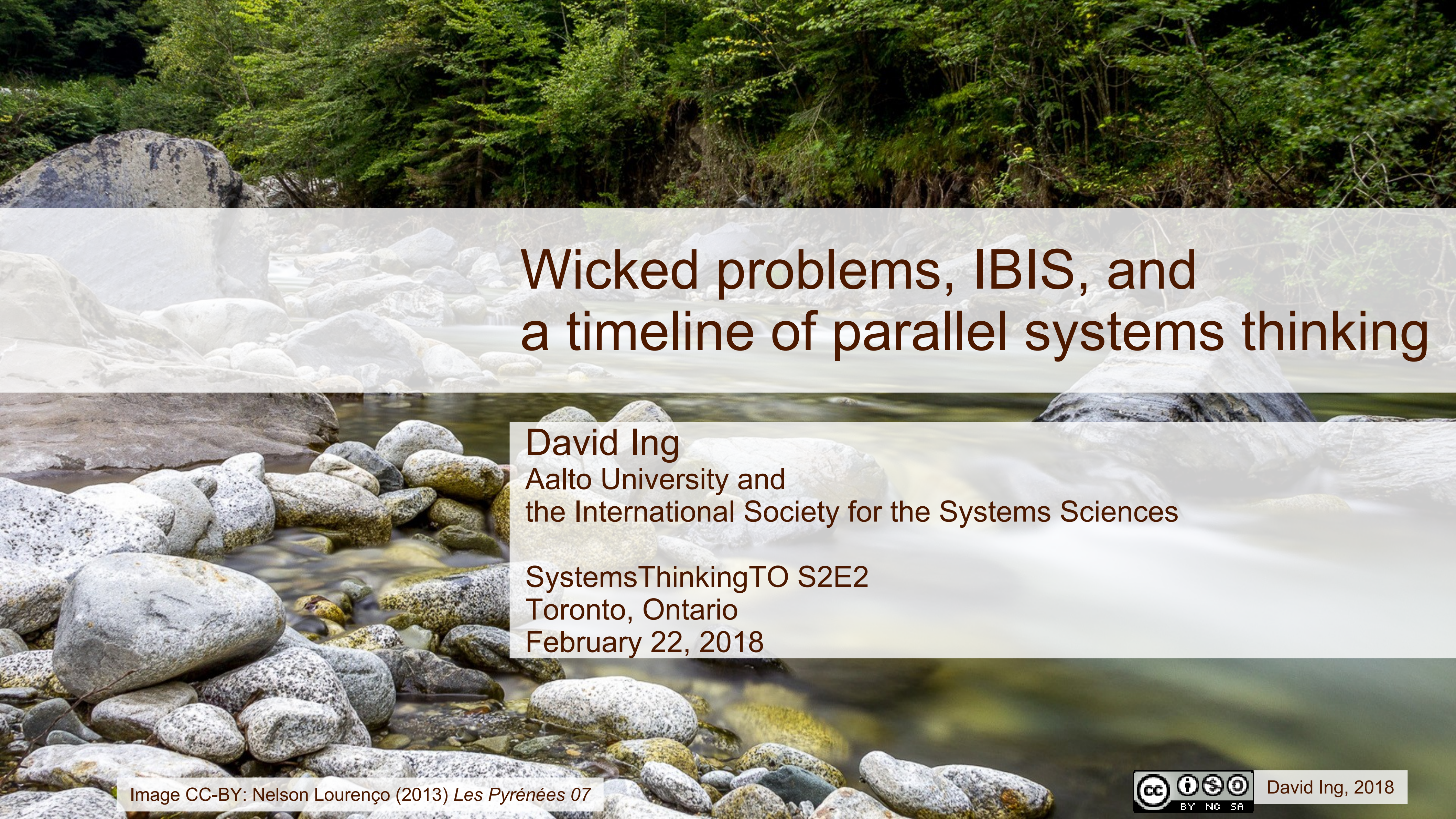


Wicked problems, IBIS, and a timeline of parallel systems thinking

David Ing
Aalto University and
the International Society for the Systems Sciences

SystemsThinkingTO S2E2
Toronto, Ontario
February 22, 2018

Image CC-BY: Nelson Lourenço (2013) *Les Pyrénées 07*

Agenda

1. Wicked problems and IBIS

2. Systems approach (Churchman) + Pattern language (Alexander)

3. Architecture + agile

“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). [....]

