

Incubating Service Systems Thinking: New frames for collaborating on a pattern language for service systems

David Ing

International Society for the Systems Sciences, and
Aalto University

July 2014



© 2014 David Ing



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Canada License](https://creativecommons.org/licenses/by-nc-nd/2.5/ca/).

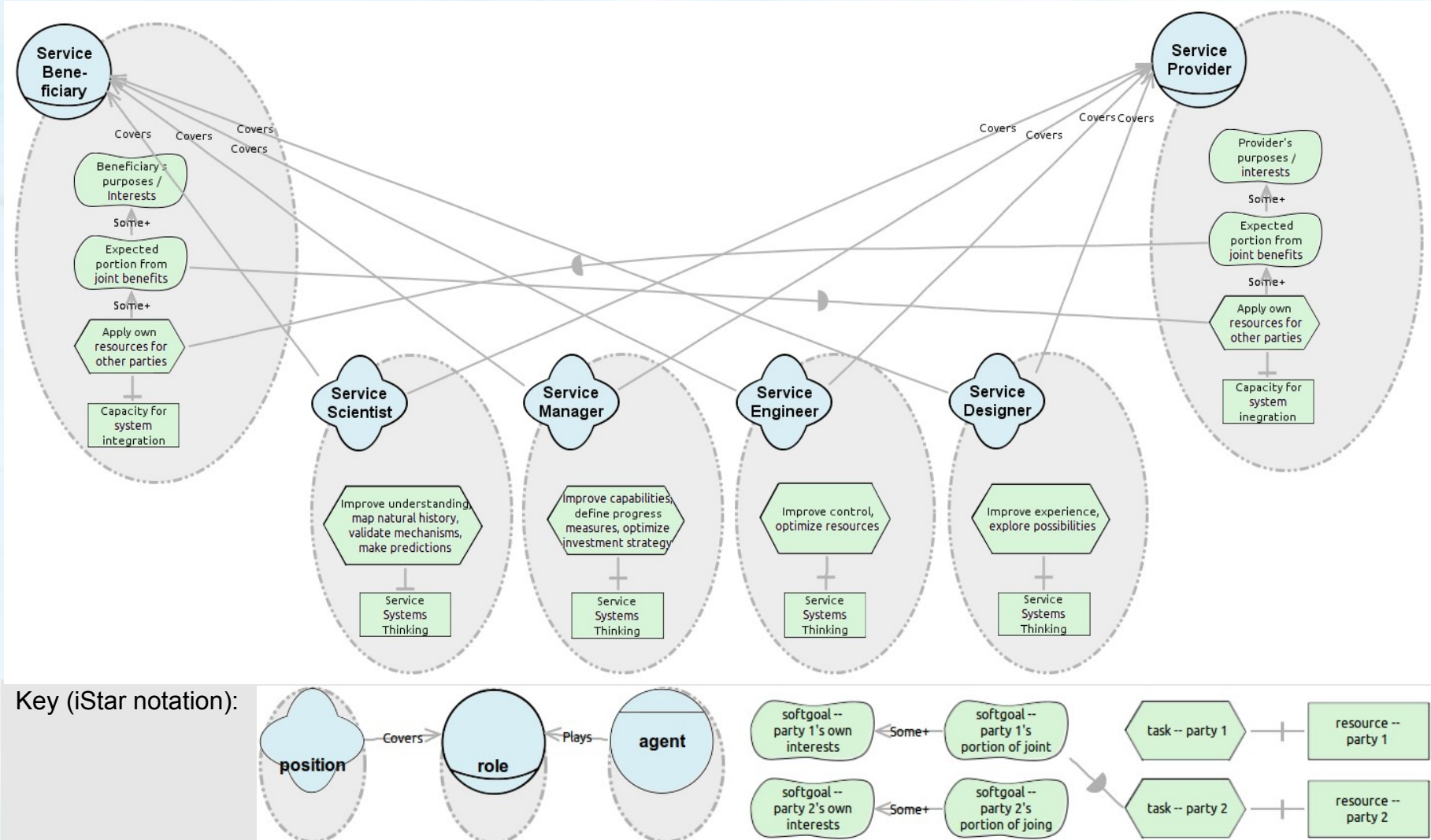
Agenda

- 1. Service Systems Thinking, In Brief
- 2. Conversations for Orientation
- 3. Conversations for Possibilities
- 4. Conversations for Action
- 5. Conversations for Clarification

Agenda

- 1. Service Systems Thinking, In Brief
 - 1.1 An intentional representation
 - 1.2 An object-process representation
- 2. Conversations for Orientation
- 3. Conversations for Possibilities
- 4. Conversations for Action
- 5. Conversations for Clarification

In an intentional representation, service systems thinking is a resource that can be applied by service scientists, managers, engineers and designers



In an object-process representation, service systems thinking is handled by a community



Key (OPM notation):

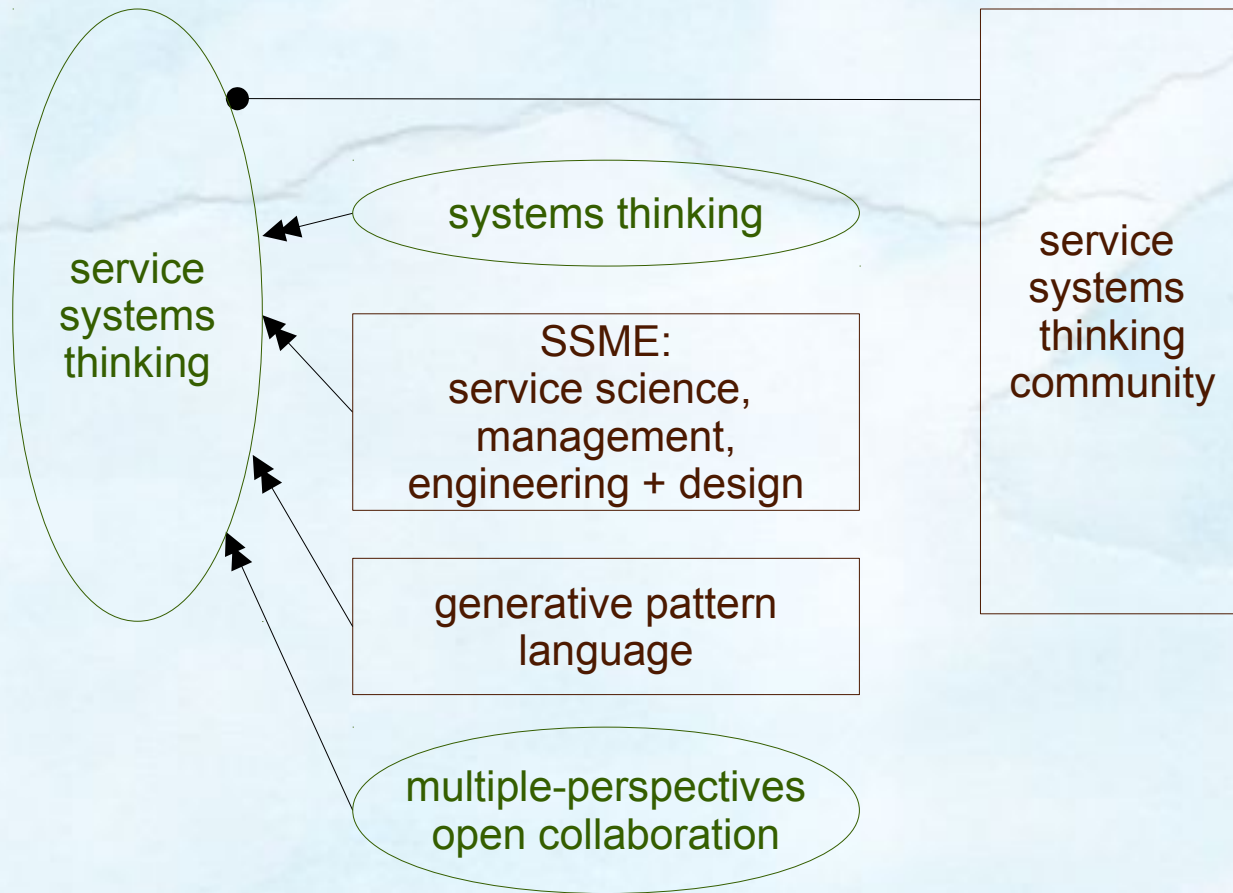
object

process

agent handles process

object is exhibited by (o or p)
process is exhibited by (o or p)

Service systems thinking exhibits systems thinking, SSME, generative pattern language and multiple perspectives open collaboration



Key (OPM notation):

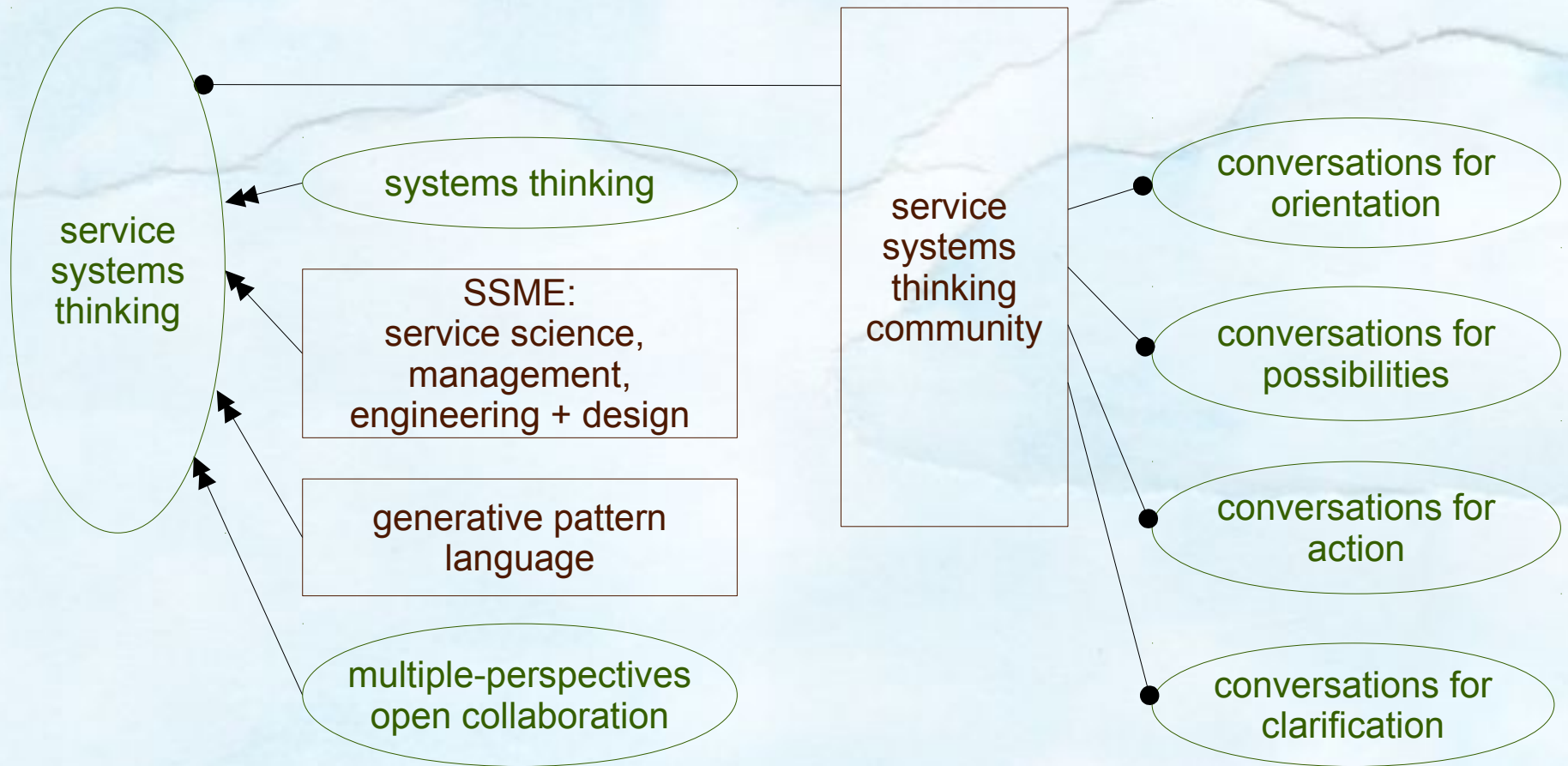
object

process

agent handles process

object is exhibited by (o or p)
process is exhibited by (o or p)

Development within the community can be recognized as conversations: for orientation, for possibilities, for action, and for clarification



Key (OPM notation):

object

process

agent handles process

object is exhibited by (o or p)
process is exhibited by (o or p)

Agenda

1. Service Systems Thinking, In Brief

- 2. Conversations for Orientation
 - 2.1 Systems thinking
 - 2.2 SSMED
(Service Science, Management, Engineering and Design)
 - 2.3 Generative Pattern Language
 - 2.4 Multiple Perspectives Open Collaboration
- 3. Conversations for Possibilities
- 4. Conversations for Action
- 5. Conversations for Clarification

Service Systems Thinking

The Service Systems Thinking was first introduced in January 2014 📄. More history is available at s2t.org



The lay of the land might be appreciated by [Service Systems Thinking, In Brief](#).

Engagement in the community may be clarified by recognizing the language action perspective with four types of conversation. [See an outline of these four types](#) 📄

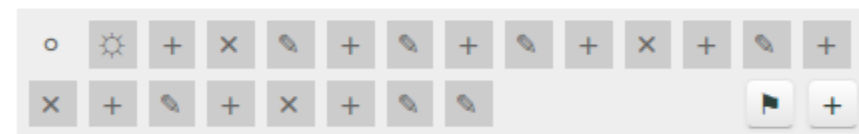
1. [Conversations for Orientation on Service Systems Thinking](#)
2. [Conversations for Possibilities on Service Systems Thinking](#)
3. [Conversations for Action on Service Systems Thinking](#)
4. [Conversations for Clarification on Service Systems Thinking](#)



Conversations for Orientation on Service Systems Thinking

Service systems thinking builds on the foundations of a variety of fields. An appreciation of parts of those fields can serve as a common understanding to be cross-appropriated for further development of a body of knowledge.

1. [Systems Thinking](#)
2. [Service Science, Management, Engineering and Design](#)
3. [Generative Pattern Language](#)
4. [Multiple Perspectives Open Collaboration](#)

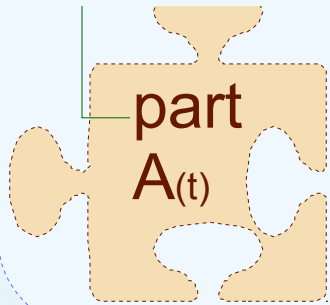


CC BY-SA 3.0 • JSON • fed.coevolving.com

Systems thinking is a perspective on wholes, parts and their relations

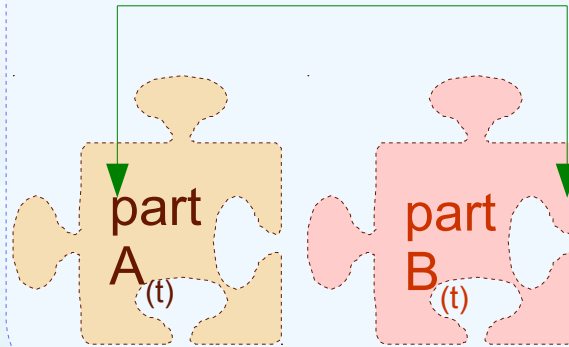
containing
whole

↑
Function (non-living)
or role (living)



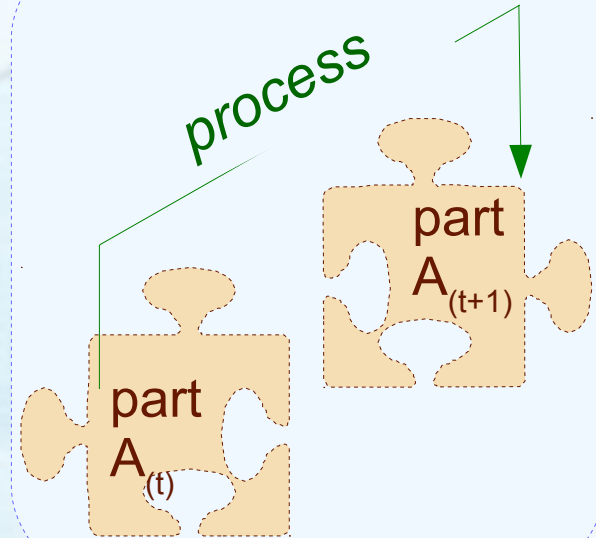
Function
“contribution of the
part to the whole”

structure



Structure
“arrangement in
space”

process



Process
“arrangement in
time”

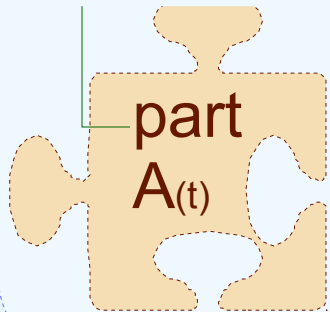
Source: Ing, David. 2013. “Rethinking Systems Thinking: Learning and Coevolving with the World.” *Systems Research and Behavioral Science* 30 (5): 527–47. doi:10.1002/sres.2229. Gharajedaghi, Jamshid. 1999. *Systems Thinking: Managing Chaos and Complexity: A Platform for Designing Business Architecture*. Elsevier. <http://books.google.ca/books?id=7N-sFxFntakC>.

In authentic systems thinking, synthesis precedes analysis and the containing whole is appreciated

containing
whole



*Function (non-living)
or role (living)*



Synthesis precedes analysis

1. Identify a containing whole (system) of which the thing to be explained is a part.
2. Explain the behavior or properties of the containing whole
3. Then explain the behavior or properties of the thing to the explained in terms of its role(s) or function(s) within its containing whole.

Source: Ackoff, Russell L. 1981. *Creating the Corporate Future: Plan or Be Planned For*. New York: John Wiley and Sons.
<http://books.google.com/books?id=8EEO2L4cApsC>.

Service systems in our society can be ranked from concrete to abstract, as subjects for schoolchildren

Systems that move,
store, harvest,
process

•Transportation	K
•Water and waste management	1
•Food and global supply chain	2
•Energy and energy grid	3
•Information and communications (ICT) infrastructure	4

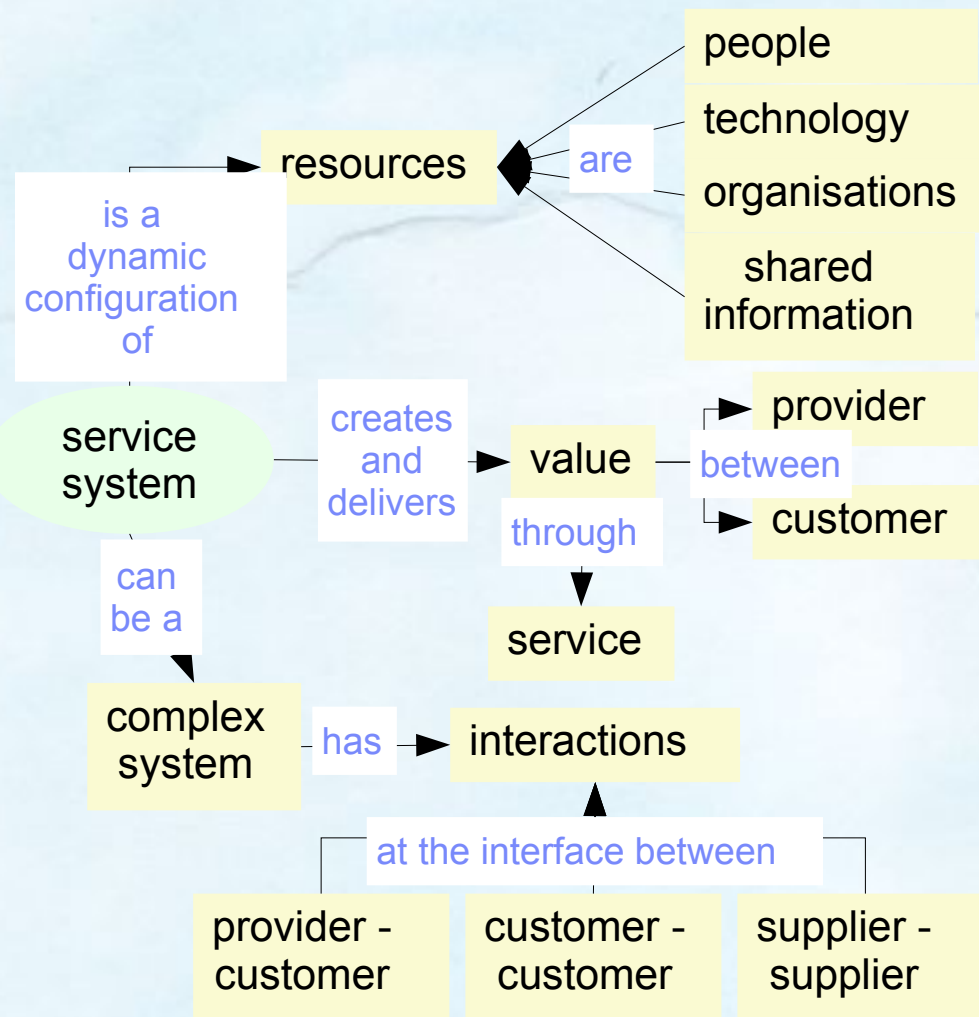
Systems that enable
healthy, wealthy and
wise people

•Building and construction	5
•Banking and finance	6
•Retail and hospitality	7
•Healthcare	8
•Education (including universities)	9

Systems that govern

•Government (cities)	10
•Government (regions / states)	11
•Government (nations)	12

Service systems (Cambridge IfM and IBM, 2008)



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.

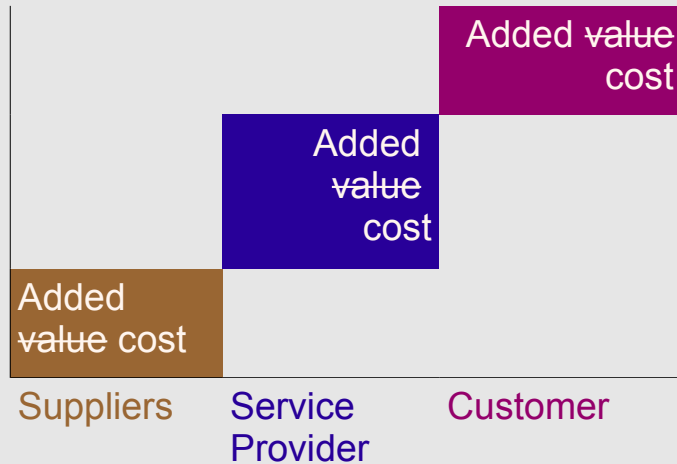
However, with the advent of ICT, customer-to-customer and supplier-to-supplier interactions have also become prevalent.

These complex interactions create a system whose behaviour is difficult to explain and predict.
(IfM and IBM, 2008, p. 6)

Source: IfM, and IBM. 2008. *Succeeding through Service Innovation: A Service Perspective for Education, Research, Business and Government*. Cambridge, UK: University of Cambridge Institute for Manufacturing. <http://www.ifm.eng.cam.ac.uk/ssme/>.

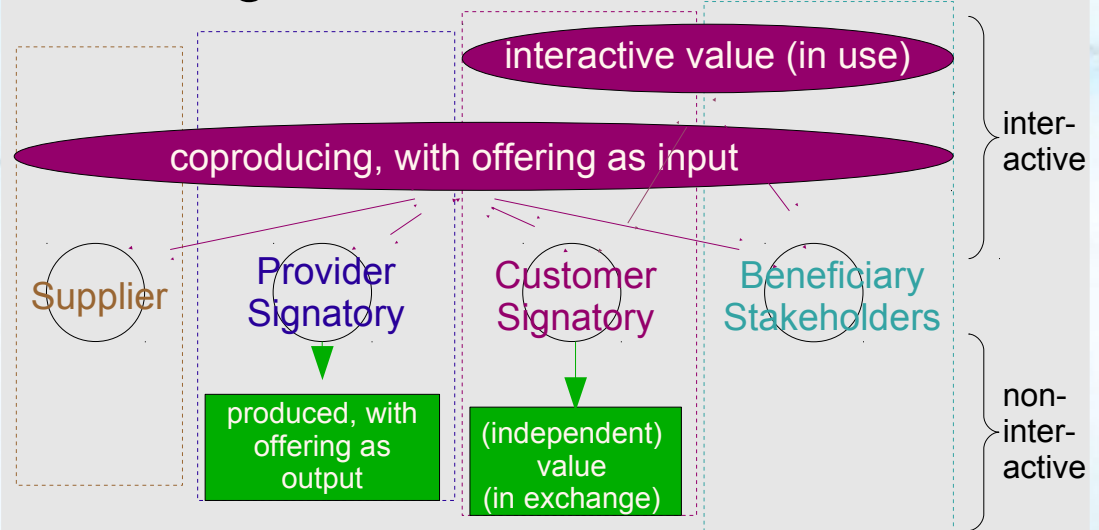
The theory of firms adding value cost has given way to mobilizing customers towards creating their own value

Adding value cost



Our traditional about value ... [says] every company occupies a position on the value chain. Upstream, suppliers provide inputs. The company then adds values to these inputs, before passing them downstream to then next actor in the chain [whether another business or the final consumer].

