Patterns of Systems Thinking

From SEBoK
Patterns of Systems Thinking

Lead Author: *Rick Adcock,* ***Contributing Authors:*** *Scott Jackson, Janet Singer, Duane Hybertson*

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

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 software engineering (e.g., Gamma et al. 1995; Buschmann et al. 1996). Their definitions portray a pattern as capturing design ideas as an archetypal and reusable description. A design pattern provides a generalized solution in the form of templates to a commonly occurring real-world problem within a given context. A design pattern is not a finished design that can be transformed directly into a specific solution. It is a description or template for how to solve a problem that can be used in many different specific situations (Gamma et al. 1995; Wikipedia 2012b). Alexander placed significant emphasis on the pattern role of reconciling and resolving competing forces, which is an important application of the yin yang principle.

Other examples of general patterns in both natural and engineered systems include: conventional designs in engineering handbooks, complex system models such as evolution and predator-prey models that apply to multiple application domains, domain taxonomies, architecture frameworks, standards, templates, architecture styles, reference architectures, product lines, abstract data types, and classes in class hierarchies (Hybertson 2009). Shaw and Garlan (Garlan 1996) used the terms pattern and style interchangeably in discussing software architecture. Lehmann and Belady (Lehmann 1985) examined a set of engineered software systems and tracked their change over time and observed regularities that they captured as evolution laws or patterns.

Patterns have been combined with model-based systems engineering (MBSE) to lead to pattern-based systems engineering (PBSE) (Schindel and Smith 2002, Schindel 2005).

Patterns also exist in systems practice, both science and engineering. At the highest level, Gregory (1966) defined science and design as behavior patterns:

> *The scientific method is a pattern of problem-solving behavior employed in finding out the nature of what exists, whereas the design method is a pattern of behavior employed in inventing things of value which do not yet exist.*

Regularities exist not only as positive solutions to recurring problems, but also as patterns of failure, i.e., as commonly attempted solutions that consistently fail to solve recurring problems. In software engineering these are called antipatterns, originally coined and defined by Koenig (Koenig 1995): An antipattern is just like a pattern, except that instead of a solution it gives something that looks superficially like a solution but isn't one. Koenig's rationale was that if one does not know how to solve a problem, it may nevertheless be useful to know about likely blind alleys. Antipatterns may include patterns of pathologies (i.e., common diseases), common impairment of normal functioning, and basic recurring problematic situations. These antipatterns can be used to help identify the root cause of a problem and eventually lead to solution patterns. The concept was expanded beyond software to include project management, organization, and other antipatterns (Brown et al. 1998; AntiPatterns Catalog 2012).

Patterns are grouped in the remainder of this section into basic foundational patterns and antipatterns (or patterns of failure).

Basic Foundational Patterns

The basic patterns in this section consist of a set of hierarchy and network patterns, followed by a set of metapatterns and SE patterns.

Hierarchy and Network Patterns

The first group of patterns are representative types of hierarchy patterns distinguished by the one-to-many relation type (extended from Hybertson 2009, 90), as shown in the table below. These are presented first because hierarchy patterns infuse many of the other patterns discussed in this section.

Table 1. Hierarchy Patterns. (SEBoK Original)

Relation	Hierarchy Type or Pattern
Basic: Repeating One-to-Many Relation	General: Tree structure
Part of a Whole	Composition (or Aggregation) hierarchy
Part of + Dualism: Each element in the hierarchy is a holon, i.e., is both a whole that has parts and a part of a larger whole	Holarchy (composition hierarchy of holons) (Koestler 1967) - helps recognize similarities across levels in multi-level systems
Part of + Interchangeability: The parts are clonons, i.e., interchangeable	Composition Hierarchy of Clonons (Bloom 2005). Note: This pattern reflects horizontal similarity.
Part of + Self-Similarity: At each level, the shape or structure of the whole is repeated in the parts, i.e., the hierarchy is self-similar at all scales.	Fractal. Note: This pattern reflects vertical similarity.
Part of + Connections or Interactions among Parts	System composition hierarchy
Control of Many by One	Control hierarchy—e.g., a command structure
Subtype or Sub-Class	Type or specialization hierarchy; a type of generalization
Instance of Category	Categorization (object-class; model-metamodel…) hierarchy; a type of generalization

Network patterns are of two flavors. First, traditional patterns are network topology types, such as bus (common backbone), ring, star (central hub), tree, and mesh (multiple routes) (ATIS 2008). Second, the relatively young science of networks has been investigating social and other complex patterns, such as percolation, cascades, power law, scale-free, small worlds, semantic networks, and neural networks (Boccara 2004; Neumann et al. 2006).