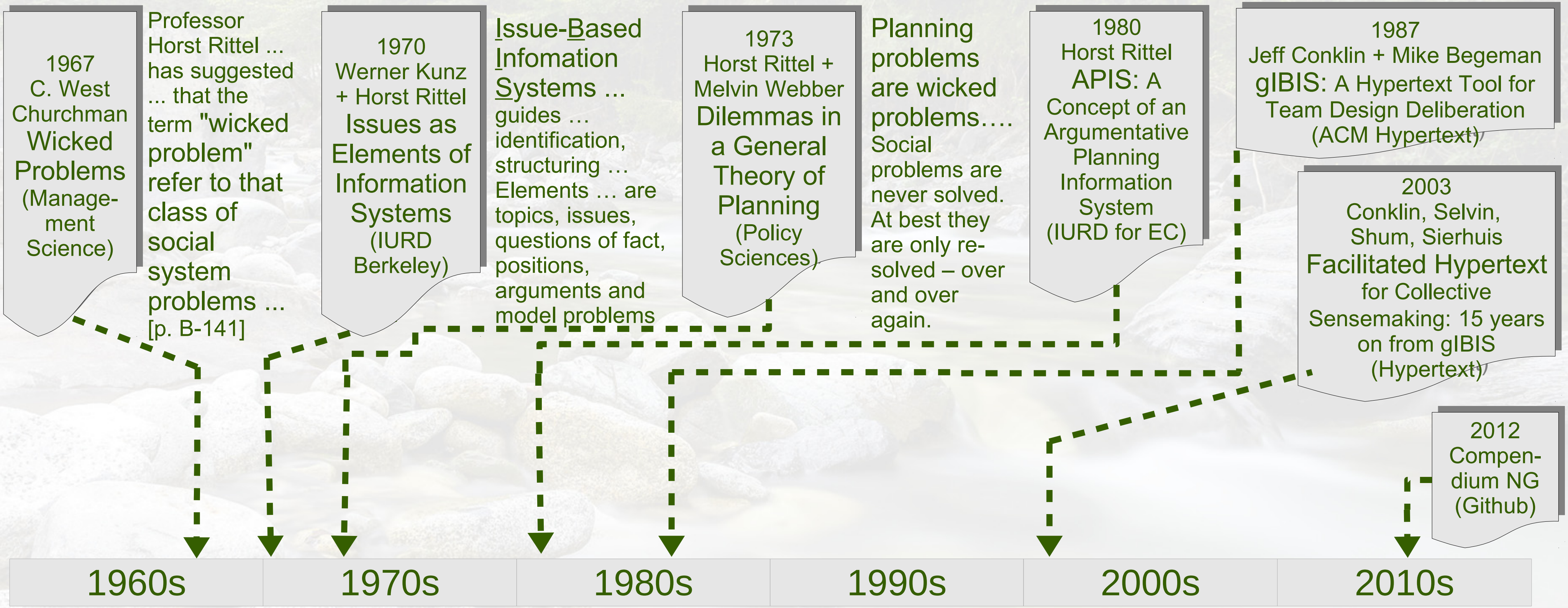
Ten distinguishing properties of planning-type (wicked) problems

	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 *the* 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

	Tame *(benign) problems*	***Wicked*** *(malignant) problems*
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 a **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)

Wicked problems led to IBIS and argumentation schemes



1967
C. West Churchman
Wicked Problems
(Management Science)

Professor Horst Rittel ... has suggested ... that the term "wicked problem" refer to that class of social system problems ... [p. B-141]

1970
Werner Kunz + Horst Rittel
Issues as Elements of Information Systems
(IURD Berkeley)

Issue-Based Infomation Systems ... guides … identification, structuring … Elements … are topics, issues, questions of fact, positions, arguments and model problems

1973
Horst Rittel + Melvin Webber
Dilemmas in a General Theory of Planning
(Policy Sciences)

Planning problems are wicked problems…. Social problems are never solved. At best they are only re-solved – over and again.

1980
Horst Rittel
APIS: A Concept of an Argumentative Planning Information System
(IURD for EC)

1987
Jeff Conklin + Mike Begeman
gIBIS: A Hypertext Tool for Team Design Deliberation
(ACM Hypertext)

2003
Conklin, Selvin, Shum, Sierhuis
Facilitated Hypertext for Collective Sensemaking: 15 years on from gIBIS
(Hypertext)

2012
Compendium NG
(Github)

1960s | 1970s | 1980s | 1990s | 2000s | 2010s

Wicked problems ↔ IBIS: Issues-Based Information Systems

Issue-Based Information Systems (IBIS) are meant to support coordination and planning of political decision processes.