Enabling interactive value creation



... IKEA's strategic intent [is] to understand how customers can create their own value and create a business system that allows them to do it better. IKEA's goal is not to *relieve* customers of doing certain things but to *mobilize* them to do easily certain things they have never done before. Put another way, IKEA invents value by enabling customers' own value-creating activities. ... Wealth is [the ability] to realize your own ideas.

Source: Richard Normann and Rafael Ramirez. 1993. "From Value Chain to Value Constellation: Designing Interactive Strategy." Harvard Business Review 71: 65–65. <http://hbr.org/1993/07/designing-interactive-strategy> .

Basic Concepts. If we are to understand human history as the evolution and design of value-cocreation mechanisms between entities, then where should we begin?

Let's start by understanding the following ten basic concepts:

1.	Resources	Businesses may own physical resources or contract for physical resources, but as a type of resource they are themselves not physical, but instead a conceptual-legal construct. So in the end, all resources fall into one of four types: <i>physical-with-rights</i> , <i>not-physical-with-rights</i> , <i>physical-with-no-rights</i> , and <i>not-physical-with-rights</i> .
2.	Service system entities	The most common types of service system entities are people and organizations. New types of service system entities are constantly emerging and disappearing. Recently, open-source and on-line communities have emerged as service systems entities.
3.	Access rights	"By what authority, do you use that resource?" Service system entities have four main types of access rights to the resources within their configuration: <i>owned outright</i> , <i>leased/contracted</i> , <i>shared access</i> , and <i>privileged access</i> . Shared access resources include resources such as air, roads, natural language, and internet web sites. Privileged access resources include resources such as thoughts, individual histories, and family relationships.
4.	Value-proposition-based interactions	"I'll do this, if you'll do that." [...] Interactions via value propositions are intended to cocreate-value for both interacting entities. Both interacting entities must agree, explicitly or tacitly, to the value proposition.
5.	Governance mechanisms	"Here's what will happen if things go wrong." [...] If value is not realized as expected, this may result in a dispute between the entities. Governance mechanisms reduce the uncertainty in these situations by prescribing a mutually agreed to process for resolving the dispute.
6.	Service system networks	"Here's how we can all link up." [...] Over time, for a population of entities, the patterns of interaction can be viewed as networks with direct and indirect connectivity strengths. A service system network is an abstraction that only emerges when one assumes a particular analysis overlay on the history of interactions amongst service system entities.
7.	Service system ecology	"Populations of entities, changing the ways they interact." Different types of service systems entities exist in populations, and the universe of all service system entities forms the service system ecology or service world
8.	Stakeholders	"When it comes to value, perspective really matters." The four primary types of stakeholders are <i>customer</i> , <i>provider</i> , <i>authority</i> , and <i>competitor</i> . In addition ... other stakeholder perspectives include employee, partner, entrepreneur, criminal, victim, underserved, citizen, manager, children, aged, and many others.
9.	Measures	"Without standardized measures, it is hard to agree and harder to trust." The four primary types of measures are <i>quality</i> , <i>productivity</i> , <i>compliance</i> , and <i>sustainable innovation</i> .
10.	Outcomes	"How did we do? Can this become a new routine or long-term relationship?" [...] Beyond a standard two player game, with a customer player and a provider player, ISPAR assumes there exists both an authority player as well as a competitor-criminal player.

Source: Jim Spohrer and Stephen K. Kwan. 2009. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline - Outline & References." *International Journal of Information Systems in the Service Sector* 1 (3): 1–31. doi:10.4018/jisss.2009070101 .

Service systems worldview. These ten basic concepts underlie the *service systems worldview* ...

... the world is made up of populations of service system entities that interact (normatively) via value propositions to cocreate-value, but often disputes arise and so governance mechanisms are invoked to resolve disputes.

1.	Resources
2.	Service system entities
3.	Access rights
4.	Value-proposition-based interactions
5.	Governance mechanisms
6.	Service system networks
7.	Service system ecology
8.	Stakeholders
9.	Measures
10.	Outcomes

Formal service system entities are types of legal entities with rights and responsibilities, that can own property, and with named identities that can create contracts with other legal entities. [...] Formal service systems exist within a legal and economic framework of contracts and expectations.

Informal service system entities include families ..., open source communities ..., and many other societal or social systems that are governed typically by unwritten cultural and behavioral norms (social systems with rudimentary political systems).

Natural history of service system entities. Service science seeks to create an understanding of the formal and informal nature of service in terms of entities, interactions, and outcomes, and how these evolve (or are designed) over time. An initial premise is that the entities, which are sophisticated enough to engage in rationally designed service interactions that can consistently lead to win-win value cocreation outcomes, must be able to build models of the past (reputation, trust), present, and future (options, risk-reward, opportunities, hopes and aspirations) possible worlds, including models of themselves and others, and reason about knowledge value

Basic questions. A general theory of service system entities and networks formed through value-proposition-based interactions has four parts

... which directly lead to the four basic types of questions that SSMED seeks to answer.

Science

(improve understanding, map natural history, validate mechanisms, make predictions).

What are service system entities, how have they naturally evolved to present, and how might they evolve in the future? What can we know about their interactions, how the interactions are shaped (value propositions, governance mechanisms), and the possible outcomes of those interactions both short-term and long-term?

Management

(improve capabilities, define progress measures, optimize investment strategy).

How should one invest to create, improve, and scale service system networks? How do the four measures of quality, productivity, compliance, and sustainable innovation relate to numerous key performance indicators (KPIs) of business and societal systems? Is there a “Moore’s Law” of service system investment? Can doubling information lead to a doubling of capabilities (performance) on a predictable basis?

Engineering

(improve control, optimize resources).

How can the performance of service system entities and scaling of service system networks be improved by the invention of new technologies (and environmental infrastructures) or the reconfiguration of existing ones? What is required to develop a CAD (Computer-Aided Design) tool for service system entity and service system network design?

Design

(improve experience, explore possibilities).

How can one best improve the experience of people in service system entities and networks? How can the experience of service system creation, improvement, and scaling be enhanced by better design? Can the space of possible value propositions and governance mechanisms be explored systematically?

Sciences of the artificial. Sciences of the artificial are different from natural sciences, and so it becomes especially important to consider these four parts – science, management, engineering, and design – as important knowledge components. In “The Sciences of the Artificial” (Simon 1996), Simon reflects “The world we live in today is much more man-made, or artificial, world than it is a natural world....

Service Science, Management, Engineering, and Design (SSMED) is emerging as one of the sciences of the artificial. Service science is knowledge about service system entities, value-proposition-based interactions (or value-cocreation mechanisms), governance mechanisms, and the other seven basic concepts. Following Simon even further, one could argue that service system entities are physical symbol systems, dealing with symbols that are named resources, and grounded in physical routines for carrying out the symbolic manipulations related to named resources.

Source: Jim Spohrer and Stephen K. Kwan. 2009. “Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline - Outline & References.” *International Journal of Information Systems in the Service Sector* 1 (3): 1–31. doi:10.4018/jisss.2009070101 .

Key concepts of value cocreation can be expressed through intentional (iStar) modeling constructs

Key service system concepts

*i** constructs

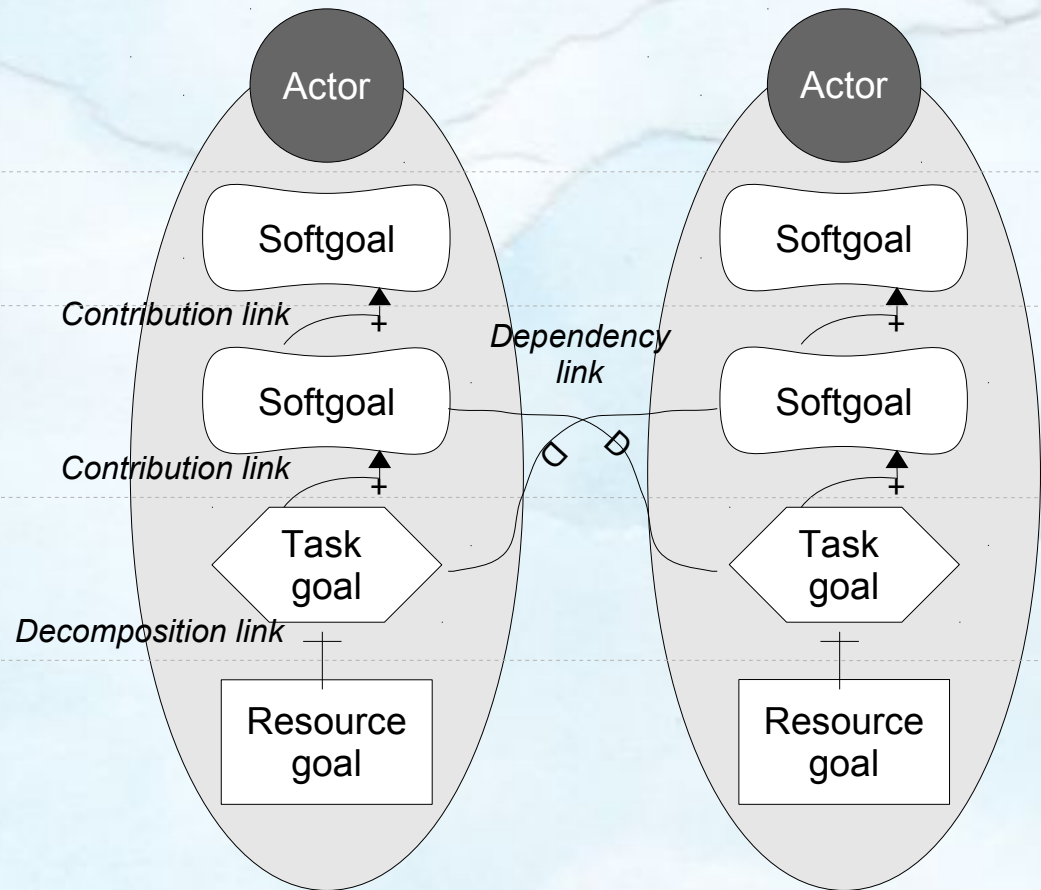
Service system entity

High-level interests

Expected benefits

Value propositions

Resources



Source: Lysanne Lessard and Eric Yu. 2013. "Service Systems Design: An Intentional Agent Perspective." *Human Factors and Ergonomics in Manufacturing & Service Industries* 23 (1): 68–75. doi:10.1002/hfm.20513.

Generative pattern language – systems generating systems (1968)

← → ↺ 🏠 [coevolving.com/blogs/index.php/archive/systems-generating-systems-architectural-design-theory-by-christopher-](http://coevolving.com/blogs/index.php/archive/systems-generating-systems-architectural-design-theory-by-christopher-alexander)

Systems generating systems — architectural design theory by Christopher Alexander (1968)

Posted on April 10, 2014 by [daviding](#)

The systems thinking roots from architect [Christopher Alexander](#) aren't completely obvious in his work on [pattern language](#). A [republished version of an 1968 article](#) resurfaces some clarification on a perspective on systems thinking originating from practices in architecture.

This article introduced ways in which systems thinking could be most directly applied to [built environments](#). The cross-appropriation of pattern languages across a variety of domain types — object-oriented programmers were the earliest motivating adopters — could be enlightened by revisiting the foundations. Alexander concisely presented 4 points, and then provided detailed reasoning for each:

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. [Alexander 2011, p. 59; Alexander 1968, p. 605]

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 2011, p. 58; Alexander 1968, p. 605]

Let's leave analytical explications of the original 1968 text as secondary, to first appreciate the idea of "systems generating systems" through sensemaking done some decades after 1968, and in the broader context of Alexander's other writings and interviews.

[Molly Wright Steenson](#), as part of her [2014 dissertation](#), has a 66-page digest of Alexander's work between 1962 and 1968. Her deep reading was reflected in a 2009 recorded presentation on "Loving and Hating Christopher Alexander". Generally speaking, interaction

Categories

Select Category ▼

Recent Posts

[Systems generating systems — architectural design theory by Christopher Alexander \(1968\)](#)

[Mediating spaces, rich research spaces and GIGA-mapping](#)

[Evolution of open source IBIS software](#)

[Reframing service systems methods as project-portfolio conversations: Appreciating the shift from structured methods to agile systems development](#)

[A Proposal for Collaboration on a Pattern Language for Service Systems](#)

[What is a system? \(and the challenges of definition\)](#)

[Filling in a non-editable PDF form](#)

[Why do power failures seem rare in our Toronto neighbourhood?](#)

[General Systems Yearbook 2013: Service Systems, Natural Systems – Sciences in synthesis](#)

[2013/10/07 Lectures at Aalto](#)

Enter search terms [Search](#)

[Twitter.com/daviding](#)



@CentrumSztuki
Współczesnej Zamek
Ujazdowski - Contemporary art
center features many video
installatio...

<http://t.co/2xwGwrPTmO> [4sq]
about 19 hours ago from
twitterfeed
[ReplyRetweetFavorite](#)

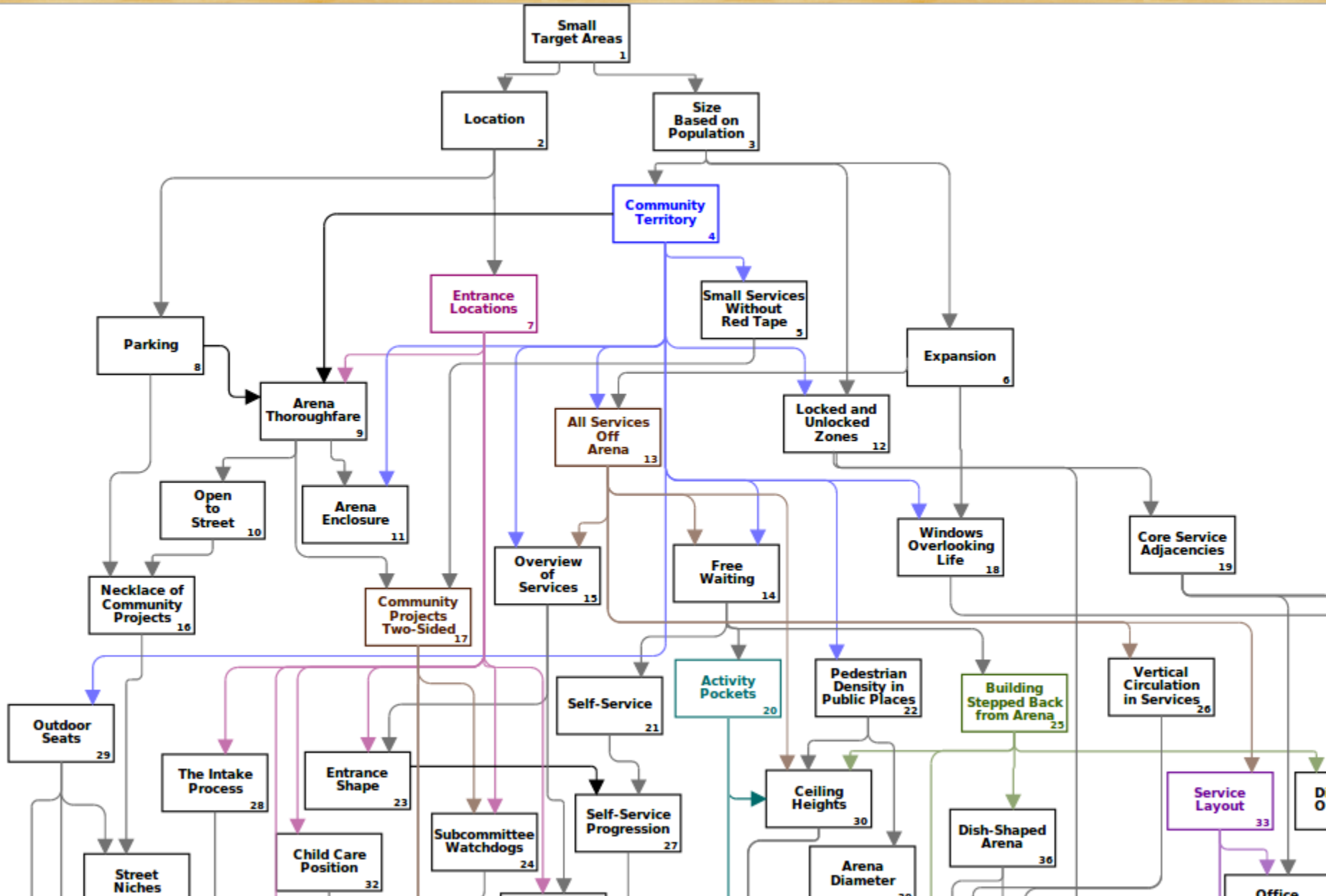
@WarszawaCentralna - S3
train from Chopin Airport to
Warsaw Centralna station
sorry short but slow. Bo...
<http://t.co/VkXRVGokSL> [4sq]
04:39:05 AM July 16, 2014
from twitterfeed
[ReplyRetweetFavorite](#)

@AirCanadaFlightAC872 -
Preoccupied printing maps of
Warsaw and Krakow, rushed
down and found comp ...
<http://t.co/674YETE0oM> [4sq]
03:44:12 PM July 15, 2014
from twitterfeed
[ReplyRetweetFavorite](#)

Sat. afternoon knock on door,
@PaulaFletcher30 said hello.
Long time neighbour,
represents our ward well. Not
yet campaigning, officially
21:21:21 PM July 12, 2014

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

 coevolving.com/maps/fed/1968_Multi-Service-Centers-cascade.svg



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](#)



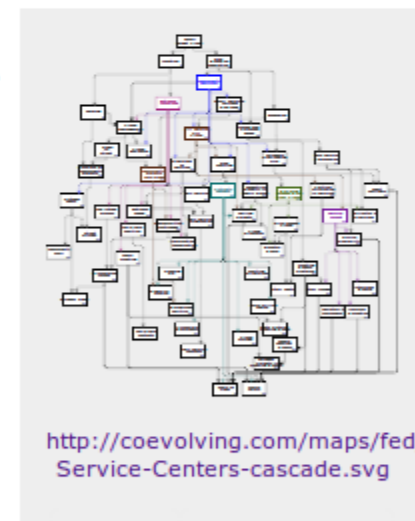
[Introduction \(Alexander et. al. 1968\)](#)

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

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

[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 of $34,000 \pm 20\%$.
2. **Location (1968)**: Service centers are located within two blocks of a major intersection.
3. **Size Based on Population (1968)**: The total size 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)**: No one service has a staff size greater than 12; each service is physically cohesive and autonomous; the services are loosely organized with respect to each other.

Community Territory (1968)

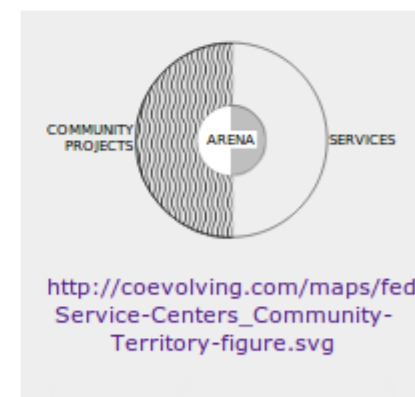
PATTERN

IF: Any multi-service center,

THEN:

1. The building should contain a major area which is established as community territory.
2. Community territory is distinct from the area devoted to services, but is interlocking with it.
3. Community territory contains two main components: An arena, and an area given over to community projects.

The arena is a public area, open to passers-by (whether or not they are visiting the service center), shaped in such a way as to encourage public discussions (both formal and informal), equipped with walls for day to day notices and poster, microphones, and loudspeakers. [p. 80]



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](#)



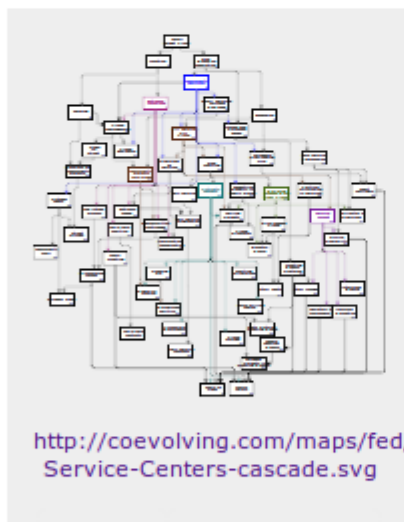
[Introduction \(Alexander et. al. 1968\)](#)

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

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

[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](#)

The Idea of a Pattern (Alexander et al. 1968)

If we examine the patterns as they are presented in full, in the Appendix, we shall see that each pattern has two parts: the PATTERN statement itself, and a PROBLEM statement. The PATTERN statement is itself broken down into two further parts, an IF part, and a THEN part. In full the statement of each pattern reads like this:

IF:X THEN:Z / PROBLEM:Y

X defines a set of conditions. Y defines some problem which is always liable to occur under the conditions Z. Z defines some abstract spatial relation which needs to be present under the conditions X, in order to solve the problem Y.

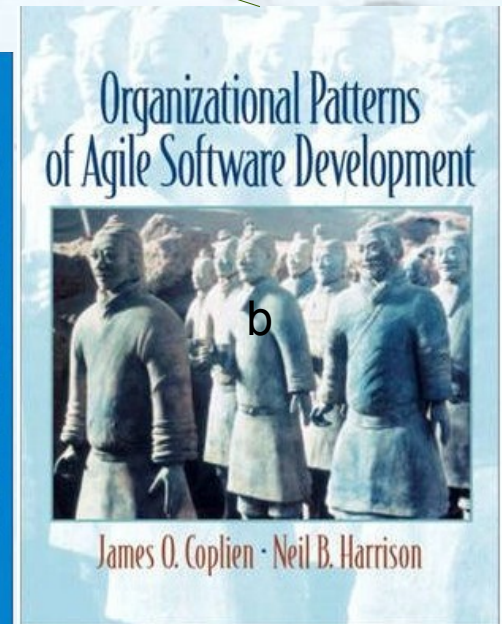
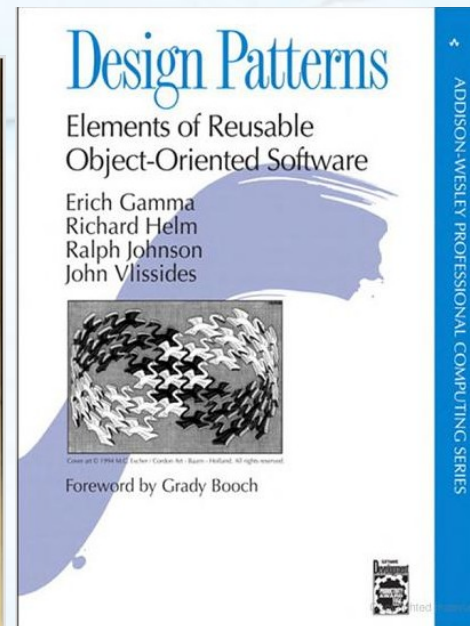
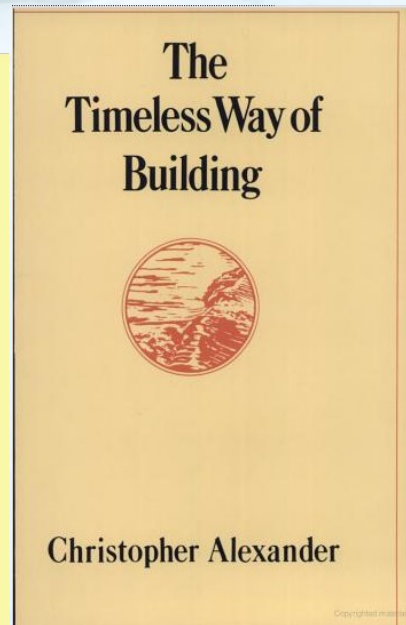
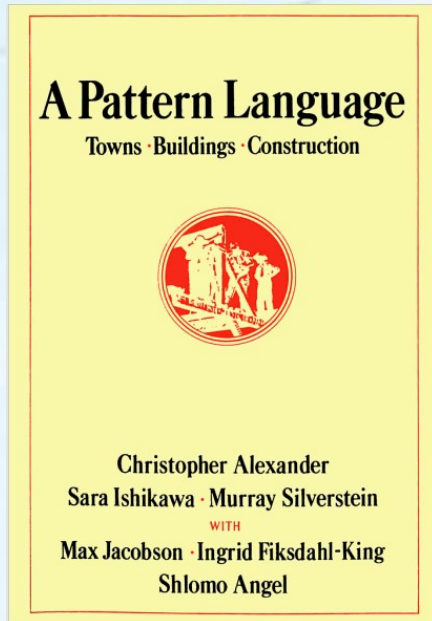
In short, IF the conditions X occur, THEN we should Z, in order to solve the PROBLEM Y. [p. 17]

This page is part of [A Pattern Language Which Generates Multi-Service Centers \(1968\)](#)



An evolution of pattern languages across domains

2005 <http://books.google.com/books?id=6K5QAAAAMAAJ> ;
<http://orgpatterns.wikispaces.com/>



1994 <http://books.google.com/books?id=6oHuKQe3TjQC>

1979 <http://books.google.com/books?id=H6CE9hlbO8sC>

1977 <http://books.google.com/books?id=hwAHmktpk5IC>
; <http://www.patternlanguage.com/>

127 INTIMACY GRADIENT**

... 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.

