glasstree Shop](#),
\$22.45USD + shipping