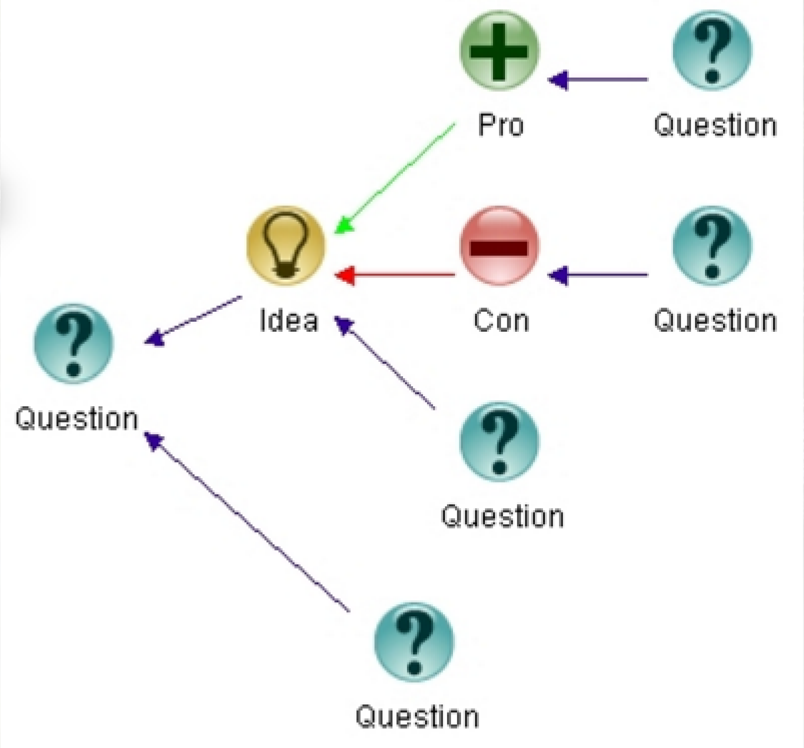
- IBIS guides the …
  - identification,
  - structuring and
  - settling of issues

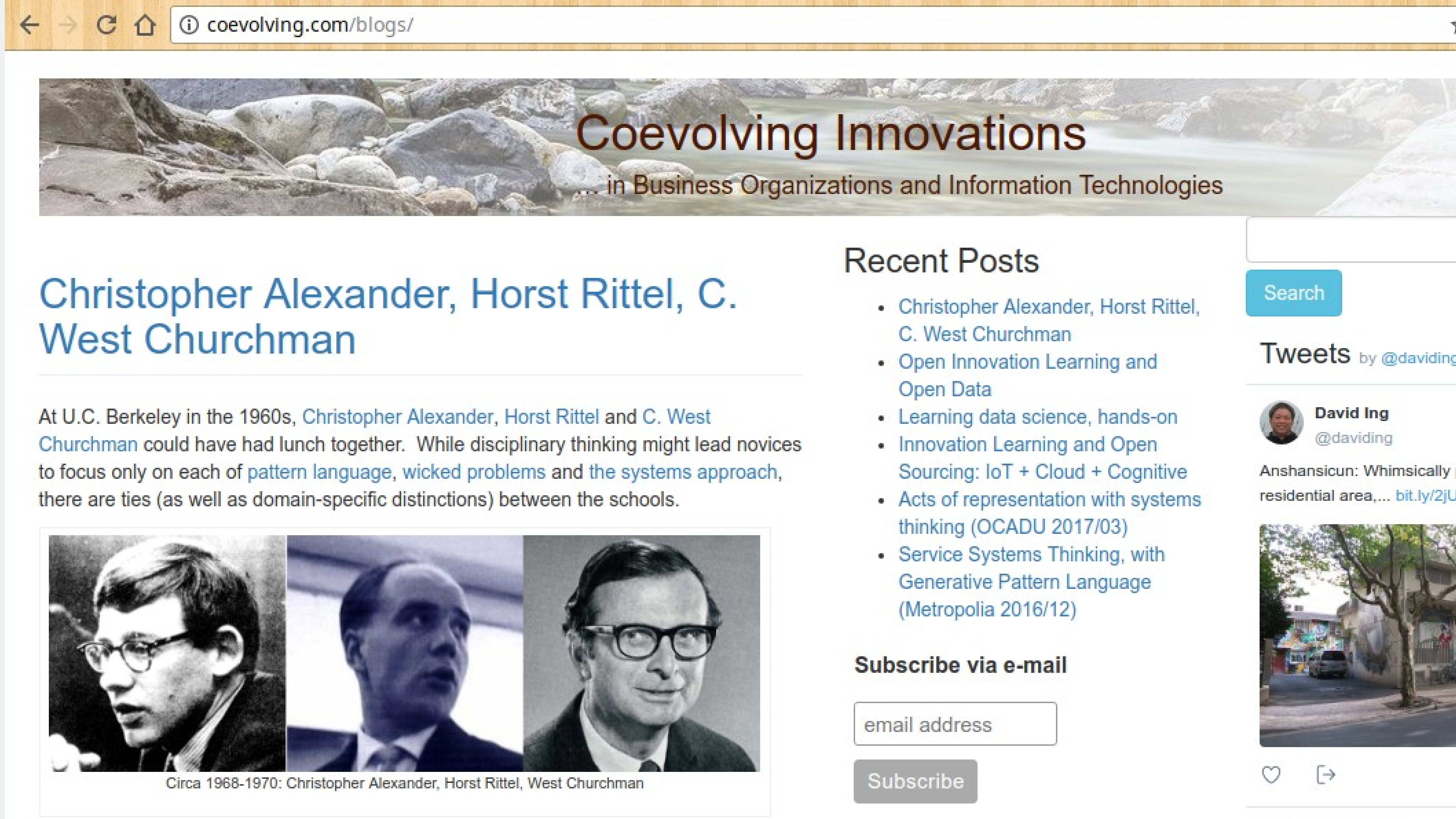
raised by problem-solving groups, and provides information pertinent to the discourse.