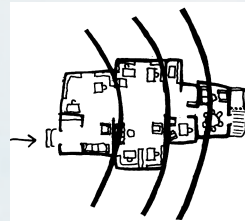
Source: Christopher Alexander et. al. 1997, *A Pattern Language: Towns, Building, Construction*, Oxford Press.-

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.

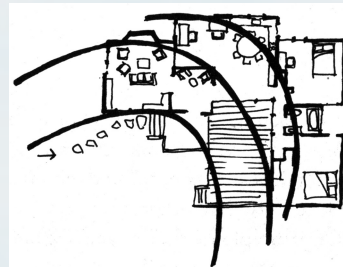
In an office the sequence might be: entry lobby, coffee and reception areas, offices and workspaces, private lounge.



Office intimacy gradient.

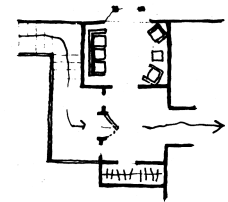
In a small shop the sequence might be: shop entrance, customer milling space, browsing area, sales counter, behind the counter, private place for workers.

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**

. . . 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.

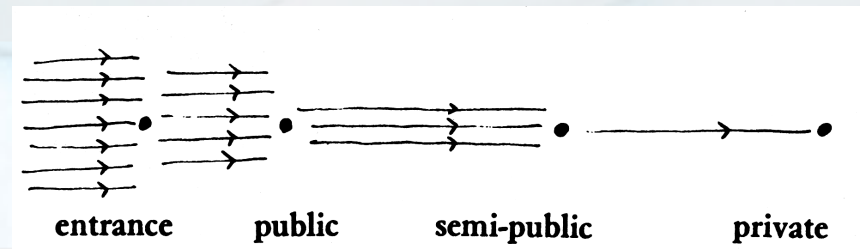
* * *

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.

Source: Christopher Alexander et. al. 1997, *A Pattern Language: Towns, Building, Construction*, Oxford Press.-

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. . . .

127 INTIMACY GRADIENT**



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 to it.

96 NUMBER OF STORIES *

107 WINGS OF LIGHT **

110 MAIN ENTRANCE **

Select Low Order Pattern and to it.

129 COMMON AREAS AT THE HEART **

130 ENTRANCE ROOM **

141 A ROOM OF ONE'S OWN **

142 SEQUENCE OF SITTING SPACES *

August 26, 2004

Intimacy Gradient and Other Lessons from Architecture

A number of my posts have been about integrating different domains to understand how human behavior should be incorporated in the design of [Dunbar Number](#) in sociology, and both [Four Kinds of Privacy](#) and [Work in the Cryptography Field](#). The topic of this post comes from the

In order to provide for [Progressive Trust](#), you need to establish what the "Intimacy Gradient".

The concept of Intimacy Gradient comes from architect Christopher Alexander's [Language: Towns, Buildings, Construction](#). (Oxford University Press)

Pattern #127 - Intimacy Gradient:

Conflict: Unless the spaces in a building are arranged in a way that corresponds to their degrees of privateness, the visits made by guests, clients, family, will always be a little awkward.

Resolution: Lay out the spaces of a building so that they begin with the entrance and the most public parts of the building, and finally to the most private areas.

In architecture there are always some areas of the house or building, the living room, the atrium, etc., and areas that are more private, bedrooms, and offices. In a good design there is some marker of these areas -- it might be a difference in ceiling height, a stairway leading to the entrance. As an example, in the classical Japanese tea house, you

Failure to respect the Intimacy Gradient results in uncomfortable behavior about a Frank Gehry building at Case Western Reserve University:

I asked many of the graduate students how they felt about "Horrible," said one. "Like living in a refrigerator" said another. "The comfortable offices and gathering places, and had the most private. Now everything is so sterile, and the acoustics so bad, the spaces are together. I have to go outside if I want any privacy."

The Intimacy Gradient is also used in other media. As I noted in my review of [Hand Circus](#):

When we arrived, we were led down the side of the theatre and all of a sudden I noticed that it looked like we were all being led backstage. We curve around and all of a sudden see an entrance -- maybe 5 foot tall requiring most of us to duck. We duck through and to our surprise, we are have walked through a fridgerator, and we are on the stage!



The Intimacy Gradient is also used in other media. As I noted in my review of [Seven Fingers of the Hand Circus](#):

When we arrived, we were led down the side of the theatre and all of a sudden I noticed that it looked like we were all being led backstage. We curve around and all of a sudden see an entrance -- maybe 5 foot tall requiring most of us to duck. We duck through and to our surprise, we are have walked through a fridgerator, and we are on the stage!

One of the 7 players welcomes us, and another offers random people a glass of tea as we walk across the stage to our seats. The stage is set like a city loft, with a tv, some couches, a bed, a bathtub and shower, a kitchen, and of course the fridgerator we entered through. On the stage, and chatting to members of the audience are the 7 cast members, all wearing comfortable looking white shorts or athletic and white t-shirts.

The audience arrives over 30 minutes and the 7 players act as if we are guests of their loft, serving some of us tea, chatting, sweeping the floor, etc.

Entering through the refrigerator door raised the intimacy of the experience for the audience of that circus. Thus in spite of it being produced in a large auditorium it felt as up-close and personal as did the much smaller [Circus Contraption](#).

The Intimacy Gradient exists in movies as well -- anywhere you see a scene taking place in a public space that transitions down through smaller and tighter shots ultimately to a closeup of a face it is much more intimate then just cutting to the closeup.

In social software design, there also needs to be an Intimacy Gradient. One of the problems with Wikis is that there is often very little transition between public and intimate, and doing so can be quite jarring. SocialText, a Wiki service vendor, is aware of this problem and is ["seeking to add more layers to the 'intimacy gradient', without recreating the highly structured collaboration tools that exist today"](#). Ross Mayfield outlines [this possible future Intimacy Gradient for SocialText](#):

- The broadest tier is a guest space, available to all
- The second tier is a knowledgebase, accessible to all employees and contractors
- The third tier is product development, for employees and contractors bound by a confidentiality agreement
- The fourth tier is for the core management team to share confidential financial and HR information.

The Hillside Group – Design Patterns



Site Search

[Hilltop](#) [Books](#) [Contact](#) [Conferences](#) [Patterns](#) [Vision](#) [Wiki](#)

[MAIN MENU](#)

► Hilltop

► Books

► **Contact**

► Conferences

► Patterns

- ☐ Books
- ☐ Patterns Catalog
- ☐ About Patterns
- ☐ TPLoP
- ☐ Education
- ☐ Mailing Lists
- ☐ Writing
- ☐ Tools
- ☐ Links

 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 [patterns](#) and [pattern languages](#). If you are new to patterns, James Coplien and Richard Gabriel have created a succinct [pattern definition](#).



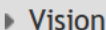

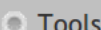
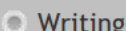
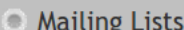
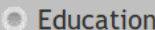
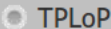
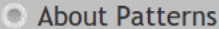
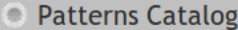
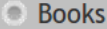
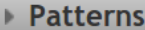

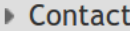
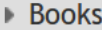

Patterns and Pattern Languages are ways to describe best practices, good designs, and capture experience in a way that it is possible for others to reuse this experience. The Hillside Group takes pleasure in sponsoring many different [PLoP conferences](#) that are provided for the betterment of the pattern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

Fundamental to any science or engineering discipline is a common vocabulary for expressing its concepts, and a language for relating them together. The goal of patterns within the software community is to create a body of literature to help 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 common pattern language for conveying the structures and mechanisms of our architectures allows us to intelligibly reason about them. The primary focus is not so much on technology as it is on creating a culture to document and support sound engineering architecture and design.



The Hillside Group – Software (Design) Pattern (Definition)

[hillside.net/patterns/50-patterns-library/patterns/222-design-pattern-definition](#)



DESIGN PATTERNS LIBRARY

Software Patterns

by James O. Coplien
Bell Laboratories
Naperville, Illinois

Patterns are a recent software engineering problem-solving discipline that emerged from the object-oriented community. Patterns have roots in many disciplines, including literate programming, and most notably in Alexander's work on urban planning and building architecture. (Alexander, 1977).

The goal of the pattern community is to build a body of literature to support design and development in general. There is less focus on technology than on a culture to document and support sound design. Software patterns first became popular with the object-oriented [Design Patterns book](#) (Gamma et. al., 1995). But patterns have been used for domains as diverse as development organization and process, exposition and teaching, and software architecture. At this writing, the software community is using patterns largely for software architecture and design.

Here is a pattern used in Lucent telecommunication products such as the Switching System® (extracted informally from Adams, 1996):.

Description

Name: *Try All Hardware Combos*

Problem: *The control complex of a fault-tolerant system can arrange its subsystems in many different configurations. There are many possible paths through the subsystems. How do you select a workable configuration when there is a faulty subsystem?*

Context: *The processing complex has several duplicated subsystems including a CPU, static and dynamic memory, and several busses. Standby units increase system reliability. 16 possible*

Example pattern – Lucent Telecommunications product

Description

Name: Try All Hardware Combos

Problem: The control complex of a fault-tolerant system can arrange its subsystems in many different configurations. There are many possible paths through the subsystems. How do you select a workable configuration when there is a faulty subsystem?

Context: The processing complex has several duplicated subsystems including a CPU, static and dynamic memory, and several busses. Standby units increase system reliability. 16 possible configurations (64 in the 4 ESS) of these subsystems give fully duplicated sparing in the 5ESS. Each such configuration is called a configuration state.

Forces: You want to catch and remedy single, isolated errors. You also want to catch errors that aren't easily detected in isolation but result from interaction between modules. You sometimes must catch multiple concurrent errors. The CPU can't sequence subsystems through configurations since it may itself be faulty. The machine should recover by itself without human intervention, many high-availability system failures come from operator errors, not primary system errors. We want to reserve human expertise for problems requiring only the deepest insights.

Solution: Maintain a 16-state counter in hardware called the configuration counter. There is a table that maps that counter onto a configuration state. The 5ESS switch tries all side 0 units (a complete failure group), then all side 1 units (the other failure group), seeking an isolated failure. When a reboot fails, the state increments and the system tries to reboot again. The subsequent counting states look for multiple concurrent failures caused by interactions between system modules.

Resulting Context: Sometimes the fault isn't detected during the reboot because latent diagnostic tasks elicit the errors. The pattern Fool Me Once solves this problem. And sometimes going through all the counter states isn't enough; see the patterns Don't Trust Anyone and Analog Timer.

Rationale: The design is based on hardware module design failure rates (in Failures in a trillion (FITs)) of the hardware modules. The pattern recalls the extreme caution of first-generation developers of stored program control switching systems.

This is a good pattern because:

- *It solves a problem:* Patterns capture solutions, not just abstract principles or strategies.
- *It is a proven concept:* Patterns capture solutions with a track record, not theories or speculation.
- *The solution isn't obvious:* Many problem-solving techniques (such as software design paradigms or methods) try to derive solutions from first principles. The best patterns generate a solution to a problem indirectly--a necessary approach for the most difficult problems of design.
- *It describes a relationship:* Patterns don't just describe modules, but describe deeper system structures and mechanisms.
- *The pattern has a significant human component (minimize human intervention).* All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

A *pattern language* defines a collection of patterns and the rules to combine them into an architectural style. Pattern languages describe software frameworks or families of related systems.

Pattern Language and Systems Thinking?

A pattern doesn't exist apart from a pattern language; its first purpose is to establish connections to other patterns in the language ([Alexander1977], p. xii). But to understand pattern languages, you must first understand what a pattern is. We know this is recursive, and to understand recursion, you must first understand recursion. We must start somewhere, and we start here: with patterns.

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.

Some of these aspects of pattern don't come out in the popular literature, and you may not find them all in the same place in Alexander's definitions. But they are the key elements of what makes a pattern a pattern, and what makes it different from a simple rule. **A pattern is a rule:** the word *configuration* should be read as "a rule to configure." But it is more than just a rule; it is a special kind of rule **that contributes to the overall structure of a system, that works together with other patterns to create emergent structure and behavior.** [p. 14]

Alexander believes that order in any system fundamentally depends on the process used to build the system. This is why the fundamental process is important (see the section *PIECEMEAL GROWTH* (6.2)).

It is important that each step preserves structure and gradually adds local symmetries, and the organization unfolds over time. It is step-by-step adaptation with feedback. Simply following the pattern language doesn't give you a clue about how to handle the feedback. So that's why the fundamental process exists: to give complete freedom to the design process to attack the weakest part of the system, wherever it may be.

However, the fundamental process cannot work on a human scale without some kind of cognitive guide that is built on experience and which can foresee some of the centers that must be built. That's what patterns are: essential centers.

If unfolding is important, how do you know what order to unfold things? The sequence is crucial. You want a smooth, structure-preserving unfolding. It shouldn't feel like "organizational design."

So, what a sequence does is:

- Preserves structure;
- Keeps you doing one thing at a time;
- Takes the whole organization into account at each step;
- May be repeated tens of thousands of times.

Sequences take you into unpredictability, and into circumstances you handle with feedback, always in the context of the whole organization. Sequences are where generativity comes from. [p. 37]

Review:

The Art of Observation: Understanding Pattern Languages

Werner Ulrich

University of Fribourg, SWITZERLAND;

The Open University, UNITED KINGDOM

wulrich@gmx.ch

**The Timeless Way of Building. Book by Christopher Alexander.
Published by Oxford University Press, New York, 1979, xv+552 pp., ISBN
0-19-502402-8, USD 34.65.**

Suggested Citation: Ulrich, W. (2006). The art of observation: Understanding pattern languages--A review of Christopher Alexander's *The Timeless Way of Building* (New York, Oxford University Press, 1979). *Journal of Research Practice*, 2(1), Article R1. Retrieved [date of access], from <http://jrj.icaap.org/index.php/jrp/article/view/26/46>

Christopher Alexander's book, *The Timeless Way of Building*, is probably the most beautiful book on the notion of quality in observation and design that I have been reading since Robert Pirsig's (1974) *Zen and the Art of Motorcycle Maintenance*. It was published in 1979, when Alexander was a professor of architecture at the University of California, Berkeley, where I was at that time studying. Although I was aware of some of Alexander's famous articles such as "A city is not a tree" (Alexander, 1965), the book (Alexander, 1979) never quite made it to the top of my reading list. This remained so until recently, when I met a software developer who enthusiastically talked to me on a book he was currently reading, about the importance of understanding design patterns. He was talking about the very book I had failed to read during my Berkeley years and which, as I now discovered, has since become a cult book among computer programmers and information scientists, as well as in other fields of research. I decided it was time

1. The Quality without a Name

... the essence of the Quality without a Name consists in the idea of design patterns that are alive and which, if identified in sufficient number, can be used to make up a whole pattern language for quality design.

2. Patterns that are Alive

As a rule, a room that does not have a window place lacks quality; its windows are just holes in the wall.

3. The Idea of a "Pattern Language"

... patterns are not arbitrary design ideas but can and need to be identified and verified through careful observation. Furthermore, patterns become meaningful only within a hierarchy of interdependent patterns, in which each pattern helps to complete larger (more generic) patterns within which it is contained, and in turn is further completed by smaller (more specific) patterns that it contains.

4. Against Modular Architecture

The way a pattern language works is not through a process of addition or combination of preformed parts of a design, but through a sequential process of unfolding, in which each pattern is developed in the context of the whole that is given by previously unfolded patterns ...

Design thus resembles more the evolution of an embryo than the drawing of an architectural plan. It is a process of growth--of increasing differentiation--with the pattern language operating as its genetic code. No application of a pattern will ever generate exactly the same result, for the result depends on the context generated by the previous stages of growth. This is different from conventional architectural design, in which the details of a building are made from identical, modular parts (e.g., prefabricated windows).

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.

Quicklinks

[Main Page](#)
[Editors' Note](#)
[Meet the Editors](#)
[Acknowledgements](#)
[How to Read the SEBoK](#)
[Download SEBoK PDF](#)
[Cite the SEBoK](#)
[Release History](#)
[Copyright Information](#)
[About the SEBoK](#)
[Go to Sandbox](#)

Outline

- [Table of Contents](#)
- [Part 1: SEBoK Introduction](#)
- [Part 2: Systems](#)
 - [Systems Fundamentals](#)
 - [Systems Science](#)
 - [Systems Thinking](#)
 - [What is Systems Thinking?](#)
 - [Concepts of Systems Thinking](#)
 - [Principles of Systems Thinking](#)
 - [Patterns of Systems Thinking](#)
 - [Representing Systems with Models](#)
 - [Systems Approach Applied to Engineering](#)
- [Part 3: SE and Management](#)
- [Part 4: Applications of Systems Engineering](#)
- [Part 5: Enabling Systems Engineering](#)
- [Part 6: Related Disciplines](#)
- [Part 7: SE Implementation Examples](#)

Navigation

[Knowledge Areas](#)
[Topics](#)
[Use Cases](#)
[Case Studies](#)
[Vignettes](#)
[Glossary of Terms](#)
[Acronyms](#)

Page

Read

[View source](#)

Go

Search

Patterns of Systems Thinking

[Patterns of Systems Thinking](#)

This topic forms part of the [Systems Thinking](#) knowledge area (KA). It identifies systems [patterns](#) as part of the basic ideas of [systems thinking](#). The general idea of patterns and a number of examples are described. A brief conclusion discusses the maturity of [systems science](#) from the perspective of [principles](#) and patterns.

Contents [\[hide\]](#)

- 1 Systems Patterns
 - 1.1 Pattern Definitions and Types
 - 1.2 Basic Foundational Patterns
 - 1.2.1 Hierarchy and Network Patterns
 - 1.2.2 Metapatterns
 - 1.3 Systems Engineering Patterns
 - 1.4 Patterns of Failure: Antipatterns
 - 1.4.1 System Archetypes
 - 1.4.2 Software and Other Antipatterns
 - 1.5 Patterns and Maturity
- 2 References
 - 2.1 Works Cited
 - 2.2 Primary References
 - 2.3 Additional References
- 3 SEBoK Discussion

Systems Patterns

This section first discusses definitions, types, and pervasiveness of patterns. Next, samples of basic patterns in the form of hierarchy and [network](#) patterns, metapatterns, and [systems engineering](#) (SE) patterns are discussed. Then samples of patterns of failure (or "[antipatterns](#)") are presented in the form of system archetypes, along with antipatterns in [software](#) engineering and other fields. Finally, a brief discussion of patterns as maturity indicators is given.

Pattern Definitions and Types

The most general definition of pattern is that it is an expression of an observed [regularity](#). Patterns exist in both natural and artificial systems and are used in both [systems science](#) and [systems engineering](#) (SE). Theories in science are patterns. Building [architecture](#) styles are patterns.

[Engineering](#) uses patterns extensively.

Patterns are a representation of similarities in a set or class of [problems](#), [solutions](#), or systems. In addition, some patterns can also represent uniqueness or differences, e.g., uniqueness pattern or unique identifier, such as automobile vehicle identification number (VIN), serial number on a consumer product, human fingerprints, DNA. The pattern is that a unique identifier, common to all instances in a class (such as fingerprint), distinguishes between all instances in that class.

The term pattern has been used primarily in building architecture and urban planning by Alexander (Alexander et al. 1977, Alexander 1979) and in

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

Current Applications of Pattern Languages

[Global Village Constructor Set Pattern Language](#) for Open Source Ecology

[Group Pattern Language](#) at the Group Works Project

[Liberating Voices Pattern Language](#) for the Public Sphere Project

[Patterns in Federated Wiki](#) by Michael Mehaffy

[Scrum \(Organizational\) Patterns](#) by the Scrum Patterns Community

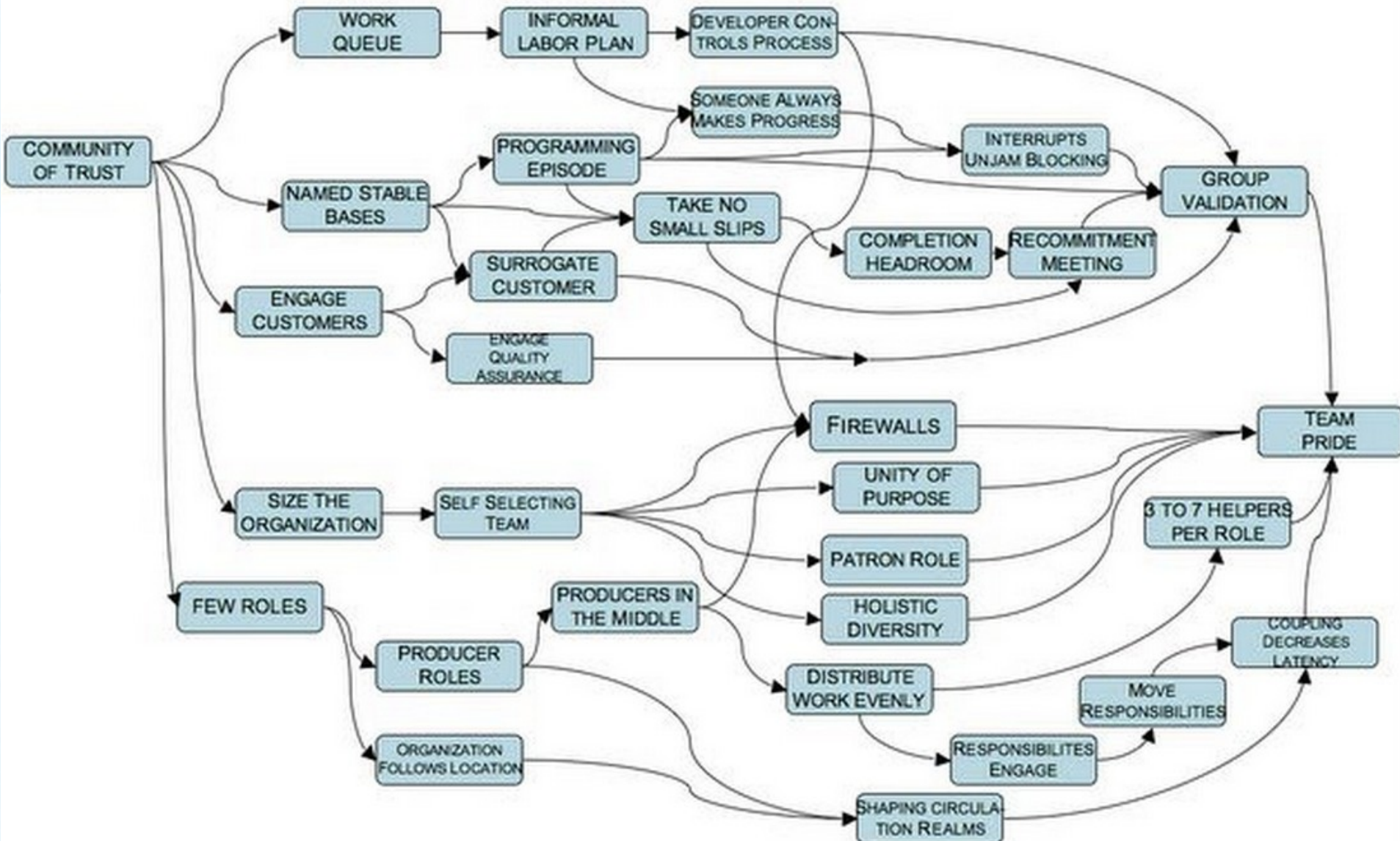
[Transition as a Pattern Language](#) in the Transition Movement

In addition, there are some communities exploring Federated Wiki.

[The Hidden History of Online Learning](#) led by Mike Caulfield



Scrum Patterns Summary



Source: <https://sites.google.com/a/scrumorgpatterns.com/www/scrumpatternssummary>

Jul 24, 2013 Posted by Javier Garzás in General | 1 comment 🇪🇸

Interview with Jim Coplien (1/2)

Jim Coplien (also known as Cope) is one of the key founders of software patterns movement and the agile development, a co-author of "Organizational Patterns of Agile Software Development" and "Lean Software Architecture for Agile Software Development" among others key books.

Now he works with the fathers of Scrum to facilitate its evolution as formalization.

Personally, it is a great honor for me to interview Jim, because part of my dissertation was about design patterns. And Jim is one of the fathers of the movement.

Jim, thank you very much for the interview.



1 – Most knows your work with patterns, but... Who is Jim? Where are you from? Background? Where

I'm Product Owner for the Scrum Patterns effort, Scrum PLoP® (<http://www.scrumplop.org>). It is, in some ways, an outgrowth of the organizational patterns work that started 20 years ago (<http://orgpatterns.wikispaces.com>). I'm proud of this work because it is the only body I know of that is chartered as a non-partisan group to evolve a rationalized definition of Scrum. [...]

There are three factors that make the Scrum Patterns special.

1. They adopt a systems thinking view of organizational transformation, rather than a rulebook approach. This means that we can get beyond technique to organizational structure and to principles and values, and really address the human issues that make complex development so hard. Patterns help us think in systems ways that are more or less the opposite of "root cause analysis."