Elements of the system are

- topics,
- issues,
- questions of fact,
- positions,
- arguments, and
- model problems.

(Kunz & Rittel, 1970)





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

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

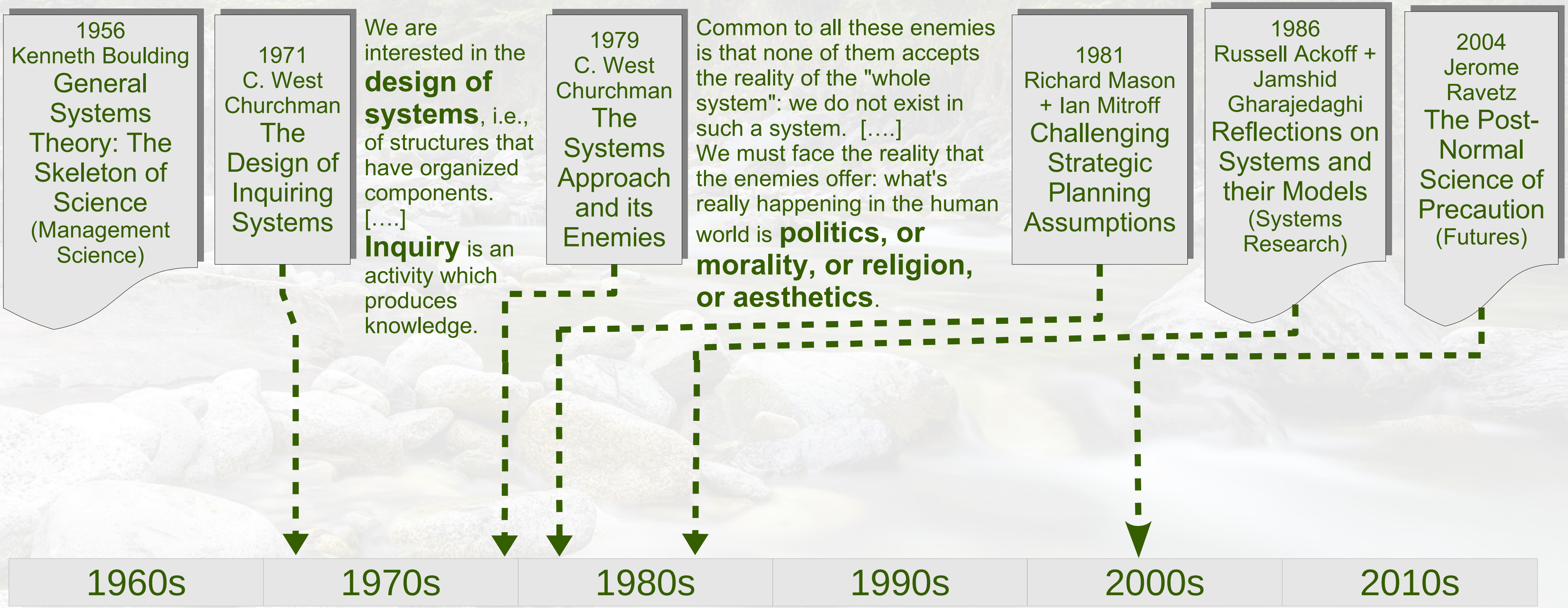
Christopher Alexander (1936 -)