2. The Scrum Patterns are shaped by the thinking at the foundations of Scrum and written first-hand by those great thinkers: Jeff Sutherland, Michael Beedle, Gabrielle Benefield, Jens Østergaard, and more.

3. They reflect input from all the major certifying entities, and where we lack engagement with key constituencies today we are always seeking to be inclusive with more folks.

I think it's important to understand that **what we're building isn't just a pattern catalog. You shop by paging through a catalog and choose one or two things to take home. We are building much more formal constructs called pattern languages. Pattern languages include sets of rules that constrain meaningful combinations of patterns according to a generative grammar, that can be used by the designer to generate a myriad of wholes.** What this means in layman's terms is that we are building a roadmap that can inspire organizations by showing them the many paths to building great Scrum teams. A pattern language requires judgment, insight, and adaptation on the part of its users. Very few of the publications currently called "patterns" have this generative ability. However, the inventor of patterns, Christopher Alexander, insists that this is an essential property of patterns. They compose with each other to create "morphological wholes." These Wholes are teams, value streams, relationships, cycles in time, and other structures in the development organization. [...]

Most engineering students think in terms of short time frames; a good mature engineer thinks ahead to how use and nature will cause a structure to weaken or become obsolete. Patterns attack that kind of entropy. Engineers building Japanese temples today plant trees that will be used to build their successors 200 years from now; patterns in construction lay a foundation for a good future by understanding the past.

Scrum Patterns Summary

The patterns without which Scrum is unlikely to work

These patterns are Scrum. In June 2008, founders of Scrum and Organizational Patterns met together with a small number of other experts from both camps to map out patterns from the book [Organizational Patterns of Agile Software Development](#) as they apply to projects under the Scrum framework. These patterns represent elements of the framework, as well as fundamental practices which dissect the framework into its fundamental underlying components. Because patterns work at the level of structure rather than cause-and-effect, we call these patterns Second-Level Scrum patterns, which can be compared to the First- and Third-level Scrum patterns described below.

These patterns are independent of software development per se. The group also agreed to patterns without which Scrum is unlikely to succeed, and about which Jeff Sutherland said he has incorporated into every Scrum he has started. These can be found on the page [Software Scrum Patterns](#).

The third and final part of the picture is the pattern language by Beedle et al. that describes the cause-and-effect components of Scrum. You can see a summary of those on the page [First-Level Scrum Patterns](#).

See the pattern language picture at the bottom of this page.

Pattern Name	Description	Scrum Notes
COMMUNITY OF TRUST	People are open enough to surface uncomfortable truths	Scrum is based on honesty, openness, and visibility
WORK QUEUE		You have a product backlog and a sprint backlog
NAMED STABLE BASES	Don't use continuous integration, because that makes it difficult for the team to have a shared vision of where the base stands. Integrate public integrations regularly	While continuous integration is the norm, you also have a delineated, agreed target where everything must converge in a coordinated way at the end of a sprint. Also, some architectural changes cannot be made piecemeal but are best handled by offline work on a branch administered by the source management system.
ENGAGE CUSTOMERS	Customers should be tightly coupled to the organization	Scrum has no customer role! However, no Scrum would be complete without one. Note that the Product Owner is not the customer: the Product Owner stands to optimize ROI for the enterprise, and though this role takes input from the customer, it is not the locus of caring for customer interests! However, also see SURROGATE CUSTOMER below.
SIZE THE ORGANIZATION	Start small and grow gradually	Small teams work best. Scrum talks about teams of seven plus or minus two people; over time, the smaller teams have been winning out over the larger ones.
FEW ROLES	The total number of roles is small	Of course, original Scrum has only three roles. However, because DOMAIN EXPERTISE IN ROLES is also important we admit a small number of specialized roles. If there are too many roles (areas of specialization) the specialization works against the process of finding out what you need to

FEW ROLES

The total number of roles is sm

INFORMAL LABOR PLAN

If developers need to do the m important thing now, then let developers negotiate among themselves or "just figure out the right thing to do" as regards short term plans, instead of master planning.

PROGRAMMING EPISODE

You have fixed-length sprints

STAND-UP MEETING

Get the team together for a regular short meeting to exchange status information.

SURROGATE CUSTOMER

Sometimes, if it is inconvenient impossible to talk to a real customer in real time, you need a local stand-in.

ENGAGE QUALITY ASSURANCE

Quality Assurance is not an afterthought

SELF-SELECTING TEAM

Appointed teams don't work. TI people should decide what people should be on which teams.

PRODUCER ROLES

If your organization has too many roles, but does not know which to eliminate, then identify roles as Producers, Supporters, or Deadbeats; eliminate the Deadbeats and combine some of the Supporters

ORGANIZATION FOLLOWS LOCATION

Units of a feather flock together the organization is colocated

DEVELOPER CONTROLS PROCESS

The developer calls the shots

115

days until
ScrumPLoP!

[Conference Home](#)
[Published Patterns Home](#)
[Works in Progress Home](#)
[Product Backlog](#)

▼ [Home](#)

[Pattern Map](#)

[A Vision](#)

▼ [Distributed Scrum](#)
[Pattern Language](#)

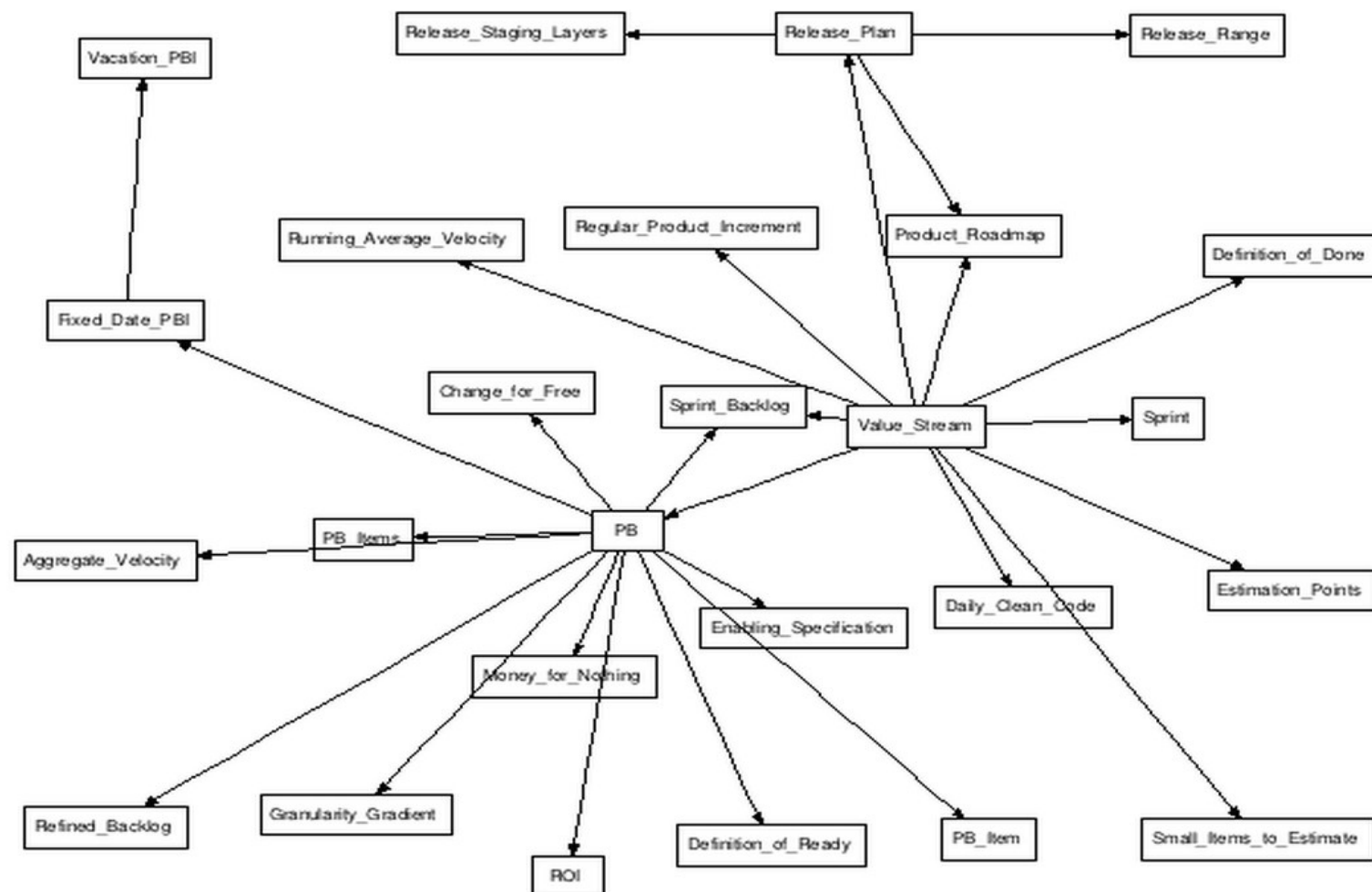
[Boot Camp](#)
[Quantum](#)
[Entanglement](#)
[Rotating Guru](#)
[Single Mediator](#)

▼ [Process Improvement](#)
[Pattern Language](#)

[Create Knowledge](#)
[Fair Memory](#)
[Happiness Metric](#)
[Historical](#)
[Retrospective](#)
[Impediment List](#)
[Learning Register](#)
[One Rung at a Time](#)

[Home](#) >

Pattern Map





- [Buy printed deck from 100fires.com](#)
- [Download free deck](#)
- [iPhone version](#)

[Donate](#)

Latest News

Successful Portland Workshop!

Last weekend we hosted another set of very successful workshop offerings around the Group Works...

Productive Eugene Work Session

Can you believe that after this packed weekend the four of us-- Daniel Lindenberg, Dave...

Portland Workshops - November 15-17, 2013

Hereby announcing a Group Works team offering...

What we mean by "pattern"

For the purposes of this project, a pattern is a feature we believe shows up repeatedly in [group processes](#) that result in deepening, connection, and a fulfillment of purpose.

Scope: Our scope is the realm of deliberative/dialogic group processes aimed at goals such as decision-making, input, feedback, strategizing, visioning, and conflict resolution, that take place in the context of meetings, conferences, and other convenings that have these goals. Thus there are many times when people gather together that are beyond our zone of exploration (weddings, soccer matches, choir practice, etc.), although some of the patterns we come up with might also sometimes apply beyond our chosen domain.

While recognizing that there is a lot of overlap between patterns of face-to-face and online interaction, we are focused on face-to-face settings.

While we believe that many of these patterns show up cross-culturally, we do not make the claim that our pattern language is universal. This language is being written by a cross-section of North Americans from a variety of group backgrounds (political activism, communal living, and other alternative cultures; higher education; corporate experience in finance, software, and other fields; religious organizations; nonprofit management; indigenous tribes; public agency work; etc.). That diversity lends strength but falls far from addressing all places and groups. Any session takes place within a specific cultural context, and our assumption is that users of this language will take what inspires them and adapt appropriately.

Range: If you think about everything that goes into making a group conversation fulfilling, there is a vast range of things to pay attention to, from the most general to the most specific. In crafting this language, we ask people to focus on a particular middle section within this range, and hold your thinking at that level. We want to avoid being too general: For example, values such as democracy and cooperation, or principles such as Schwarz's "valid information" and "free and informed choice," are assumed or embedded rather than explicitly considered in the body of this work. We also want to avoid being too specific: There is a level of detail that is already well-represented in the existing literature and that we do not seek to replicate, namely:

- listing and explanations of methods (e.g. [The Change Handbook](#));
- catalogues of tools and techniques.

Mailing Lists

Join "Pattern-Hi" discussion list:

[Pattern-Hi archives](#)

Join "Pattern-Lo" announcement list:

[Pattern-Lo archives](#)

Group Works: A Pattern Language for Bringing Life to Meetings and Other Gatherings



1. Intention

Serving and attending to the larger purpose for the gathering and how it is manifested, including addressing its longer term meaning and consequence. Why are we here, what's our shared passion, and what are we aiming to accomplish.

Commitment
Invitation

Priority Focus
Purpose

Setting Intention



2. Context

Understanding and working with the broader context and circumstances both in place and in culture.

Aesthetics of Space
Circle
Gaia

Group Culture
History and Context
Nooks in Space and Time

Power of Place
Whole System in the Room



3. Relationship

Creating and maintaining quality connection with each other, honouring our full selves, and recognizing power relations. Includes being authentic and sometimes foregrounding emotional needs in the moment rather than task.

Appreciation
Breaking Bread Together
Celebrate
Good Faith Assumptions

Honour Each Person
Hosting
Power Shift

Shared Airtime
Tend Relationships
Transparency



4. Flow

Covers rhythm, energy, and pacing. When we do what and for how long. Things to pay attention to both in anticipating the event and in responding to circumstances in the moment, to support movement along the intended trajectory toward the desired outcome.

Balance Process
and Content
Balance Structure
and Flexibility
Closing
**Divergence and
Convergence Rhythm**

Follow the Energy
Iteration
Opening and Welcome
Preparedness
Reflection/Action Cycle
Rest

Right Size Bite
Ritual
Seasoned Timing
Subgroup and Whole Group
Trajectory



5. Creativity

Using multiple intelligences and a variety of modes to open up creative possibilities.

Challenge
Expressive Arts

Improvise
Mode Choice

Playfulness
Power of Constraints

Generate Possibilities



6. Perspective

Noticing and helping the group more openly and thoughtfully explore different ways of seeing an issue. Watching, understanding, and appreciating divergent viewpoints, ideas, values and opinions. The key is in how you look at something.

Common Ground
Embrace Dissonance
and Difference
Fractal

Go Meta
Seeing the Forest,
Seeing the Trees
Time Shift

Translation
Unity and Diversity
Value the Margins
Viewpoint Shift



7. Modelling

The essential skills and responsibilities for both facilitator and participants, to demonstrate good group practice and ensure the process goes well. Includes monitoring, nurturing and mentoring the group, enabling their effective personal and collective self-management.

Appropriate Boundaries
Courageous Modelling
Discharging
Dwell with Emotions
Guerrilla Facilitation

Holding Space
Listening
Mirroring
Not About You
Self-Awareness

Shared Leadership and Roles
Simplify
Taking Responsibility
Witness with Compassion



8. Inquiry & Synthesis

Discovering coherence and moving toward convergence. From gathering information to exploring knowledge to arriving at understanding, shared meaning, consensus, or clear outcomes.

Deliberate
Distilling
Experts on Tap
Feedback
Go Deeper

Harvesting
Inform the Group Mind
Inquiry
Mapping and Measurement

Moving Toward Alignment
Naming
Story
Yes, and



9. Faith

Trusting and accepting what happens in a spirit of letting go and letting come. The mystery, synergy, and ineffable, complex magic of emergence. You can invite it, but you can't control it. Felt as a deep sense of connection not only to those assembled and to the work's purpose but to the larger universe as well.

All Grist for the Mill
Dive In
Emergence

Letting Go
Magic
Presence

Silence
Spirit
Trust the Wisdom of the Group

Source: http://groupworksdeck.org/patterns_by_category



Liberating Voices!

A Pattern Language for Communication Revolution

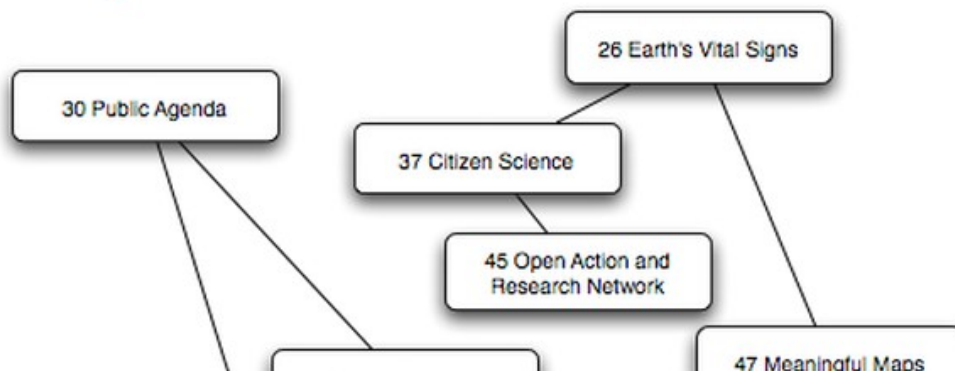
We are now in the 7th year of a 12-year project to help understand, motivate and inform the worldwide movement to establish full access to information and communication — including the design and management of information and communication systems.

We're working together to develop one or more "pattern languages" which can help people think about, design, develop, manage and use information and communication systems that more fully meet human needs now — and in the future.

Our "pattern language" is a holistic collection of "patterns" that can be used together to address an information or communication problem. Each "pattern" in this pattern language, when complete, will represent an important insight that will help contribute to a communication revolution.

The book that contains the first version of the Liberating Voices pattern language is now [available from MIT Press](#). There are several "context" chapters in addition to the 136 patterns. The patterns are also available on this site, some in slightly different form.

Using a Network of Patterns



ACCESS TO PATTERNS

[Contributors](#)

[Liberating Voices \(book\) Patterns](#)

[Pattern Pool \(all submissions\)](#)

[References in book patterns](#)

[I feel lucky! Look at a random pattern](#)

[Pattern Language Mailing List](#) Join the Liberating Voices! discussion!

SEVEN RANDOM PATTERNS

[Transforming Institutions](#)

[Public Agenda](#)

[Digital Emancipation](#)

[Transparency](#)

[Community Building Journalism](#)

[Engaged Tourism](#)

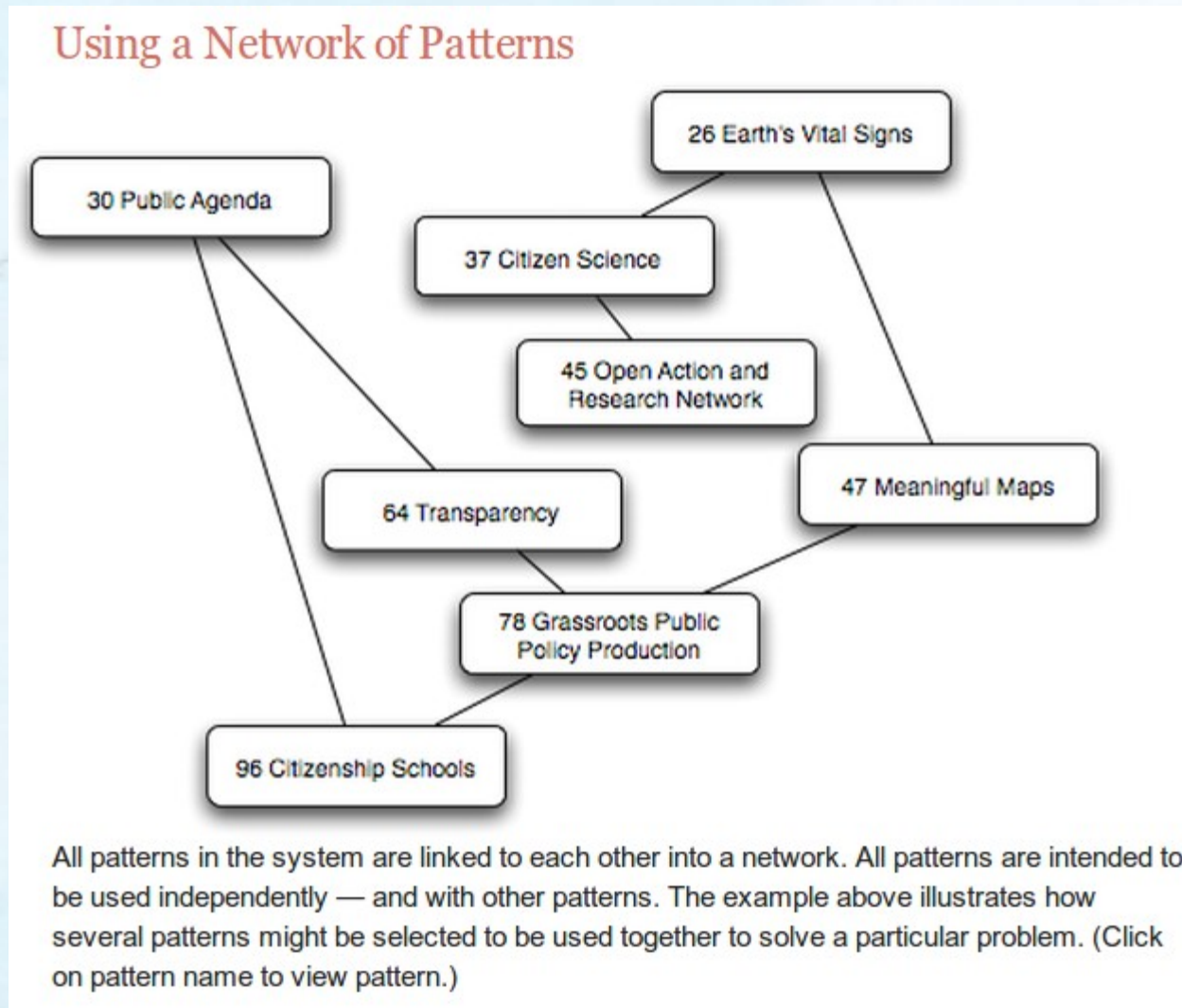
[Whistle Blowing](#)

PATTERN LANGUAGE DOCUMENTS

[Pattern Cards for workshops \(under development\)](#)

[Statement of Purpose](#)

All patterns in the system are linked to each other into a network.
All patterns are intended to be used independently — and with other patterns.



MEANINGFUL MAPS

in [community action](#) [education](#) [engagement](#) [Orientation](#) [product](#) [social critique](#)

Pattern number within this pattern set: 47



Andy Dearden
Sheffield Hallam University
Scot Fletcher
Handspring design, Sheffield, UK

Problem:

People are often unaware of the state of the world around them — especially "invisible", second-order relationships. Many of the important issues for the community, the environment and for humanity are improve the world, we must understand the current situation, highlight the important factors, and help understand the issues. How can we collect up to date information and present it in a way that people understand?

Context:

This pattern is useful for community groups, advocacy groups and campaigns that are working to im around them. This might be about local environmental quality, promoting international respect for hun needs such as clean- water or nutrition, or access to opportunities, in a neighbourhood, city or count target their resources carefully to achieve the maximum impact. They also want to communicate the encourage others to support their work. To be effective they need to reveal hidden relationships.

Discussion:

To act effectively to improve a situation, we need to understand that situation. Using a map can act i monitoring the current situation in an area and showing how this is changing; and as a way of presen other people to raise their awareness and encourage them to support the work. The image above is t Apple Map of New York City <http://www.greenapplemap.org>

EARTH'S VITAL SIGNS

in [case study](#) [collaboration](#) [education](#) [engagement](#) [globalism](#) [localism](#) [Orientation](#) [policy](#) [research for action](#) [servi](#)
[Theory](#)

Pattern number within this pattern set: 26



Jenny Frankel-Reed
Problem:

Societys great scientific capacity to measure and interpret the world and the role of humans in nature has failed translate into improved environmental stewardship. Modern environmental challenges are often difficult to see, time and space from their sources, and threaten global consequences. The increasing complexity and chronic acute nature of today's environmental problems requires a revolution of decisionmaking the systematic integrat earths vital signs.

Context:

Signals detected by scientists about earth's natural patterns and processes and the impacts of humans on the processes are earth's signs - indicators of what can be seen as either ecological health or the capacity of the e accommodate human demands. The conditions of earth's systems tend to be worsening on a global scale, but dramatically from place to place. Human decisions about how to live on earth drive these trends and can potent reverse their negative directions.

Policymakers, public interest organizations, universities, and governments can utilize earth's signs to better ma human and environmental well-being. Policymakers' decisions about sustainable practices in land- and resource dependent sectors can be backed by scientific understanding about the effects of policies on resources. Citizen demand better environmental stewardship from their leaders at local to global scales with improved access to a translation of relevant earth information at the proper scale. Governments and enforcement bodies can strength monitoring capabilities and base development decisions on the latest information about trends in human impact


Global Village Constructor Set Pattern Language

GVCS Pattern Language

← → ↺ 🏠

opensourceecology.org/wiki/GVCS_Pattern_Language

☆ 🏠 d



OPEN
SOURCE
ECOLOGY

Navigation

Main page

Site Map

Wiki Instructions

Recent Wiki Changes

Blog

Open Source Ecology

About OSE

Support OSE

Contact Us

Mailing List

In The News

Events

Discussion Forums

FAQ

Toolbox

What links here

Related changes

Log in

Login with OpenID

Create account

Page

Discussion

Select Language ▼

Read

Edit

View history

Search

Go

Search

Custom Search

GVCS Pattern Language

Contents [hide]

1 Introduction

2 Icon Specifications and Design

3 Icon Categories

3.1 OSA Passive Produce

3.2 OSA Value-added food products

3.3 OSA Value-added Processing Equipment

3.4 OSA Perennial Agriculture

3.5 OSA Material Products

3.6 OSA Agricultural Machinery

3.7 Construction

3.8 Fuel and Energy

3.9 Water Resources

3.10 Additional Icons

3.11 RepLab - the Open Source Fab Lab

4 Universal Power Components

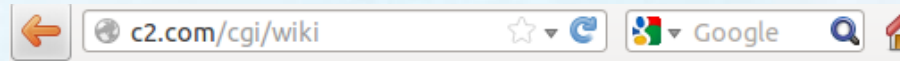
Introduction

[edit]

To facilitate communication about the [RCCS](#), it is useful to define a set of icons to communicate the components of the RCCS. The point of this is to address that:

- The RCCS is a well-bounded set of items

Wiki was invented to support pattern language collaborations



Front Page

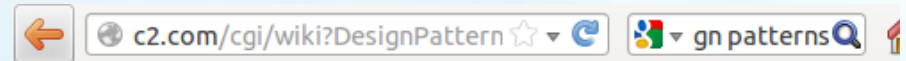
This [ContentCreationWiki](#) is focused on [PeopleProjectsAndPatterns](#) in [SoftwareDevelopment](#).

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 editing practice. Go there now to try it. (Please don't edit this page; changes here will likely be reversed within a few minutes).



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...

Why it is easier to find an [AntiPattern](#) than a [DesignPattern](#) or an [AmeliorationPattern](#) in this Wiki?

C2 Portland Pattern Repository → Hillside Group



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...

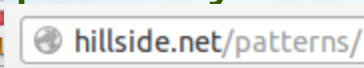
- [Pattern Language Catalog](#)

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 HILLSIDE GROUP

[Hilltop](#) [Books](#) [Contact](#) [Conferences](#) [Patterns](#) [Vision](#) [Wiki](#)

MAIN MENU

► [Hilltop](#)

► [Books](#)

► [Contact](#)

► [Conferences](#)

► **[Patterns](#)**

- [Books](#)
- [Patterns Catalog](#)
- [About Patterns](#)
- [TPLoP](#)
- [Education](#)
- [Mailing Lists](#)
- [Writing](#)
- [Tools](#)
- [Links](#)

► [Vision](#)

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

DESIGN PATTERNS LIBRARY

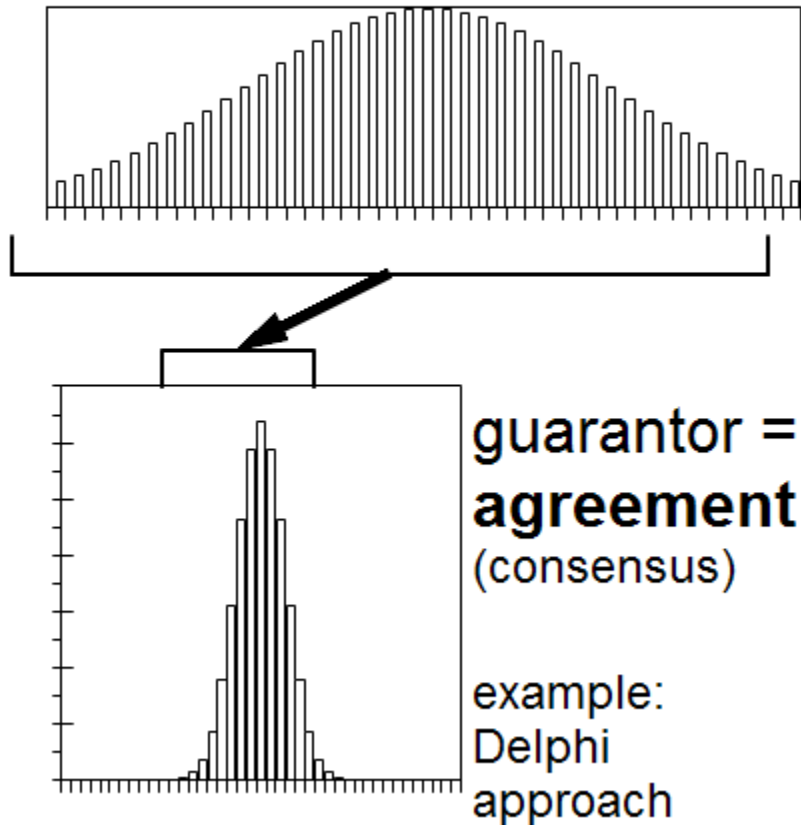
Welcome to the patterns home page. It is a source for information [languages](#). If you are new to patterns, James Coplien and Richard

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

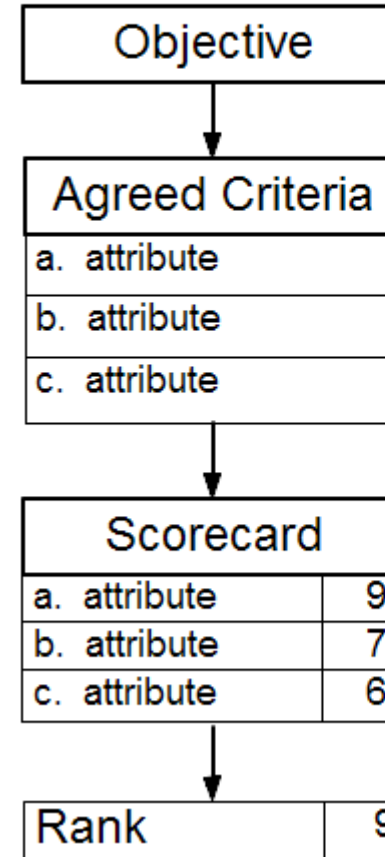
Fundamental to any science or engineering discipline is the need for a common vocabulary for expressing its concepts, and the Hillside Group is dedicated to creating a culture to document and support software developers resolve recurring problems in software development. Patterns help create a shared language about these problems and their solutions. The goal of patterns within the software community is that their relationships lets us successfully capture the understanding of good architectures that meet the need for a pattern language for conveying the structures and help us to intelligibly reason about them. The primary focus is on creating a culture to document and support software

Design of inquiring systems: Ways of knowing (1, 2)

The first way of knowing Inductive-Consensual IS



The second way of knowing Analytic-Deductive IS

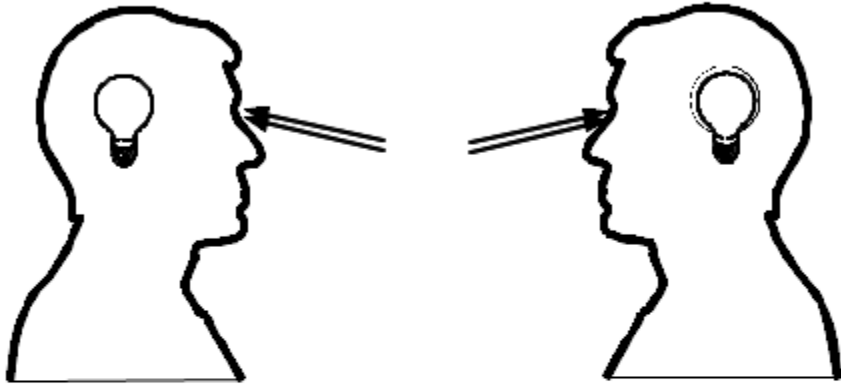


guarantor = logical consistency (fact nets)

example: finding the "best man" for the job

Design of inquiring systems: Ways of knowing (3, 4)

The third way of knowing Multiple Realities IS

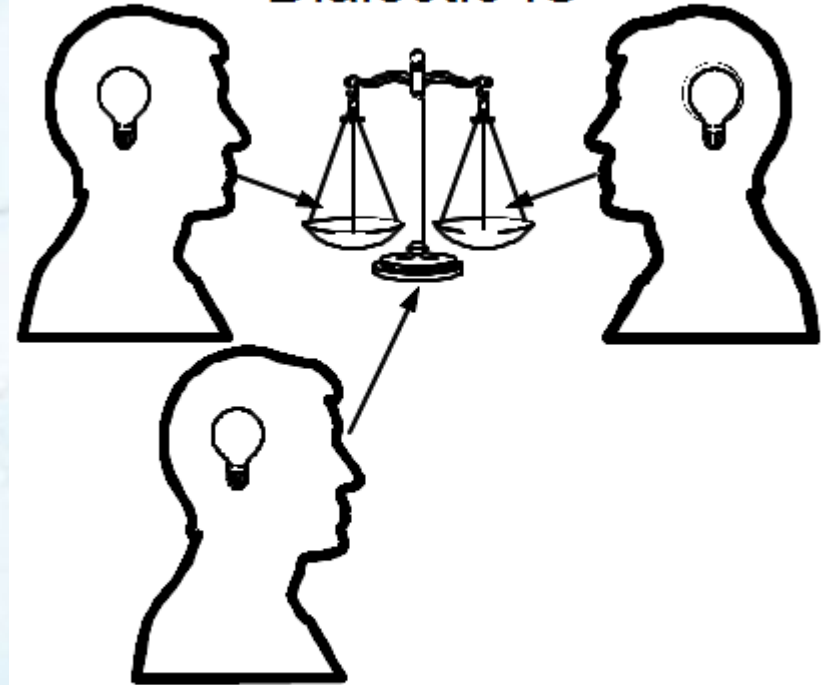


model + data as inseparable whole
For human beings to have experience or gain knowledge about the external world, something must be built into the internal structure of their minds ...

guarantor = (ability to see)
range of views (representations)

example: disciplinary views of the causes of the drug problem

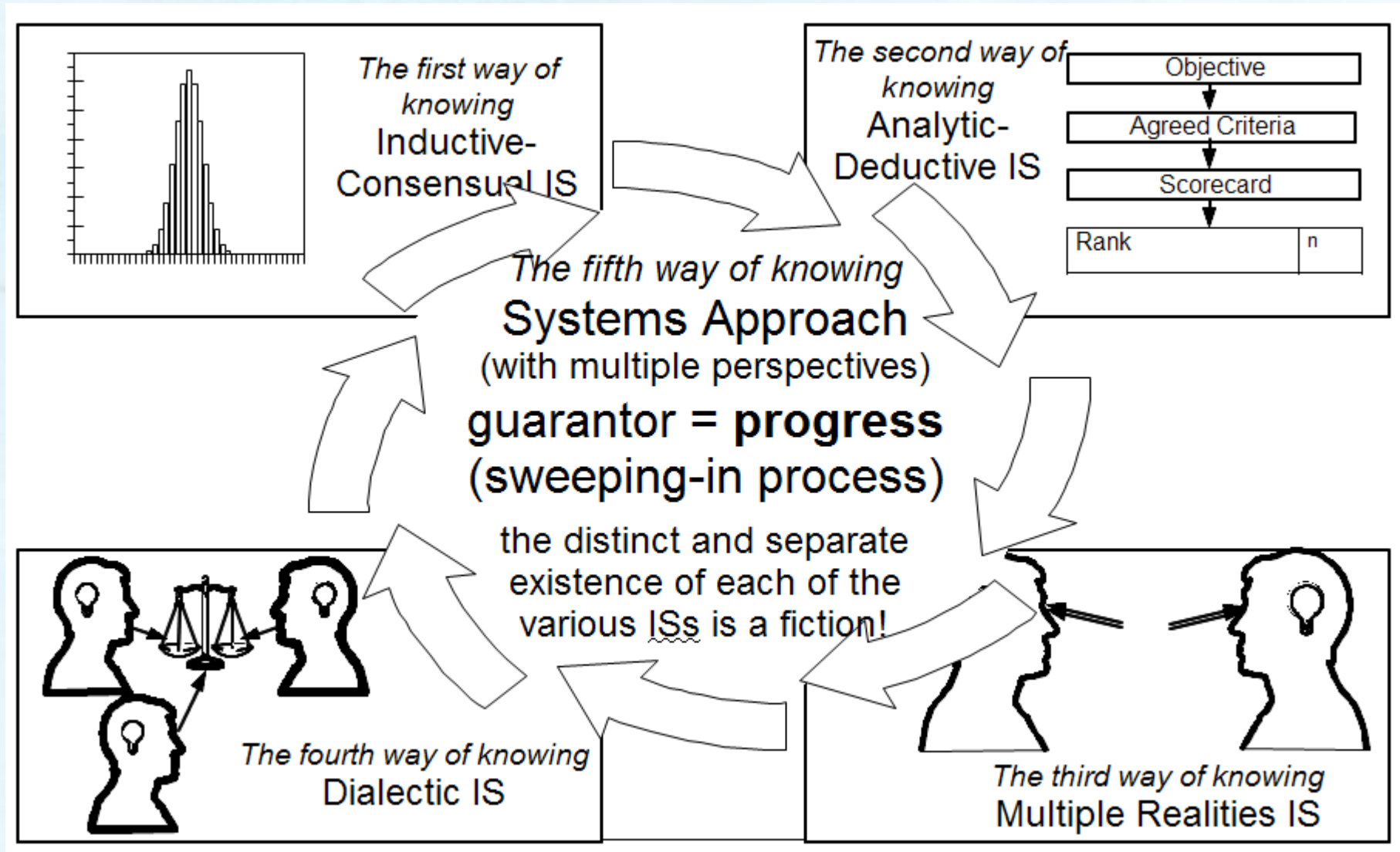
The fourth way of knowing Dialectic IS



guarantor = conflict

example: challenging assumptions of what skid row housing should be

Design of inquiring systems: Ways of knowing (5)



Source: Ian I. Mitroff, and Harold A. Linstone. 1993. *The Unbounded Mind: Breaking the Chains of Traditional Business Thinking*. Oxford U Press.

Agenda

1. Service Systems Thinking, In Brief

2. Conversations for Orientation

→ 3. Conversations for Possibilities

4. Conversations for Action

5. Conversations for Clarification

3.1 Multiple Perspectives Open Collaboration:
We could have federated authored content on open source platforms

3.2 Generative Pattern Language:
We could be reoriented for unfolding wholeness, layering systems of centers and/or with creating interactive value

3.3 SS MED: We could have trans-disciplinary cooperation on service systems improvement

3.4 Systems thinking: We could have service systems evolving from the systems thinking tradition

Conversations for Possibilities on Service Systems Thinking

With Service Systems Thinking at its inception, community participants can discuss potential futures for their collaborations.

For [Multiple Perspectives Open Collaboration](#),

1. We could have [Federated Authored Content](#) on open source platforms.

For [Generative Pattern Language](#),

2. We could be reoriented for [Unfolding Wholeness, Layering Systems of Centers and/with Creating Interactive Value](#).

For [Service Science, Management, Engineering and Design](#),

3. We could have [Transdisciplinary Cooperation on Service Systems Improvement](#).

For [Systems Thinking](#),

4. We could have service systems [Evolving from the Systems Thinking Tradition](#).

Federated Authored Content

The [Smallest Federated Wiki](#) technology enables authors to easily:

- (i) reference the work from other authors (with a [Page Drag Link](#) ↗);
- (ii) curate a neighborhood (see [Search Neighborhood Curation](#) ↗); and
- (iii) amend content on a personal version in the federation (see [Journal Fork Cache](#) ↗).

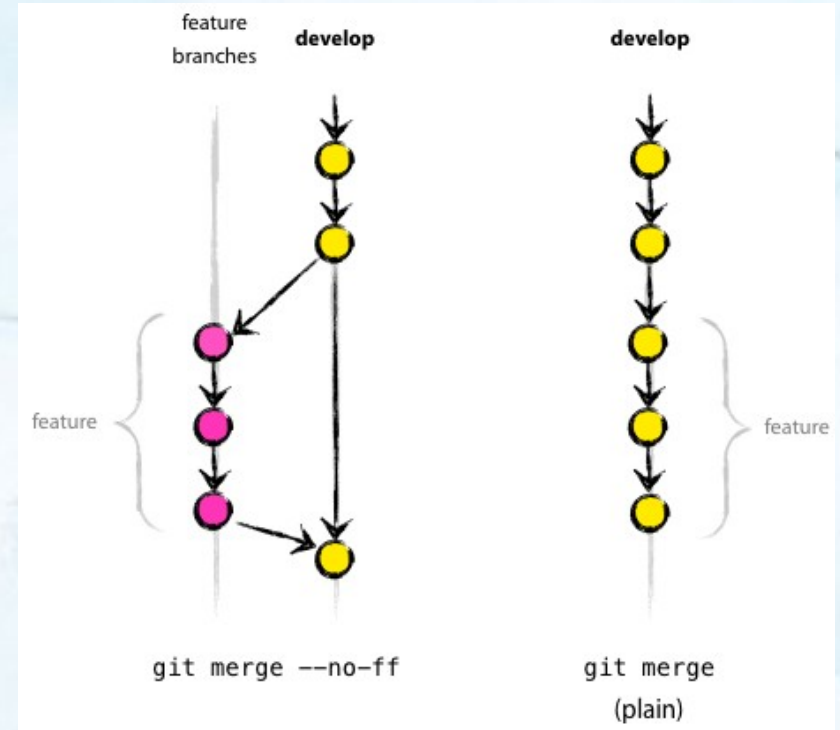
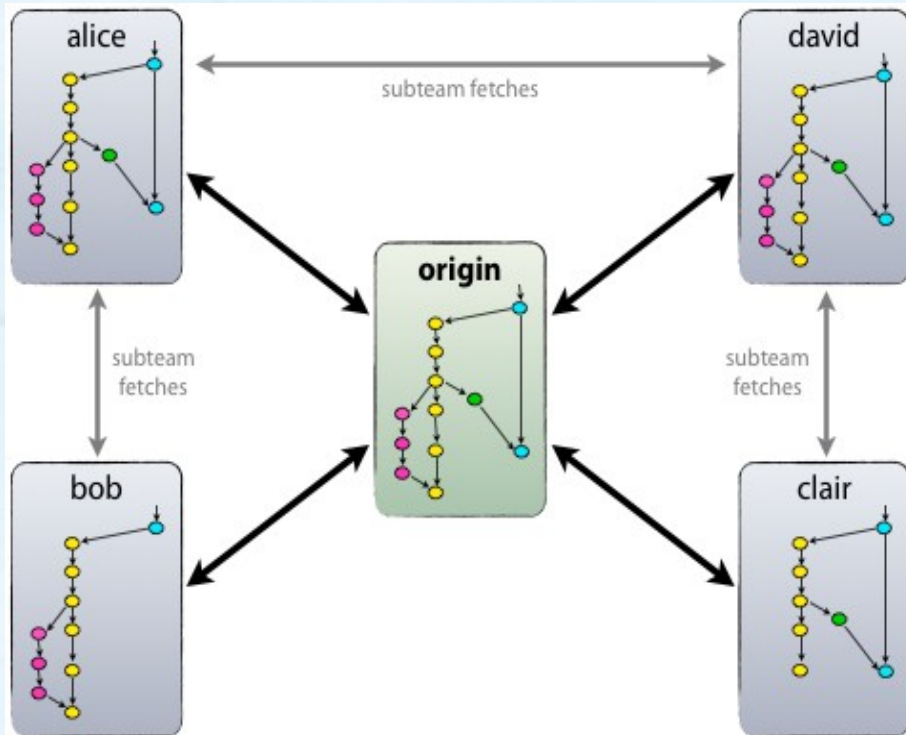
Each author has or her own wiki site in the federation. The [Wiki OpenShift Quickstart](#) outlines (i) an easy installation procedure on the rhcloud.com domain; or (ii) a slightly more complicated procedure for authors who have registered their own web domains.

Authors may first want to experiment with [How to Wiki](#) on a [Sandbox](#) ↗ (that periodically gets restored to its original state).

As an alternative to reading, [Federated Wiki Videos](#) ↗ show the technology in use.

The open source community around Smallest Federated Wiki is relatively small. Contributing questions and sharing with others is managed through [an issues list on Github](#) ↗.

Git and Github: Work Organization

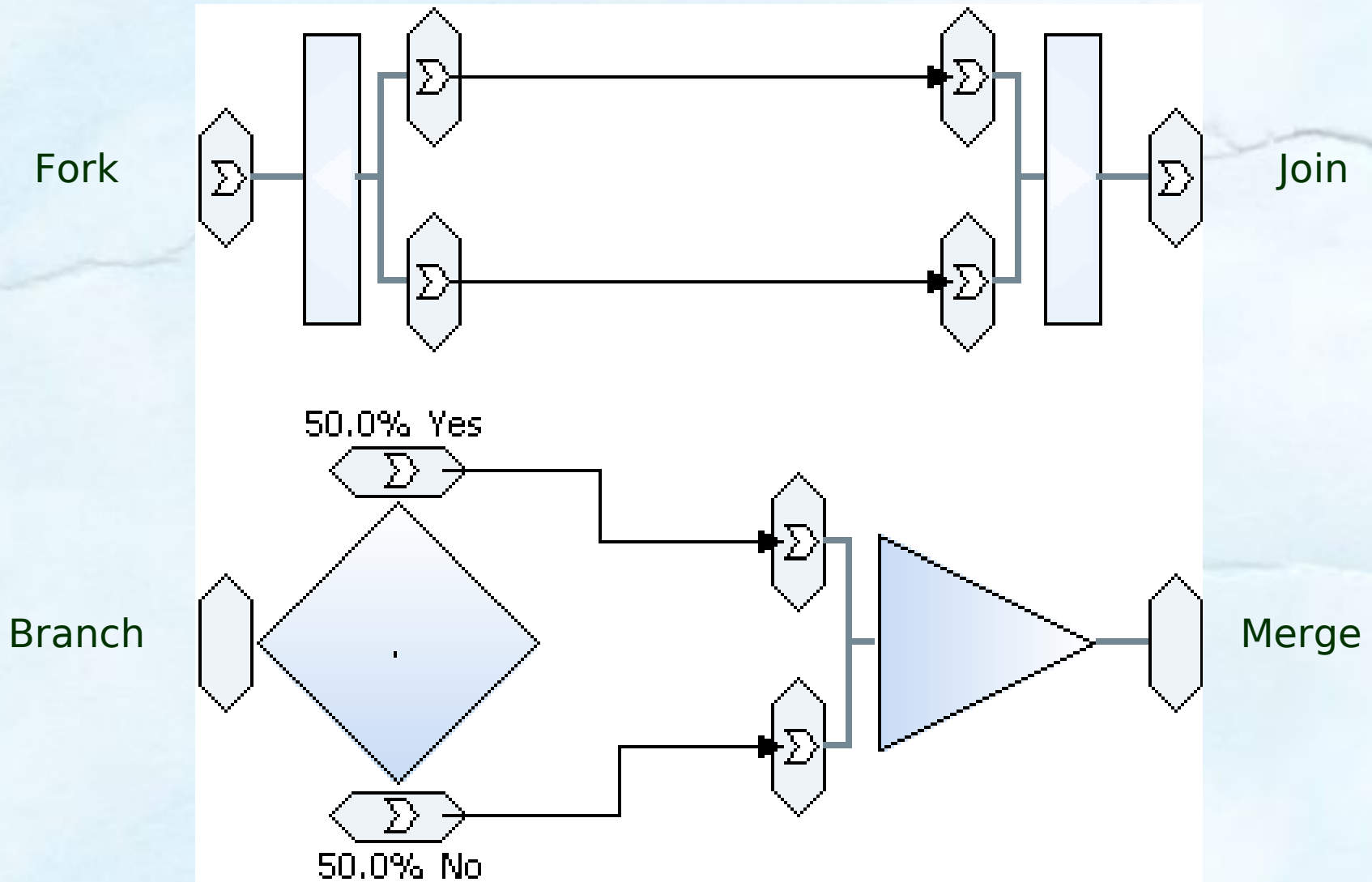


Git is architected as decentralized, with an origin from where individuals may push to and pull from (as well as amongst each other).

This organization of work enables individuals to first work independently, and then subsequently discuss merging their changes together.