- 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 approach led to assumption surfacing, postnormal science



1956 Kenneth Boulding **General Systems Theory: The Skeleton of Science** (Management Science)

1971 C. West Churchman **The Design of Inquiring Systems**

We are interested in the **design of systems**, i.e., of structures that have organized components. [….] **Inquiry** is an activity which produces knowledge.

1979 C. West Churchman **The Systems Approach and its Enemies**

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. [….] 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**.

1981 Richard Mason + Ian Mitroff **Challenging Strategic Planning Assumptions**

1986 Russell Ackoff + Jamshid Gharajedaghi **Reflections on Systems and their Models** (Systems Research)

2004 Jerome Ravetz **The Post-Normal Science of Precaution** (Futures)

1960s	1970s	1980s	1990s	2000s	2010s

“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).

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

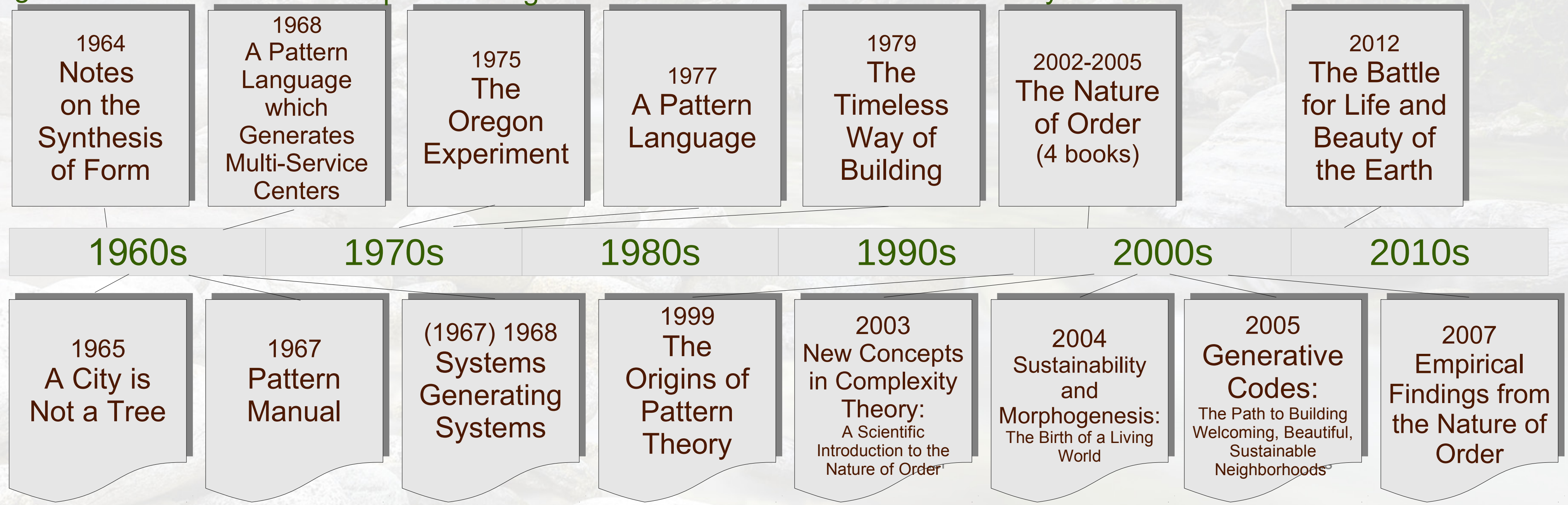
“process of design”, “goodness of fit”