Source: Vincent Driessen, "A successful Git branching model" January 05, 2010 at <http://nvie.com/posts/a-successful-git-branching-model/>

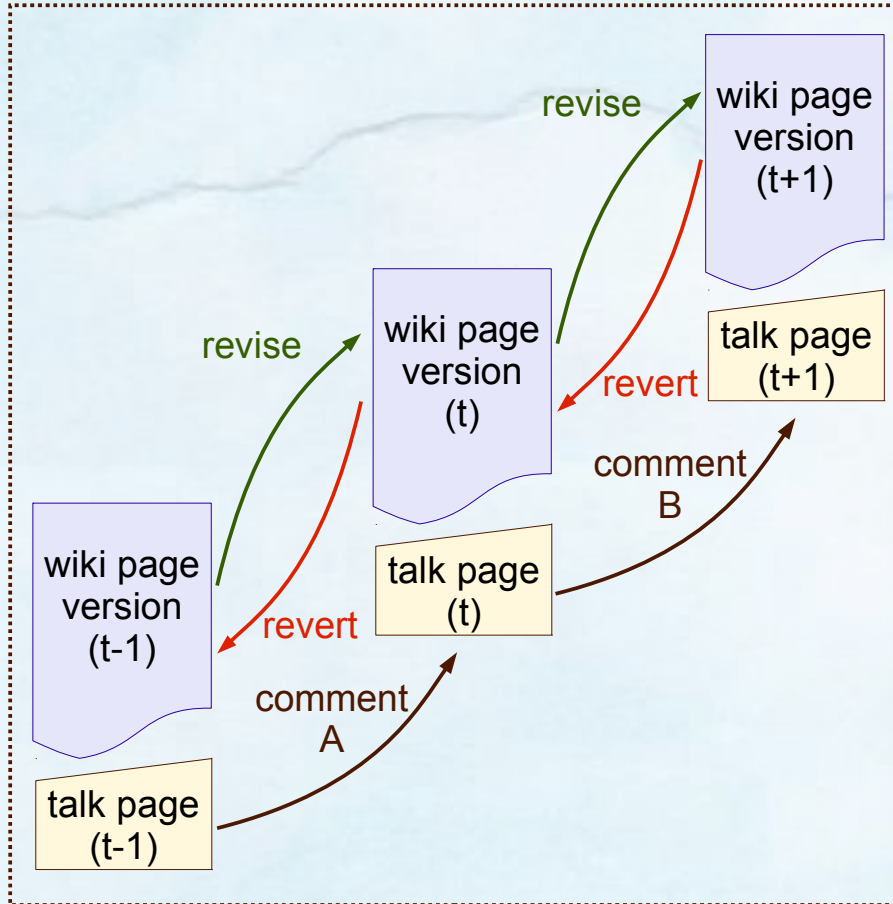
Collaboration evolution as Fork-Join, and Branch-Merge



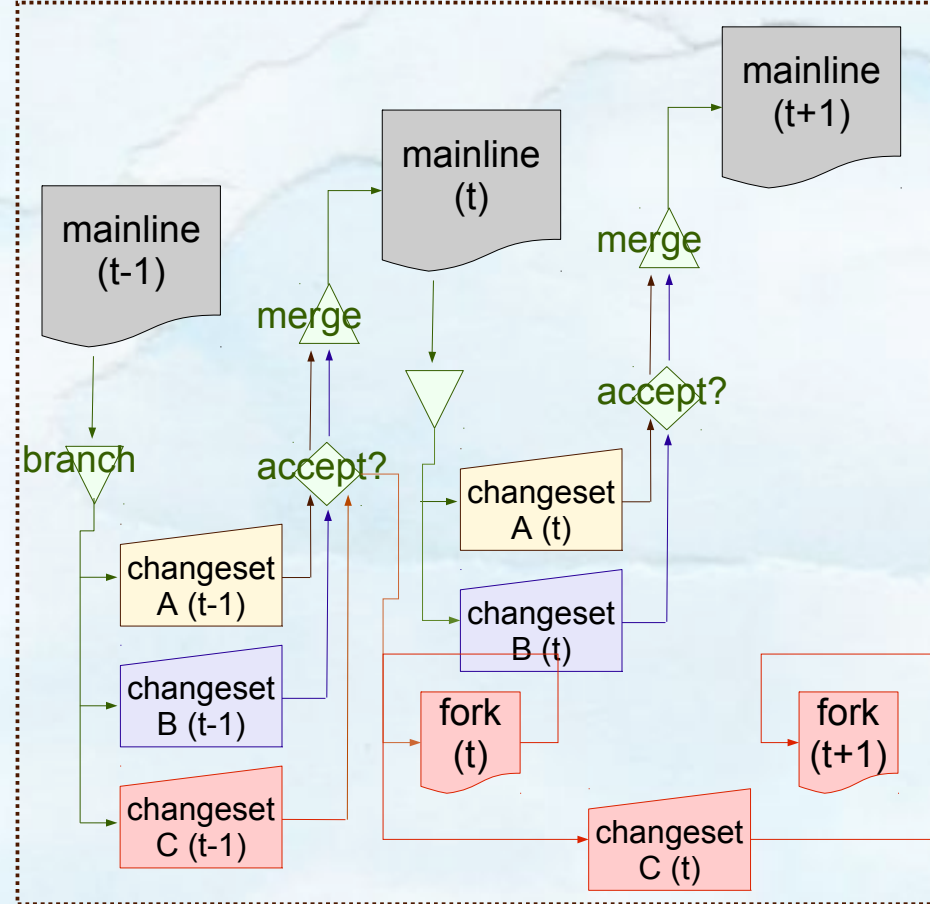
Source: David Ing, *Open source with private source: coevolving architectures, styles and subworlds in business* (forthcoming)

Inductive-consensual Wiki revise-revert cycles become Federated Wiki perspectives, branch-merge or fork

Wiki as Inductive-Consensual



(Federated) Wiki as Multiple Perspectives



Source: Mitroff, Ian I., and Richard O. Mason. 1982. "Business Policy and Metaphysics: Some Philosophical Considerations." *The Academy of Management Review* 7 (3) (July 1): 361–371. doi:10.2307/257328. <http://www.jstor.org/stable/257328>.



 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](#) 

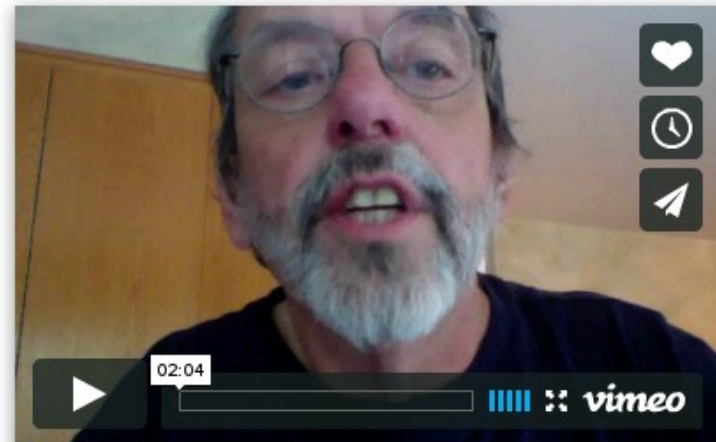
Learn More

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](#).

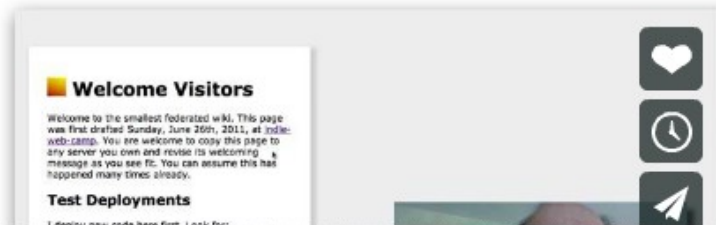


Smallest Federated Wiki

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.



OK

The terms of contributions to a multi-organization, multi-person project should be clear at the outset

Copyright Assignment and Ownership

There are three ways to handle copyright ownership for free code and documentation that were contributed to by many people.

The first is to ignore the issue of copyright entirely (I don't recommend this).

The second is to collect a **contributor license agreement (CLA)** from each person who works on the project, explicitly granting the project the right to use that person's contributions. [...]

The third way is to get **actual copyright assignments from contributors**, so that the project (i.e., some legal entity, usually a **nonprofit**) is the **copyright owner** for everything. This is the most legally airtight way, but it's also the most burdensome for contributors; only a few projects insist on it.

Note that even under centralized copyright ownership, the code remains free, because open source licenses do not give the copyright holder the right to retroactively propertize all copies of the code....

Source: Karl Fogel, Producing Open Source Software,
<http://producingoss.com/en/copyright-assignment.html>

The Open Source Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. [...]

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license must explicitly permit distribution of software built from modified source code.

5. No Discrimination Against Persons or Groups

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

8. License Must Not Be Specific to a Product

9. License Must Not Restrict Other Software

10. License Must be Technology-Neutral

Source: Open Source Initiative, "The Open Source Definition",
<http://opensource.org/osd>

Attribution of the source may be sufficient recognition

Choosing a License

Although there are many open source licenses [1], the important ones can be divided into two categories, and within each category only a few licenses are in widespread use.

The “Anything Goes” Licenses

These place very few restrictions on what can be done with the code, including using the code in proprietary derivative works. They require only attribution in a specified manner. The most widely-used licences of this type are:

- BSD-style
- MIT/X11-style
- Apache Software License, version 2

The Copyleft (so-called “viral”) Licenses

These also allow open distribution, modification, and re-use of the code (with attribution), but insist that any derivative works be distributed under the same terms. Thus proprietary derivatives by third parties are not possible (unless the copyright holder gives permission). Note, however, that commercial use and derivation by anyone is permitted, as long as the terms of the license are honored. Widely-used licenses of this type are:

- GPLv3 (GNU General Public License, version 3)
- AGPLv3 (Affero GPL, version 3)

Source: Civic Commons,
http://wiki.civiccommons.org/Choosing_a_License

The MIT License (MIT)

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Apache License, Version 2.0

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions. [...]

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work,....

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions: [...]

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. [...]

The Apache Foundation incubation process might be adapted for our needs, or compared to alternatives

Formulating a Proposal

- Preparation
- Project Name
- Presentation
- Developing The Proposal
- The Vote

Proposal Template

- Abstract
- Proposal
- Background
- Rationale
- Initial Goals
- Current Status
 - Meritocracy
 - Community
 - Core Developers
 - Alignment
- Known Risks
 - Orphaned products
 - Inexperience with Open Source
 - Homogenous Developers
 - Reliance on Salaried Developers
 - Relationships with Other Apache Products
 - A Excessive Fascination with the Apache Brand
- Documentation
- Initial Source
- Source and Intellectual Property Submission Plan
- External Dependencies
- Cryptography
- Required Resources
 - Mailing lists
 - Subversion Directory
 - Issue Tracking
 - Other Resources
- Initial Committers
- Affiliations
- Sponsors
 - Champion
 - Nominated Mentors
 - Sponsoring Entity

Source: "A Guide To Proposal Creation", Apache Software Foundation, <http://incubator.apache.org/guides/proposal.html>
<http://incubator.apache.org/guides/proposal.html><http://incubator.apache.org/guides/proposal.html>

Conversations for Possibilities on Service Systems Thinking

With Service Systems Thinking at its inception, community participants can discuss potential futures for their collaborations.

For [Multiple Perspectives Open Collaboration](#),

1. We could have [Federated Authored Content](#) on open source platforms.

For [Generative Pattern Language](#),

2. We could be reoriented for [Unfolding Wholeness](#), [Layering Systems of Centers](#) and/[with Creating Interactive Value](#).

For [Service Science, Management, Engineering and Design](#),

3. We could have [Transdisciplinary Cooperation on Service Systems Improvement](#).

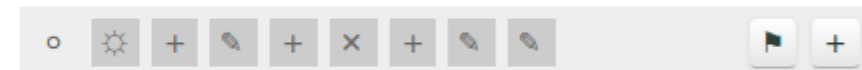
For [Systems Thinking](#),

4. We could have service systems [Evolving from the Systems Thinking Tradition](#).

Unfolding Wholeness

In Christopher Alexander's post-*Nature of Order* work, the ideas of [Generative Code](#), [Wholeness](#) and [Unfolding](#) are strongly coupled.

Some reconciliation between designing goal-oriented and platform-oriented approaches can be understood through the distinction between [Teleological Systems Development](#) and [Ateleological Systems Development](#). While the pattern language work of Christopher Alexander has largely been seen as ateleological, the philosophy behind the design of service systems is not always clear.



CC BY-SA 3.0 • JSON • fed.coevolving.com

Generative Code

The context for generative code, in the work of Christopher Alexander, is in built environments, and directly related to [Morphogenesis](#).

A generative code is a system of explicit steps, for creating [a social-spatial] fabric. It defines the end product, not by specifying the end-product itself, but by defining the steps that must be used to reach the end product. Unlike a process which defines the end product, and then leaves the getting there to the developer, the processes initiated by a generative code assure that the end product will be unique each time it occurs, and will be unique in just the ways that matter.¹

¹This approach to building is based on morphogenesis, which has been the basis of Alexander's work throughout his career as architect, planner, educator, theorist and builder. The theory, connections to other fields of science, and hundreds of examples of putting this theory into action are covered in the four books of *The Nature of Order*, Alexander's recently completed four-volume work.

Morphogenesis

Morphology is "the study of the form of things", and *morphogenesis* (in biology) is "the origin and development of morphological characteristics, according to Oxford Dictionaries [↗](#)".

Christopher Alexander speaks to morphogenesis in the context of built environments, drawing inspiration from biology.

Things in the biological world, almost by definition, are created continuously by morphogenesis, that is by a process which is all the time growing and adapting, whether it be in a growing embryo or in a forest or a field, and which gives form, progressively, while growth and change and adaptation are happening. In real morphogenesis the form of what is coming, or what is about to be, is always drawn from the form of what was in the moment just before. That is, things are always going like that. If a tree is growing for 500 years, it is continuously unfolding from its previous state, and then what we see and recognize is first of all in itself a process. But even if you just look at it in its static state, it is at that moment the end product of transformations that have

Wholeness

The "quality without a name" described in *The Timeless Way of Building* (1979) came to be described by Christopher Alexander as "wholeness" in the *The Nature of Order* (2004):

The argument of Book 1, *The Phenomenon of Life*, may be captured by the following results that summarize 30 years of observation and experiment:

1. A previously unknown phenomenon that may be called "life" or "wholeness" has been observed in artifacts. This quality has been noticed in certain works of art, buildings, public space, parts of buildings, and in a wide range of other humanmade things.
2. The idea of how much life is in things is objective in the sense of observation and is thus common to people of different inclinations and cultures. This is a surprise, since the finding seems to contradict the accepted wisdom of cultural relativity. (demonstrated)
3. This quality of life seems to be correlated with the repeated appearance of 15 geometric properties—or geometrical invariants—that appear

Unfolding

While patterns have generally been static, unfoldings are intended as dynamic.

Unfoldings slightly resemble patterns, in that they are nuggets of information, which help you shape some part or aspect of the environment. However, unfoldings are vastly different from patterns in the way they work, and it is these differences which give them their power and effectiveness. Each unfolding has three key features which define its operation and its effect.

1. Unlike a pattern, which is a static configuration, an unfolding is dynamic. It acts to generate form.
2. Unlike a pattern, an unfolding arises from the whole, is shaped by the whole, and acts upon the whole.
3. An unfolding is by its nature personal, and requires human input and human feeling from the people doing the work, as an essential part of its contribution to the formation of the environment.

See "[What is an unfolding](https://livingneighborhoods.org)" on livingneighborhoods.org 📄

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^[1]

**Pattern
Name:**

(Use italics for pattern names per Meszaros).

Aliases:

(Aliases, or none)

Problem

Give a statement of the problem that this pattern resolves. The problem may be stated as a question.

Context

Describe the context of the problem.

Forces

Describe the forces influencing the problem and solution. This can be represented as a list for clarity.

- Force one
- Force two

Solution

Give a statement of the solution to the problem.

Resulting Context

Describe the context of the solution.

Rationale

Explain the rationale behind the solution.

Known Uses

List or describe places where the pattern is used.

**Related
Patterns**

List or describe any related patterns.

Source: [1] "Patterns", The Hillside Group, <http://hillside.net/patterns> ; [2] "Writing Patterns", AG's HTML template at <http://hillside.net/index.php/ag-template> ; "Canonical Form" (for writing patterns) at <http://c2.com/cgi/wiki?CanonicalForm>

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 up 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>

Writing Software Patterns

I've spent a lot of my writing energy writing patterns. From time to time I ask myself questions about why I do that and what makes a good pattern. This is how I look at patterns with my suggestions for people who are interested in patterns themselves.

01 August 2006



Martin Fowler

Find similar writing

Gang-of-Four

(Gamma, Helm, Johnson, Vlissides 1994, Design Patterns)

- Intent
- Motivation
- Applicability
- Structure
- Participants
- Collaborations
- Consequences
- Implementation
- Sample Code
- Known Uses
- Related Patterns

Contents

What is a Pattern
 Patterns versus Recipes
 Why are Patterns important?
 Important Parts of Patterns
 Patterns are Solutions
 An Evocative Name
 Why as well as how
 Code Examples
 Common Pattern Forms
 Alexandrian Form
 GOF Form
 Portland Form
 Coplien Form
 POSA Form
 P of EAA Form
 Choosing Your Pattern Form
 Common Issues
 Arranging Patterns into a Structure
 Patterns and Pattern Languages
 Granularity of Patterns
 Tasks rather than Tools
 Nothing new here

Christopher Alexander

- (1) Picture with archetypal example
- (2) Paragraph sets context with how it helps to complete larger patterns
- (3) Three diamonds (start of problem)
- (4) Headline essence of problem (bold type)
- (5) Body of problem, empirical background
- (6) Solution instructions (bold type) describing field of physical and social relations
- (7) Diagram
- (8) Three diamonds (main body finished)
- (9) Paragraph that ties pattern to smaller patterns

Portland

(C2 wiki, short)

- Problem
- ... therefore ...
- Solution

Pattern-Oriented Software Architecture

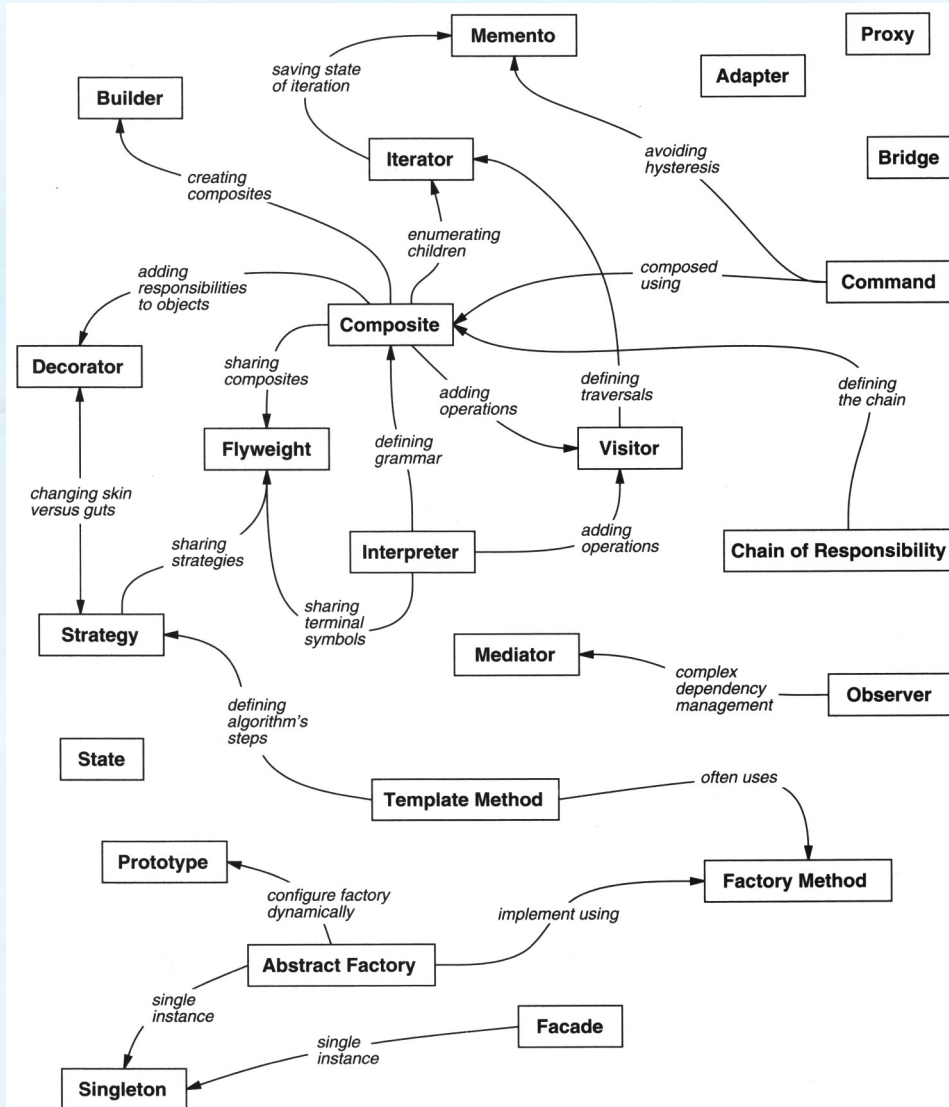
- Summary
- Example
- Context
- Problem
- Solution
- Structure
- Dynamics
- Implementation
- Example resolved
- Variants
- Known uses
- Consequences
- See also

Patterns of Enterprise Application Architecture

- How it works
- When to use it
- Examples

July 2014

Design Patterns (Catalog)



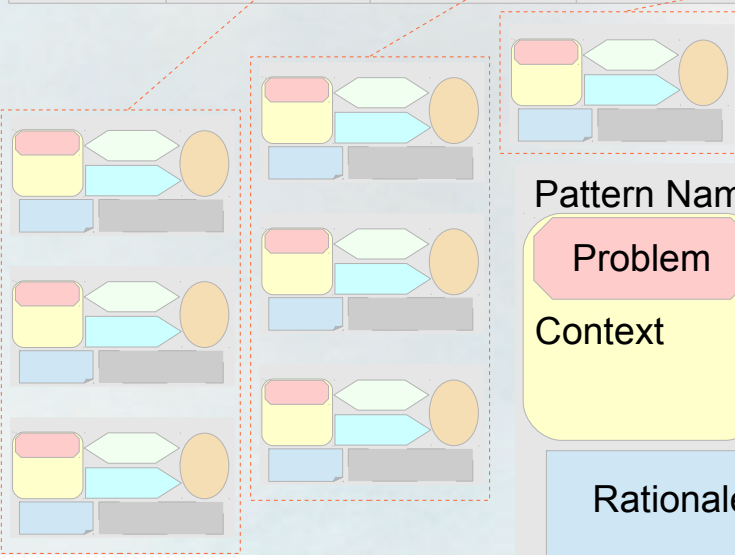
Source: Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. <http://books.google.ca/books?id=6oHuKQe3TjQC>.

Purpose	Design Pattern	Aspects That Can Vary
Creational	Abstract Factory	families of product objects
	Builder	how a composite object gets created
	Factory Method	subclass of object that is instantiated
	Prototype	class of object that is instantiated
	Singleton	the sole instance of a class
Structural	Adapter	interface to an object
	Bridge	implementation of an object
	Composite	structure and composition of an object
	Decorator	responsibilities of an object without subclassing
	Facade	interface to a subsystem
	Flyweight	storage cost of objects
	Proxy	how an object is accessed; its location
	Chain of Responsibility	object that can fulfill a request
	Command	when and how a request is fulfilled
Behavioral	Interpreter	grammar and interpretation of a language
	Iterator	how an aggregate's elements are accessed, traversed
	Mediator	how and which objects interact with each other
	Memento	what private information is stored outside an object, and when
	Observer	number of objects that depend on another object; how the dependent objects stay up to date
	State	states of an object
	Strategy	an algorithm
	Template Method	steps of an algorithm
	Visitor	operations that can be applied to object(s) without changing their class(es)

To appreciate service systems, can we aspire beyond a (Design) Pattern Catalog to a Generative Pattern Language?

(Design) Pattern Catalog

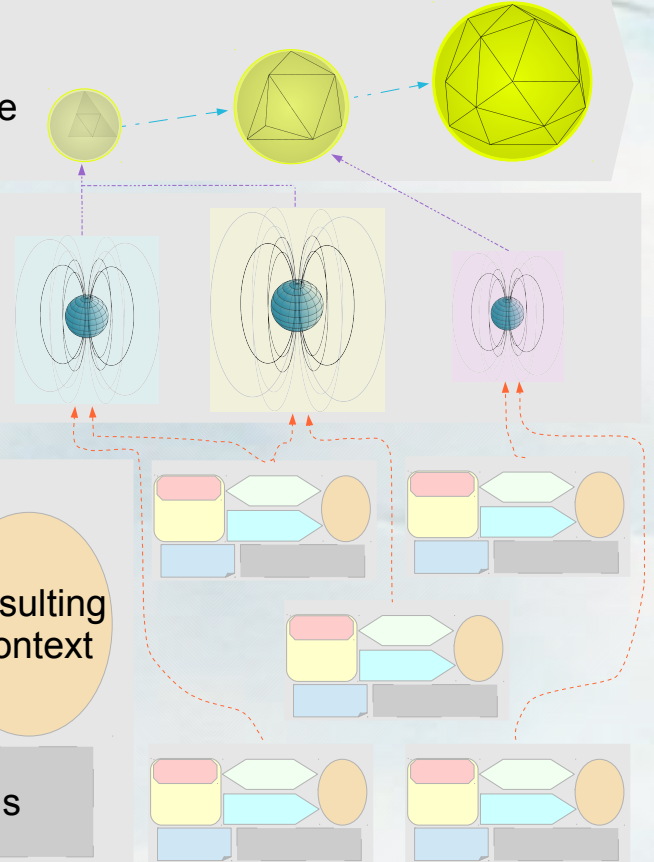
SCOPE	PURPOSE		
	Creational	Structural	Behavioral
Class	Factory Method	Adapter	Interpreter Template Method
Object	Abstract Factory Builder Prototype Singleton	Factory Method Bridge Composite ...	Chain of Responsibility Command Iterator ...



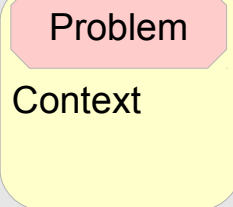
Generative Pattern Language

Unfolding
wholeness
(+ interactive
value?)

Centres
and
spaces,
in layers
and paces



Pattern Name



Forces

Solution

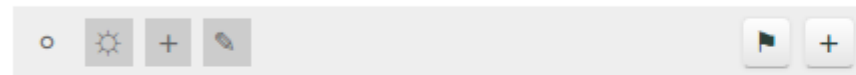
Resulting
Context

Rationale

Related Patterns

Transdisciplinary Cooperation on Service Systems Improvement


While we are having conversations for possibilities, initial engagement has been to [Seek Concurrence Across Professional Communities](#)





CC BY-SA 3.0 • JSON • fed.coevolving.com


Seek Concurrence Across Professional Communities


Preliminary interest in Service Systems Thinking has been expressed by leaders of associated organization, in conference presentations.

[INCOSE Systems Sciences Working Group](#) 
(International Council on Systems Engineering)

[ISSIP](#)  (International Society of Service Innovation Professionals)

- [2nd International Conference on The Human Side of Service Engineering, July 2014](#) , Krakow, Poland

[ISSS](#)  (International Society for the Systems Sciences)

- [58th Meeting of the International Society for the Systems Sciences, July 2014](#) , Washington, DC

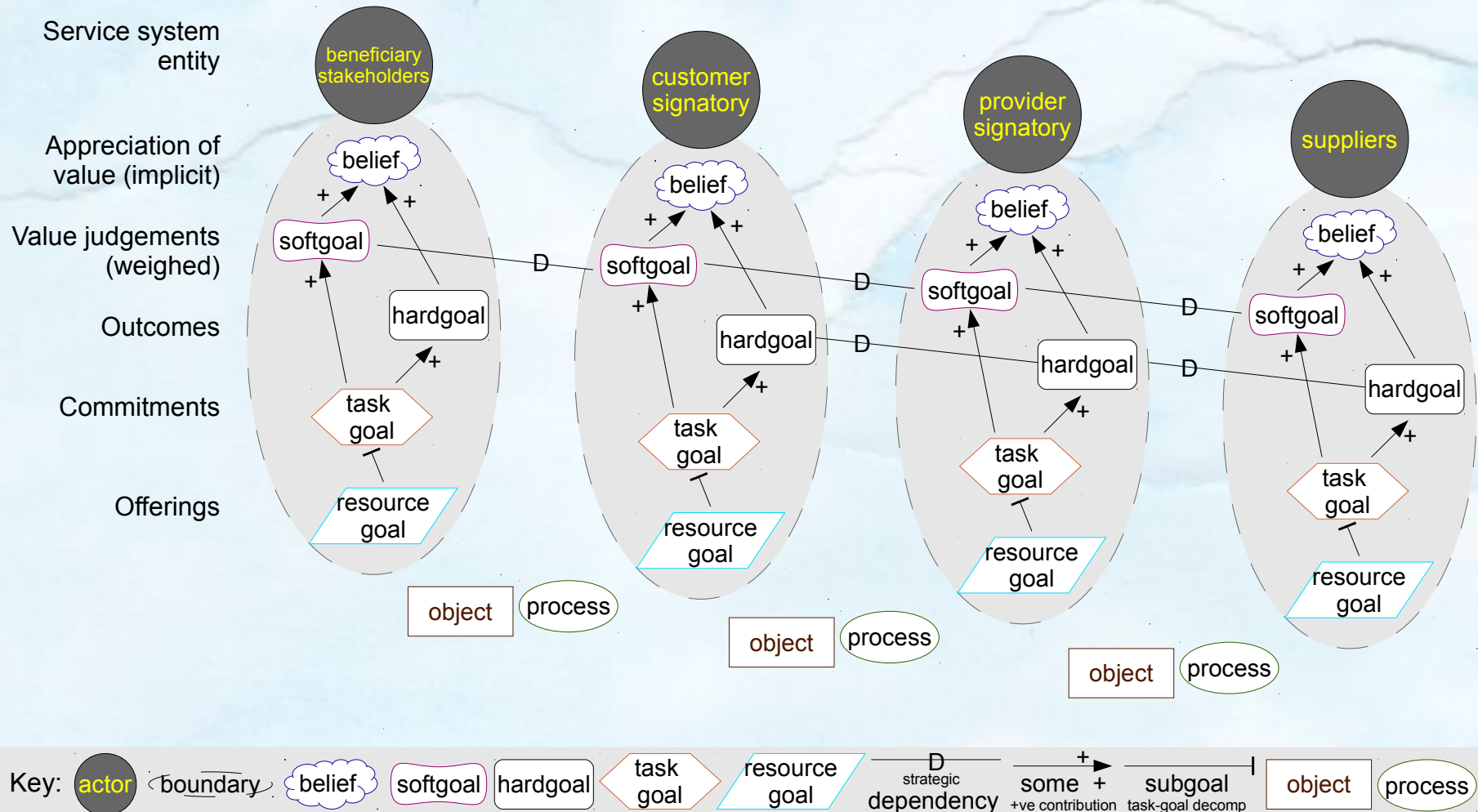
[The Hillside Group](#) 

- [21st Conference on Pattern Languages of Programs, September 2014](#) , Allerton Park, Monticello, IL

[Systemic Design Research Network](#) 

- [Relating Systems Thinking and Design 3 Symposium, October 2014](#) , Oslo, Norway

Could we model value constellation ontology synthesizing iStar and OPM representations?



Welcome to the i* Tools Site

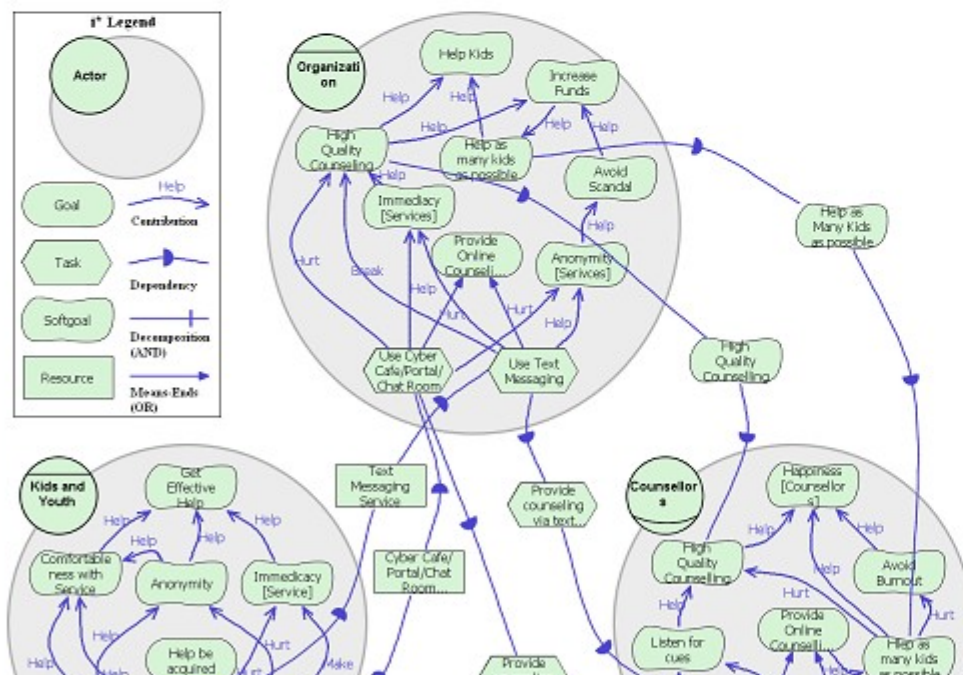
As part of the efforts of various current and former researchers at the [University of Toronto](http://www.cs.utoronto.ca/km/tools/), we have developed tools to support goal modeling: [OpenOME](#) and [OME](#).

Both the [OpenOME](#) and [OME](#) tools allow users to draw and analyze goal- and agent-oriented requirements models (i* models) as part of a model-based Software Engineering Methodology.

[OpenOME](#) is an open-sourced, Eclipse-based i* modeling tool. Development on the tool is ongoing. See the [OpenOME](#) site for more information, including [download](#) information. For development information see our [Trac Wiki](#).

[OME](#) is a close-sourced, java-based goal modeling tool. Development on this tool ceased in 2004. The tool is available free for [download](#), but requires u to sign a licence. See the [OME](#) site for more information.

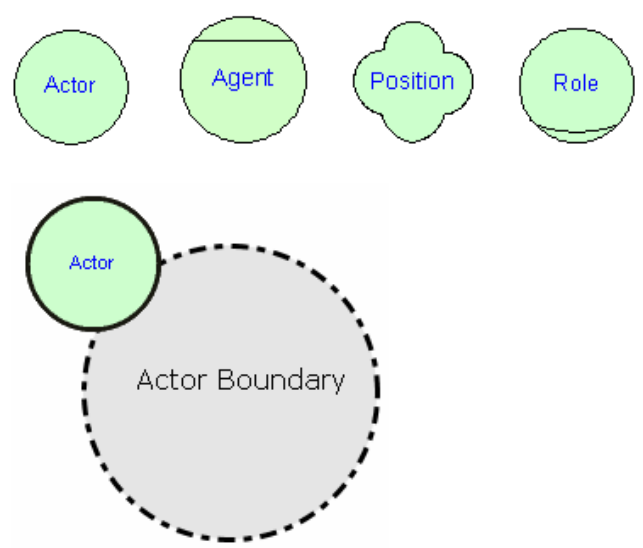
Example i* Model Created in OpenOME



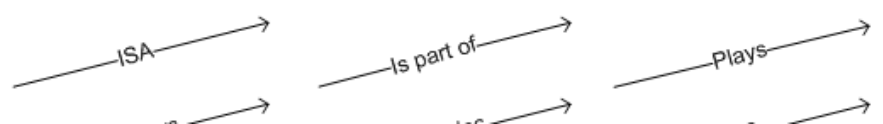
2. Basic i* Notation

This section provides only the graphical notation of i* syntax. An explanation of each notation can be found in the i* Glossary Section. Note that as i* models can be created by a variety of software tools, there can be small variations in notation appearance, mainly pertaining to color and line size.

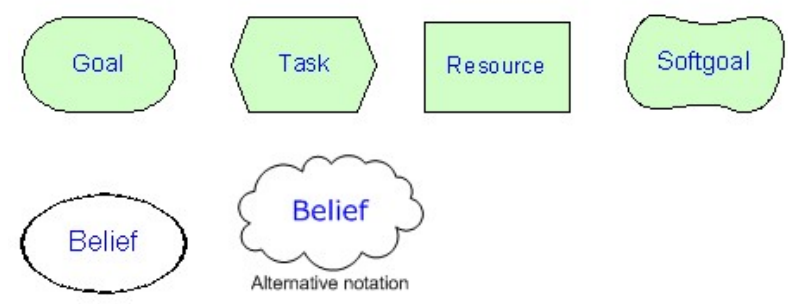
2.1 Actors



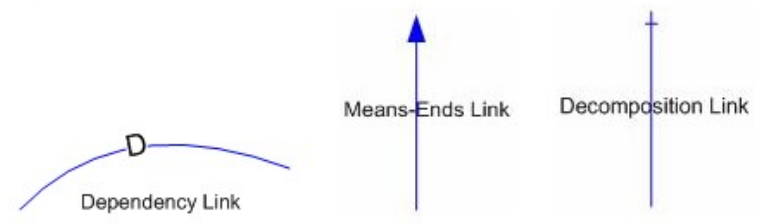
2.2 Actor Associations



2.3 Elements

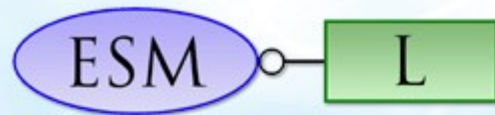


2.4 Links



2.5 Contribution Links





ENTERPRISE SYSTEMS MODELING LABORATORY

Home > About > Courses > Projects > Publications > OPM > Conferences > People > Gallery

OPM

system design and
management

The Maturation of Model-Based Systems Engineering: OPM as the ISO Conceptual Modeling Language Standard



Dov Dori

Massachusetts Institute of Technology
Technion, Israel Institute of Technology

SDM Webinar

June 2, 2014

Leadership, Innovation, Systems Thinking

ESML Events

August 2014

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Links

[MIT SDM Webinar presentation, 2014](#)

[MIT SDM Webinar recording, 2014](#)

[OPCAT Download](#)



[Faculty of Industrial Engineering and
Management](#)

[Lab Head - Prof. Dov Dori](#)

[OPCAT - OPM's support software](#)

Object-Process Methodology (OPM)

Things: **Objects** and **Processes**



Object

A thing that exists or might exist physically or informatically

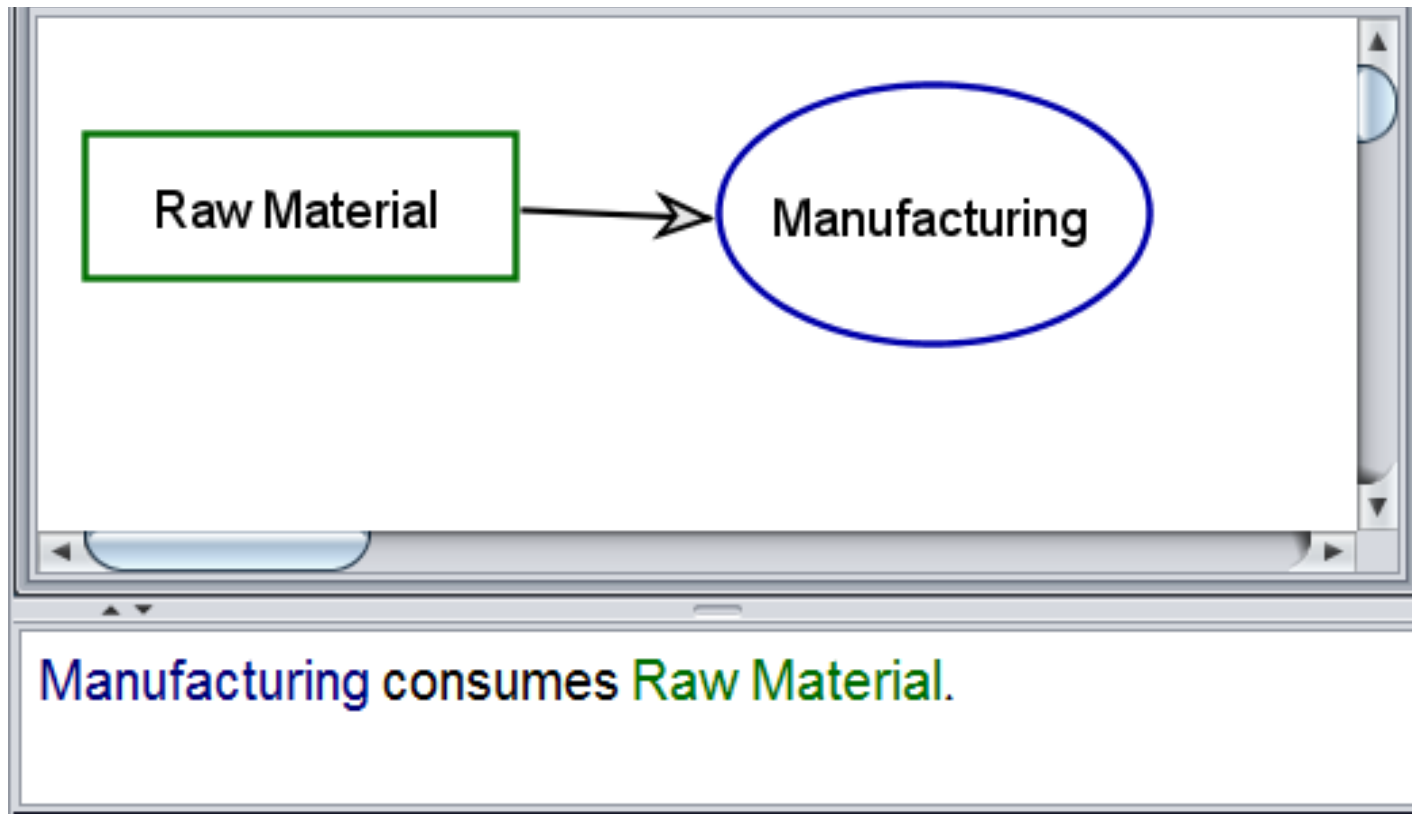


Process

A thing that transforms one or more objects

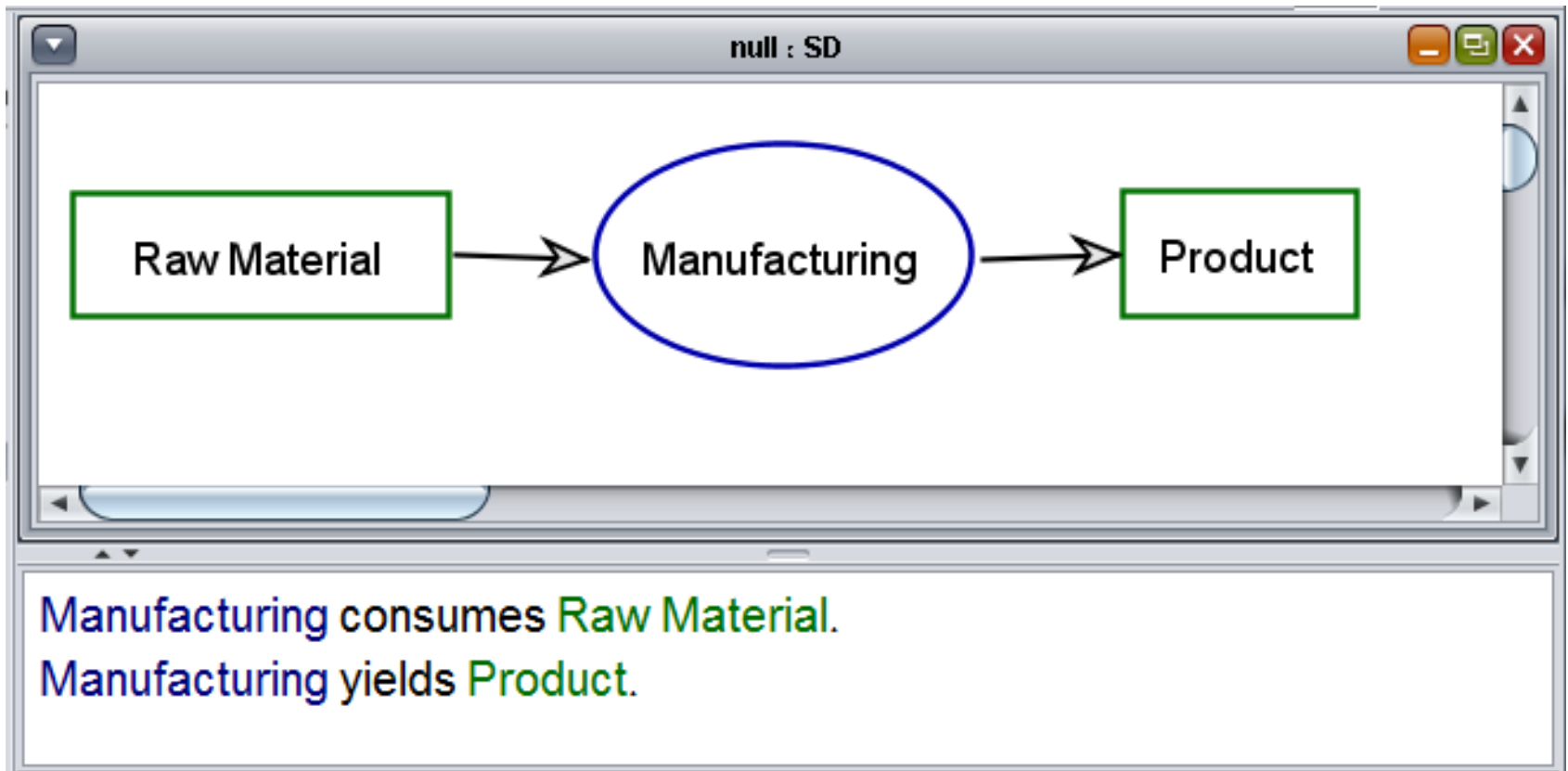
Processes transform objects by

- (1) Consuming them:



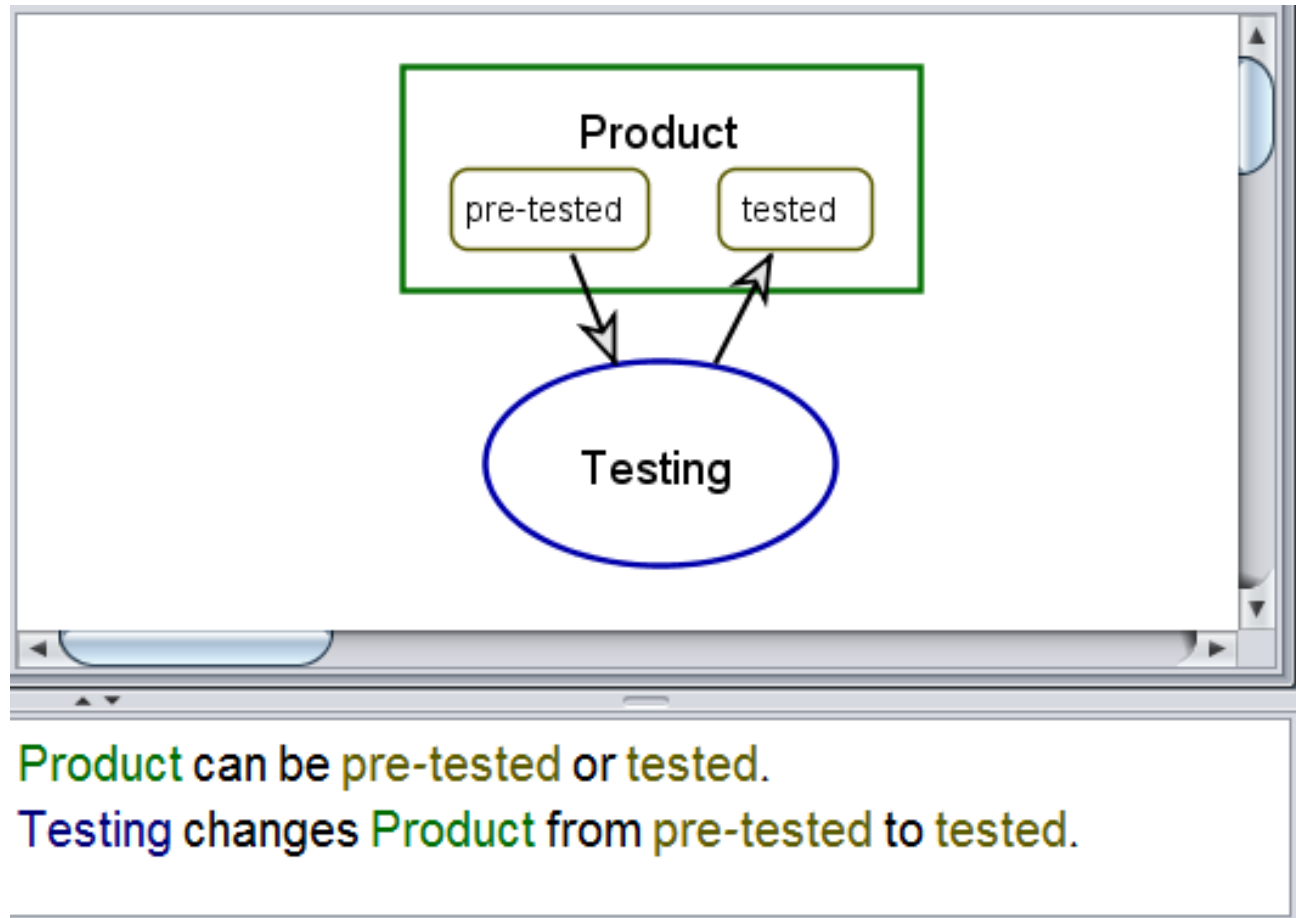
Processes transform objects by

- (2) Creating them:



Processes transform objects by

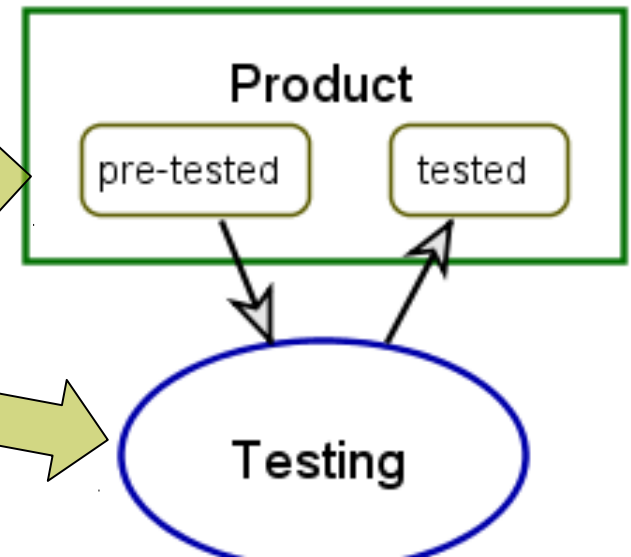
- (3) Changing their state:



So the OPM Things are:

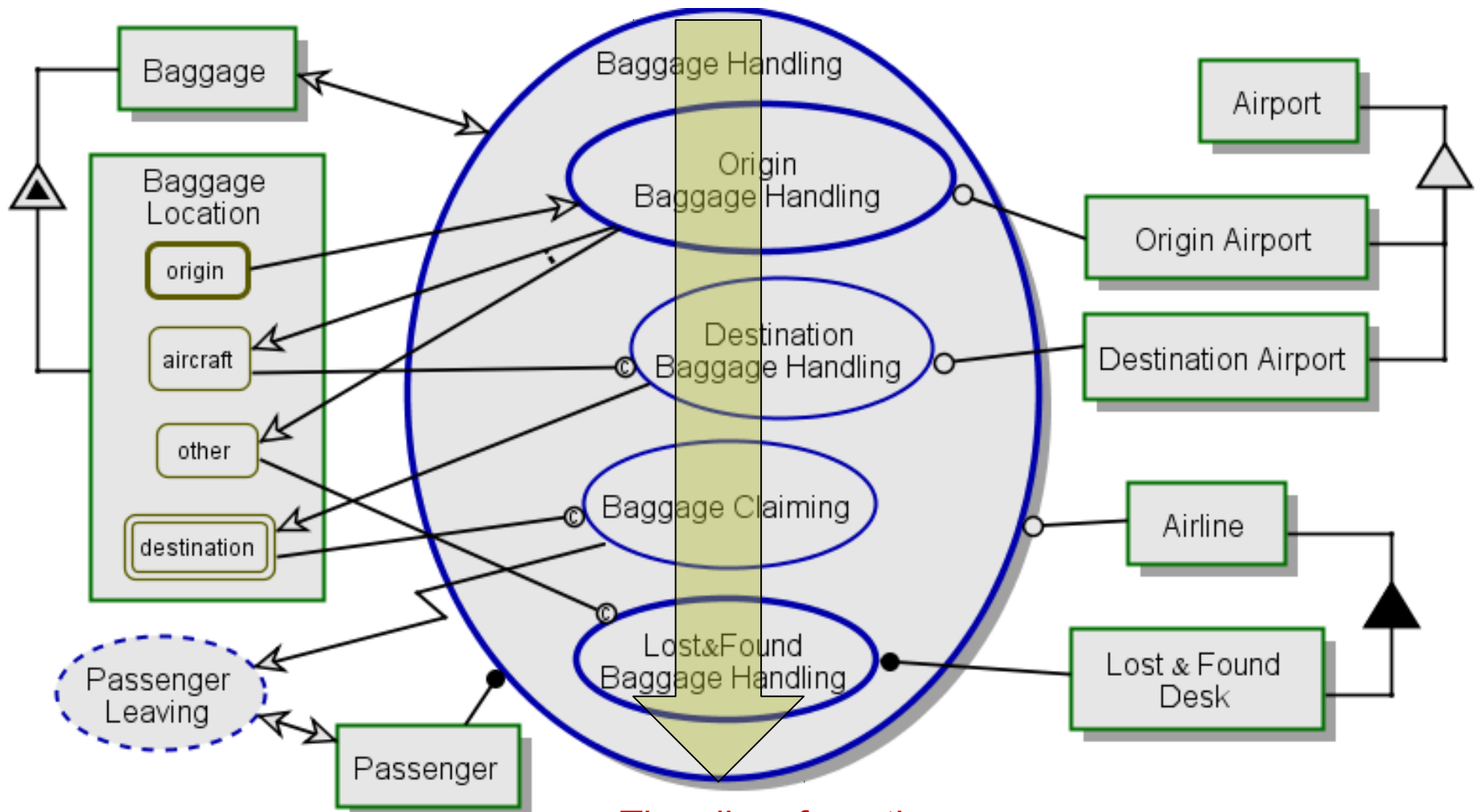
- 1. Object

- 2. Process



- All the rest are relations between things!