“organic order”, “participation”, “piecemeal growth”

“the quality without a name”

“wholeness and the theory of centers”

“production system-A” “life-giving, “environment building”; “system-B” “mechanical”, mass-produced”



1964
Notes on the Synthesis of Form

1968
A Pattern Language which Generates Multi-Service Centers

1975
The Oregon Experiment

1977
A Pattern Language

1979
The Timeless Way of Building

2002-2005
The Nature of Order
(4 books)

2012
The Battle for Life and Beauty of the Earth

1960s | 1970s | 1980s | 1990s | 2000s | 2010s

1965
A City is Not a Tree

1967
Pattern Manual

(1967) 1968
Systems Generating Systems

1999
The Origins of Pattern Theory

2003
New Concepts in Complexity Theory:
A Scientific Introduction to the Nature of Order

2004
Sustainability and Morphogenesis:
The Birth of a Living World

2005
Generative Codes:
The Path to Building Welcoming, Beautiful, Sustainable Neighborhoods

2007
Empirical Findings from the Nature of Order

“natural cities”, (“artificial cities”), “semilattice”

“system as a whole”, “generative system”

“wholeness and value”, “recursive structure”, “objective measures of coherence”

“life”, “wholeness”, “wholeness-extending transformations”

“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)

All architecture is design, but not all design is architecture

Architectural thinking as
shaping the structure of the environment ...

Living systems are *autopoietic*,
self-organizing and self-generating;

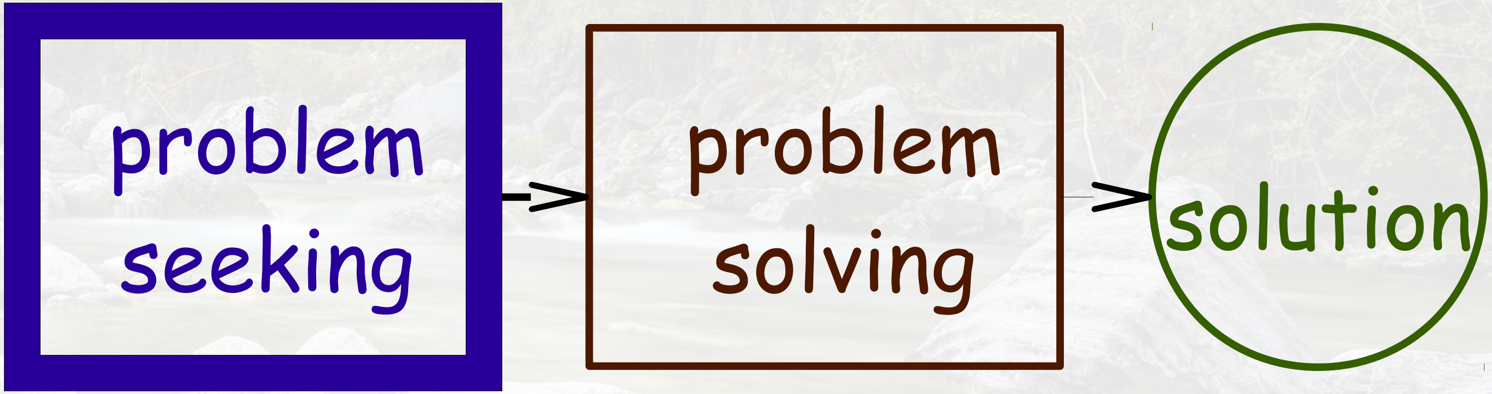
assembly lines are *allopoietic*,
externally-organizing and externally-generating.

Design thinking as
divergent steps (i.e. creating choices) and
convergent steps (i.e. making choices)

In 1969, problem seeking was *architectural programming*, and problem solving was *design*