Zooming into Baggage Handling



Time line: from the process
ellipse top to its bottom

Conceptual Model-Based Systems Biology: Mapping Knowledge and Discovering Gaps in the mRNA Transcription Cycle

Judith Somekh , Mordechai Choder, Dov Dori

Article

About the Authors

Metrics

Comments

Related Content

Download

Print

Show Figures

Abstract

Introduction

Results

Summary and Discussion

Materials and Methods

Supporting Information

Author Contributions

References

Reader Comments (0)

Abstract

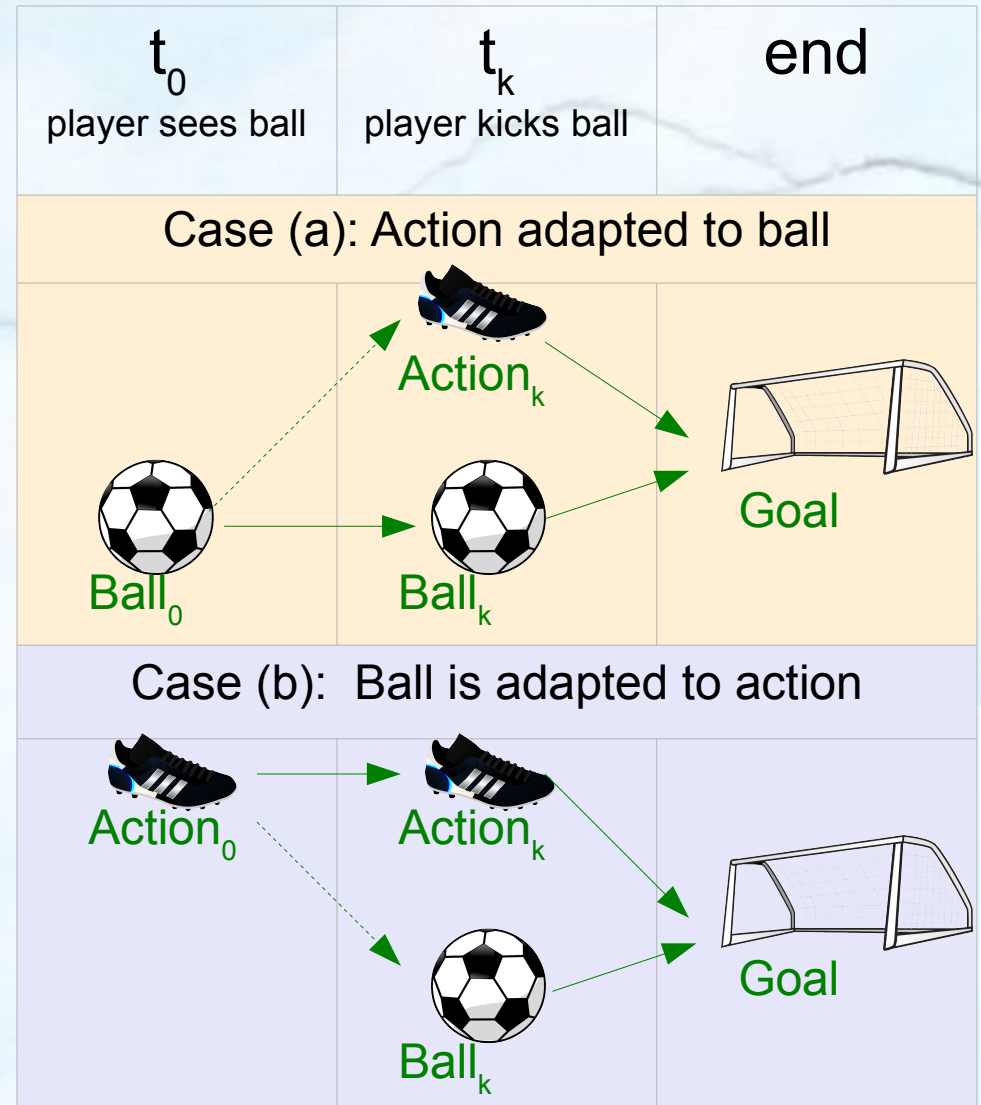
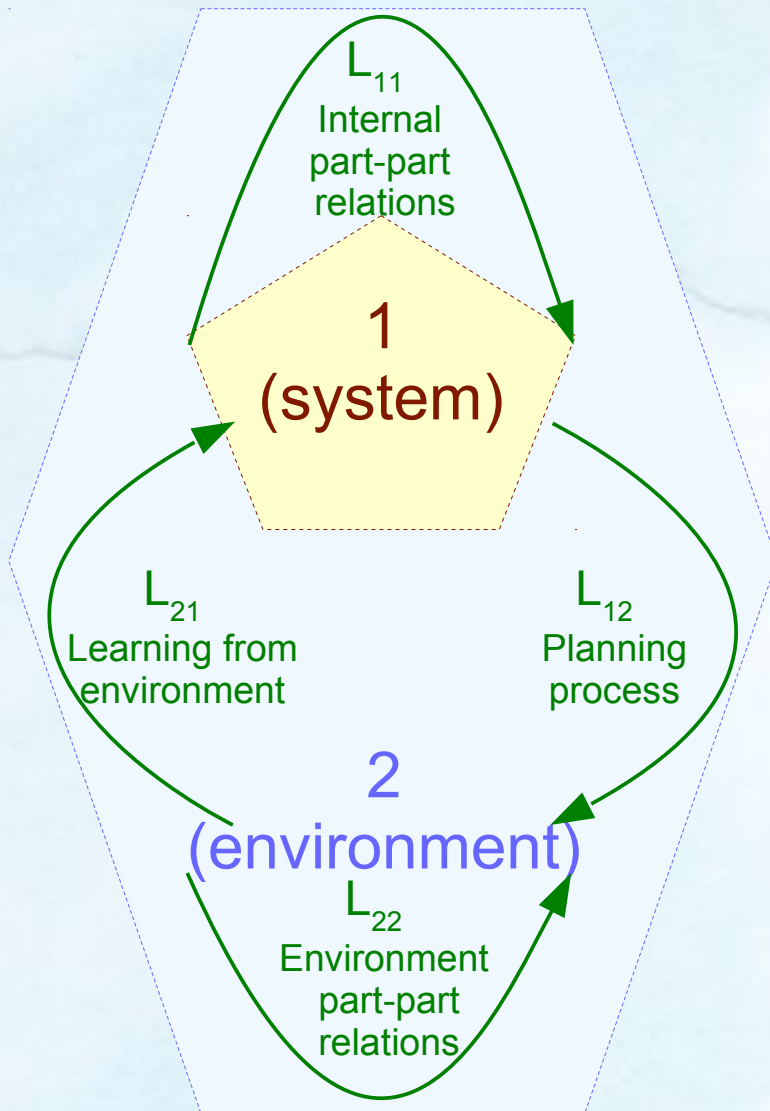
We propose a Conceptual Model-based Systems Biology framework for qualitative modeling, executing, and eliciting knowledge gaps in molecular biology systems. The framework is an adaptation of Object-Process Methodology (OPM), a graphical and textual executable modeling language. OPM enables concurrent representation of the system's structure—the objects that comprise the system, and behavior—how processes transform objects over time. Applying a top-down approach of recursively zooming into processes, we model a case in point—the mRNA transcription cycle. Starting with this high level cell function, we model increasingly detailed processes along with participating objects. Our modeling approach is capable of modeling molecular processes such as complex formation, localization and trafficking, molecular binding, enzymatic stimulation, and environmental intervention. At the lowest level, similar to the Gene Ontology, all biological processes boil down to three basic molecular

ADVERTISING

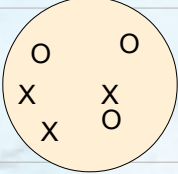
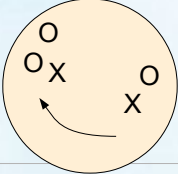
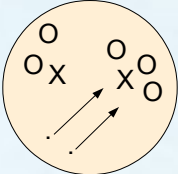
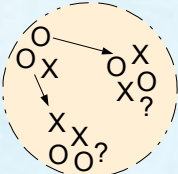


Rede
PLOS
Journ

Open systems (Emery and Trist), directive correlation (Sommerhoff)



The Causal Texture of Social Environments – Extended fields of directive correlations (Emery and Trist)

	Where O = goals (goodies), X = noxiants (baddes)		Elements to know	Ideals	Forms of learning	Forms of planning
Type 1. Random Placid		Goals and noxiants randomly distributed. Strategy is tactic. "Grab it if it's there". Largely theoretical of micro, design, e.g. concentration camps, conditioning experiments. Nature is not random.	system	Homonymy – sense of belonging	conditioning	tactics
Type 2. Clustered Placid		Goals and noxiants are lawfully distributed – meaningful learning. Simple strategy – maximize goals, e.g. use fire to produce new grass. Most of human span spent in this form. Hunting, gathering, small village. What people mean by the "good old days".	system, action	Nurturance – caring for	meaningful	tactics / strategies
Type 3. Disturbed Reactive		Type 2 with two or more systems of one kind <i>competing</i> for the same resources. Operational planning emerges to out-manoeuvre the competition. Requires extra knowledge of both Ss and E. E is stable so start with a set of givens and concentrate on problem solving for win-lose games. Need to create instruments that are variety-reducing (foolproof) – elements must be standardized and interchangeable. Birth of bureaucratic structures where people are redundant parts. Concentrate power at the top – strategy becomes a power game.	system, action, learning	Humanity – in broadest sense	problem solving	tactics / operational strategies
Type 4. Turbulent		Dynamic, not placid/stable. Planned change in type 3 triggers off unexpected social processes. Dynamism arises from the field itself, creating unpredictability and increasing <i>relevant uncertainty</i> and <i>its continuities</i> . Linear planning impossible, e.g. whaling disrupted reproduction, people react to being treated as parts of machine. Birth of open systems thinking, ecology, and catastrophe theory.	system, action, learning, environment	Beauty – includes fitting together naturally	puzzle- solving	active adaptive planning

Social Systems Fields as three perspectives: socio-psychological, socio-technical, socio-ecological

[... the] socio-psychological, the socio-technical and the socio-ecological perspectives ... emerged from each other in relation to changes taking place in the wider social environment. One could not have been forecast from the others. Though interdependent, each has its own focus. Many of the more complex projects require all three perspectives. [Trist & Murray 1997, p. 30]

Socio-psychological

... in Institute projects, the psychological forces are directed towards the social field, whereas in the Clinic, it is the other way around [with social forces directed toward the psychological field].

[Trist & Murray 1997, p. 31]

Social-technical

... the best match between the social and technical systems of an organization, since called the principle of joint optimization

... the second design principle, the redundancy of functions, as contrasted with the redundancy of parts.

[Trist & Murray 1997, p. 32]

Socio-ecological

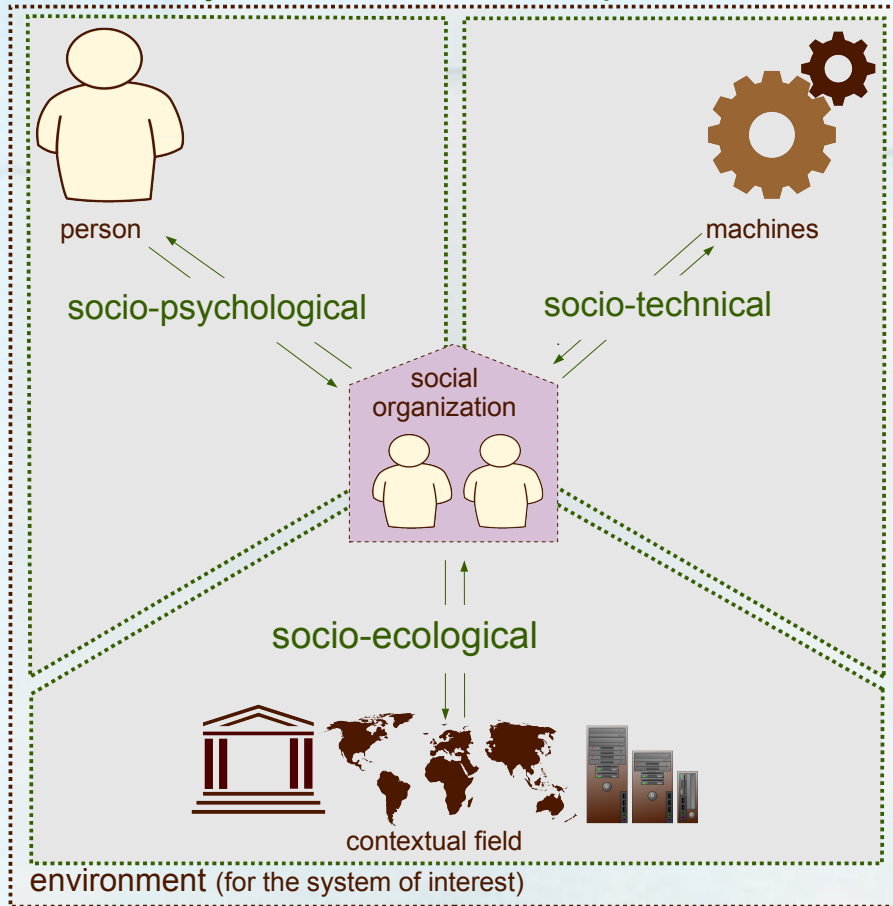
... the context of the increasing levels of interdependence, complexity and uncertainty that characterize societies at the present time.

... new problems related to emergent values such as cooperation and nurturance.

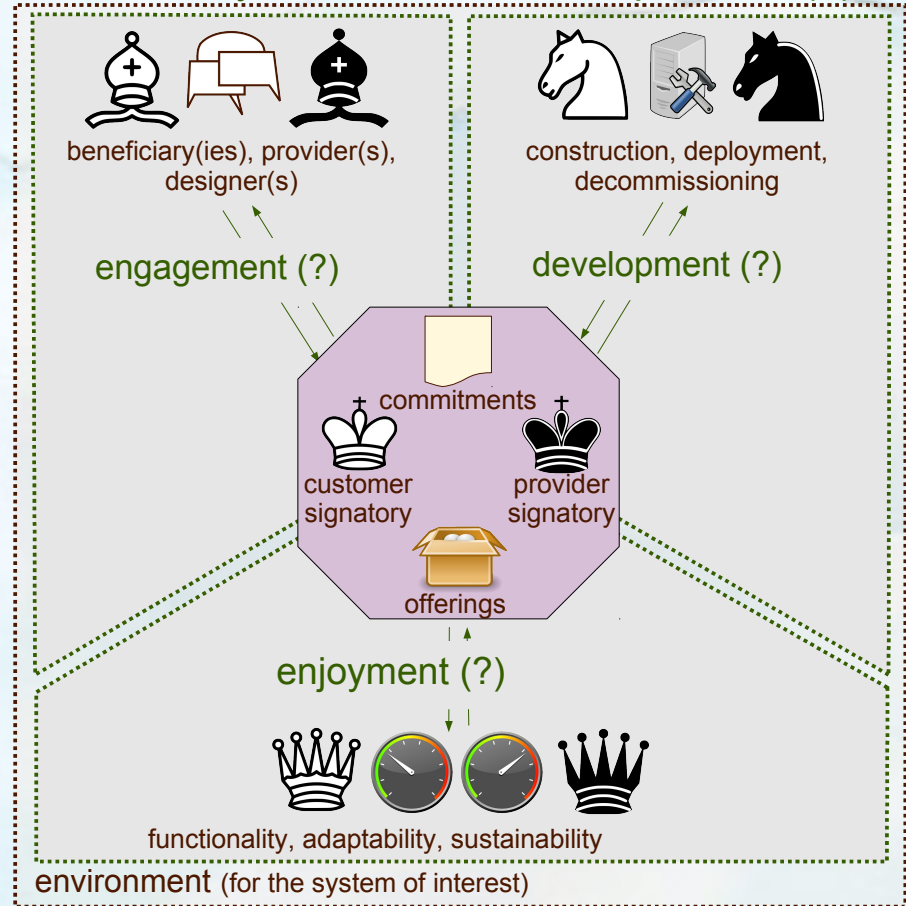
[Trist & Murray 1997, p. 33]

Can we build on Social Systems Science towards a new Service Systems Science?

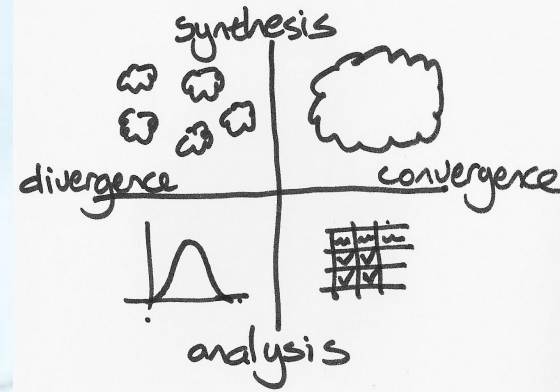
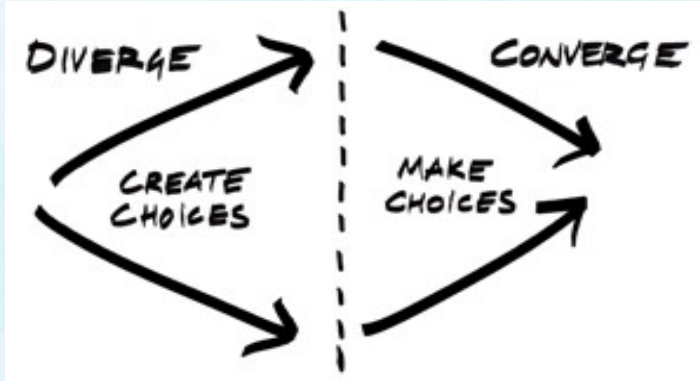
Social Systems Science Perspectives



Service Systems Science Perspectives (?)



Design Thinking: Divergent-Convergent, Synthesis-Analysis



Design thinking is different and therefore it feels different.

Firstly it is not only convergent. It is a series of divergent and convergent steps. **During divergence we are creating choices and during convergence we are making choices.**

For people who are looking to have a good sense of the answer, or at least a previous example of one, before they start divergence is frustrating. It almost feels like you are going backwards and getting further away from the answer but this is the essence of creativity. Divergence needs to feel optimistic, exploratory and experimental but it often feels foggy to people who are more used to operating on a plan. Divergence has to be supported by the culture.

The second difference is that design thinking relies on **an interplay between analysis and synthesis, breaking problems apart and putting ideas together.** Synthesis is hard because we are trying to put things together which are often in tension.

Less expensive, higher quality for instance. [...]

Designers have evolved visual ways to synthesize ideas and this is another one of the obstacles for those new to design thinking; a discomfort with visual thinking. A sketch of a new product is a piece of synthesis. So is a scenario that tells a story about an experience. A framework is a tool for synthesis and design thinkers create visual frameworks that in themselves describe spaces for further creative thinking.

Source: Tim Brown "What does design thinking feel like?" *Design Thinking* (blog), Sept. 7, 2008 at <http://designthinking.ideo.com/?p=51> ; "Why Social Innovators Need Design Thinking", *Stanford Social Innovation Review*, Nov. 15, 2011 at http://www.ssireview.org/blog/entry/why_social_innovators_need_design_thinking .

Agenda

1. Service Systems Thinking, In Brief
2. Conversations for Orientation
3. Conversations for Possibilities
- 4. Conversations for Action
5. Conversations for Clarification

4.1 Seek concurrence across professional communities

4.2 Seek collaborating authors on service systems thinking

Seeking concurrence



- International Workshop, January 2014, Los Angeles
- International Symposium, June 2014, Las Vegas



ISSIP
INTERNATIONAL SOCIETY OF SERVICE INNOVATION PROFESSIONALS

- Human Side of Service Engineering, July 2014, Krakow



- 58th Annual Meeting, July 2014, Washington, DC



- Pattern Languages of Programming Conference, September 2014, Allerton, IL



- Relating Systems Thinking and Design Symposium, October 2014, Oslo

Agenda

1. Service Systems Thinking, In Brief
2. Conversations for Orientation
3. Conversations for Possibilities
4. Conversations for Action
- 5. Conversations for Clarification

5.1 Explore methods that encourage multiple perspectives inquiry

Strategic Assumption Surfacing and Testing

I. ASSUMPTION SPECIFICATION

Original Strategies → Data → Assumptions

II. DIALECTIC PHASE

Counter Strategies ← Data ← Assumption Negation

III. ASSUMPTION INTEGRATION PHASE

Strategy Pool → Data → Assumption Pool

IV. COMPOSITE STRATEGY CREATION

“Best” Strategy ← Data ← Acceptable Assumptions

By working backwards to underlying assumptions, the proposed process ... requires that each strategy contain in addition to supporting data a list of assumptions (i.e., given conditions, events, or attributes that are or must be taken as true) which implicitly underlie the strategy.

... each assumption previously identified is negated and reformulated as a counter-assumption that negates the spirit of the original statement. If the counter-assumption is implausible, it is dropped. Those counter assumptions which one can conceive of as being true or plausible in some circumstances are then examined individually and collectively to see if they can be used as a basis both for defining and deducing an entirely new strategy.

Instead of trying to resolve differences in strategies directly at the resultant level of strategy, the process concentrates on negotiating an acceptable set of assumptions that the decision makers are prepared to take as given conditions for the formulation of the problem.

... development operates on a more rational basis as defined in traditional problem solving and decision theory terms. The composite set of acceptable assumptions can be used as an explicit foundation upon which the problem can be defined.

Source: Ian I. Mitroff and James R. Emshoff. 1979. "On Strategic Assumption-Making: A Dialectical Approach to Policy and Planning." *The Academy of Management Review* 4 (1) (January): 1. doi:10.2307/257398. <http://dx.doi.org/10.2307/257398>.