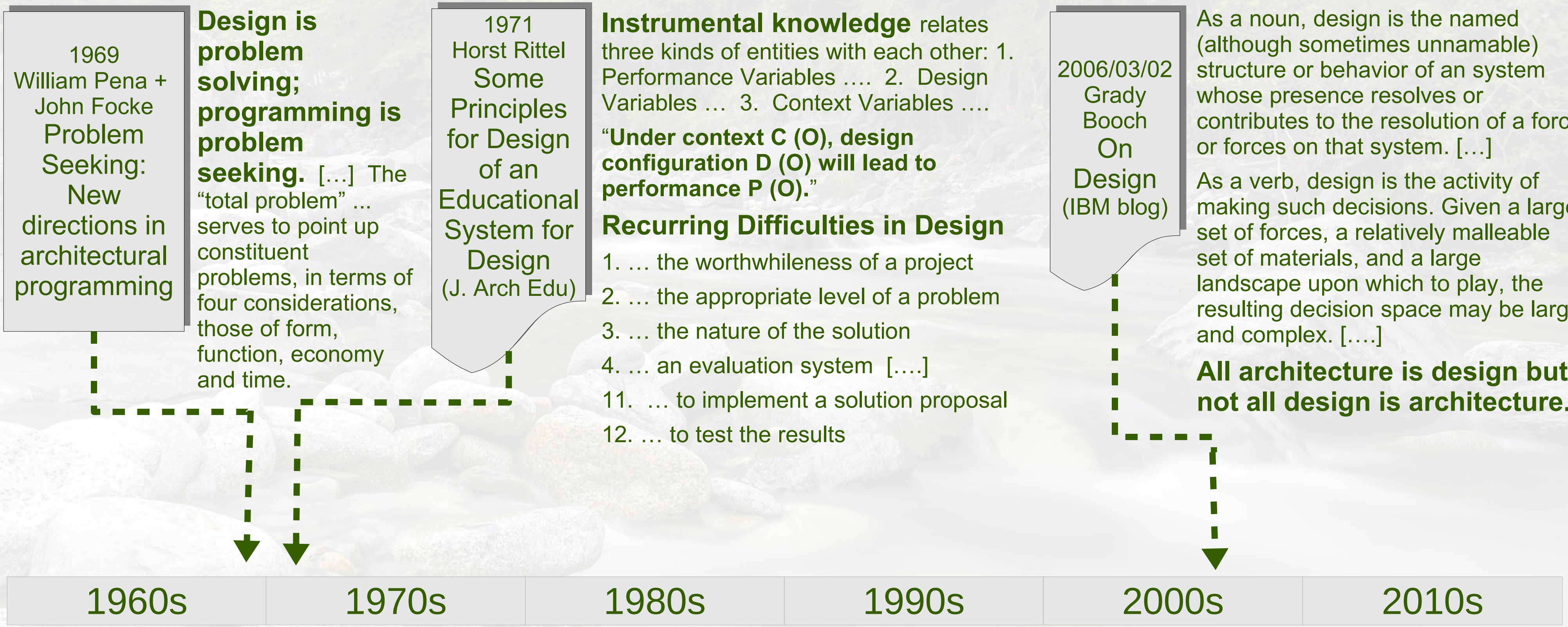
Programming is a specialized and often misunderstood term. It is "*a statement of an architectural problem* and the requirements to be met in offering a solution. While the term is used with other descriptive adjectives such as *computer* programming, *educational* programming, *functional* programming, etc., in this report, programming is used to refer only to architectural programming.

Why programming? The client has a project with many unidentified sub-problems. The architect must define the client's total problem.



Design is problem solving; programming is problem seeking. The end of the programming process is a statement of the total problem; such a statement is the element that joins programming and design. The "total problem" then serves to point up constituent problems, in terms of four considerations, those of form, function, economy and time. The aim of the programming is to provide a sound basis for effective design. The State of the Problem represents the essense and the uniqueness of the project. Furthermore, it suggests the solution to the problem by defining the main issues and giving direction to the designer (Pena and Focke 1969, 3).

Architecture ~ problem-seeking. Design ~ problem-solving



1969
William Pena +
John Focke
Problem Seeking: New directions in architectural programming

Design is problem solving; programming is problem seeking. […] The "total problem" ... serves to point up constituent problems, in terms of four considerations, those of form, function, economy and time.

1971
Horst Rittel
Some Principles for Design of an Educational System for Design
(J. Arch Edu)

Instrumental knowledge relates three kinds of entities with each other: 1. Performance Variables …. 2. Design Variables … 3. Context Variables ….

"Under context C (O), design configuration D (O) will lead to performance P (O)."

Recurring Difficulties in Design

1. … the worthwhileness of a project
2. … the appropriate level of a problem
3. … the nature of the solution
4. … an evaluation system [….]
11. … to implement a solution proposal
12. … to test the results

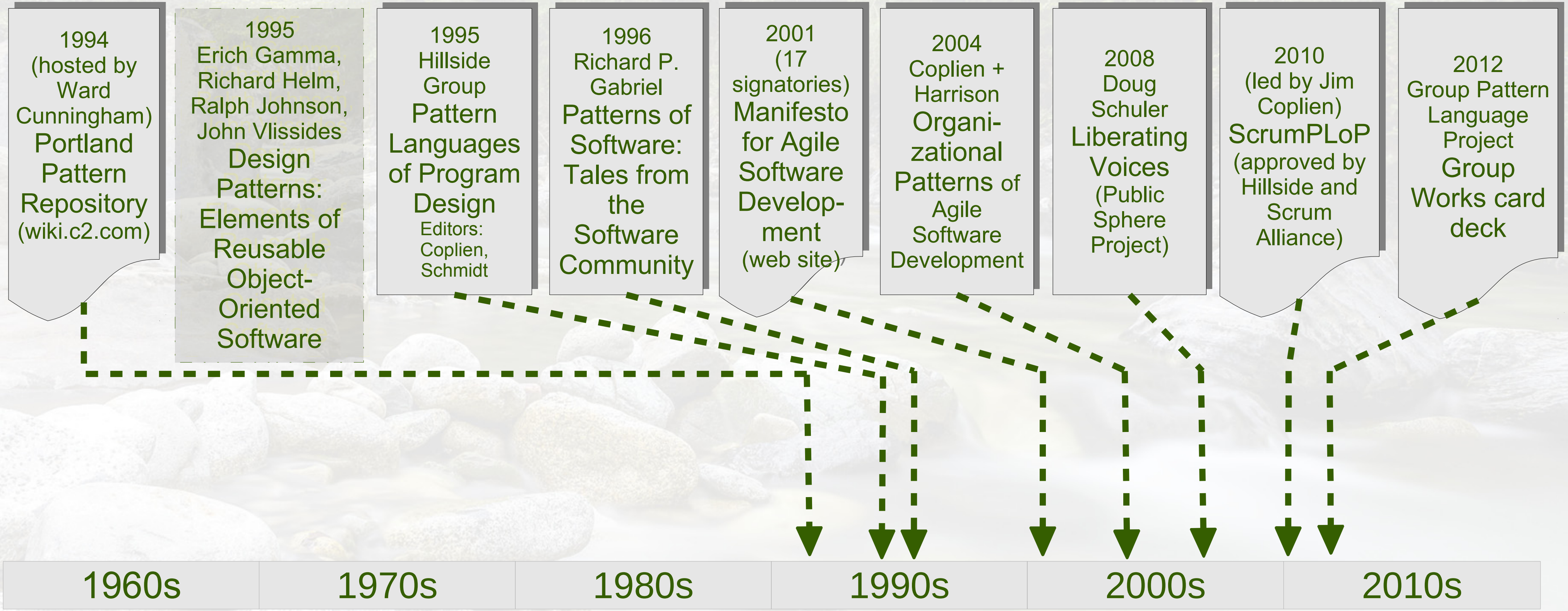
2006/03/02
Grady Booch
On Design
(IBM blog)

As a noun, design is the named (although sometimes unnamable) structure or behavior of an system whose presence resolves or contributes to the resolution of a force or forces on that system. […]

As a verb, design is the activity of making such decisions. Given a large set of forces, a relatively malleable set of materials, and a large landscape upon which to play, the resulting decision space may be large and complex. [….]

All architecture is design but not all design is architecture.

Pattern language has risen in agile, groups, public sphere



1994
(hosted by Ward Cunningham)
Portland Pattern Repository
(wiki.c2.com)

1995
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Design Patterns: Elements of Reusable Object-Oriented Software

1995
Hillside Group
Pattern Languages of Program Design
Editors: Coplien, Schmidt

1996
Richard P. Gabriel
Patterns of Software: Tales from the Software Community

2001
(17 signatories)
Manifesto for Agile Software Development
(web site)

2004
Coplien + Harrison
Organizational Patterns of Agile Software Development

2008
Doug Schuler
Liberating Voices
(Public Sphere Project)

2010
(led by Jim Coplien)
ScrumPLoP
(approved by Hillside and Scrum Alliance)

2012
Group Pattern Language Project
Group Works card deck

1960s	1970s	1980s	1990s	2000s	2010s

Image CC-BY: Nelson Lourenço (2013) *Les Pyrénées 07*