Metapatterns

The metapatterns identified and defined in the table below are from (Bloom 2005), (Volk and Bloom 2007), and (Kappraff 1991). They describe a metapattern as convergences exhibited in the similar structures of evolved systems across widely separated scales (Volk and Bloom 2007).

Table 2. Metapatterns. (SEBoK Original)

Name	Brief Definition	Examples
Spheres	Shape of maximum volume, minimum surface, containment	Cell, planet, dome, ecosystem, community
Centers	Key components of system stability	Prototypes, purpose, causation; Deoxyribonucleic acid (DNA), social insect centers, political constitutions and government, attractors
Tubes	Surface transfer, connection, support	Networks, lattices, conduits, relations; leaf veins, highways, chains of command

Binaries Plus	Minimal and thus efficient system	Contrast, duality, reflections, tensions, complementary/symmetrical/reciprocal relationships; two sexes, two-party politics, bifurcating decision process
Clusters, Clustering	Subset of webs, distributed systems of parts with mutual attractions	Bird flocks, ungulate herds, children playing, egalitarian social groups
Webs or Networks	Parts in relationships within systems (can be centered or clustered, using clonons or holons)	Subsystems of cells, organisms, ecosystems, machines, society
Sheets	Transfer surface for matter, energy, or information	Films; fish gills, solar collectors
Borders and Pores	Protection, openings for controlled exchange	Boundaries, containers, partitions, cell membranes, national borders
Layers	Combination of other patterns that builds up order, structure, and stabilization	Levels of scale, parts and wholes, packing, proportions, tiling
Similarity	Figures of the same shape but different sizes	Similar triangles, infant-adult
Emergence	General phenomenon when a new type of functionality derives from binaries or webs.	Creation (birth), life from molecules, cognition from neurons
Holarchies	Levels of webs, in which successive systems are parts of larger systems	Biological nesting from biomolecules to ecosystems, human social nesting, engineering designs, computer software
Holons	Parts of systems as functionally unique	Heart-lungs-liver (holons) of body
Clonons	Parts of systems as interchangeable	Skin cells (clonons) of the skin; bricks in constructing a house
Arrows	Stability or gradient-like change over time	Stages, sequence, orientation, stress, growth, meanders, biological homeostasis, growth, self-maintaining social structures
Cycles	Recurrent patterns in systems over time	Alternating repetition, vortex, spiral, turbulence, helices, rotations; protein degradation and synthesis, life cycles, power cycles of electricity generating plants, feedback cycles
Breaks	Relatively sudden changes in system behavior	Transformation, change, branching, explosion, cracking, translations; cell division, insect metamorphosis, coming-of-age ceremonies, political elections, bifurcation points
Triggers	Initiating agents of breaks, both internal and external	Sperm entering egg or precipitating events of war
Gradients	Continuum of variation between binary poles	Chemical waves in cell development, human quantitative and qualitative values

Systems Engineering Patterns

Some work has been done on various aspects of explicitly applying patterns to SE. A review article of

much of this work was written by Bagnulo and Addison (Bagnulo and Addison 2010), covering patterns in general, capability engineering, pattern languages, pattern modeling, and other SE-related pattern topics. Cloutier (Cloutier 2005) discussed applying patterns to SE, based on architecture and software design patterns. Haskins (Haskins 2005), and Simpson and Simpson (Simpson and Simpson 2006) discussed the use of SE pattern languages to enhance the adoption and use of SE patterns. Simpsons identified three high-level, global patterns that can be used as a means of organizing systems patterns:

- Anything can be described as a system.
- The problem system is always separate from the solution system.
- Three systems, at a minimum, are always involved in any system activity: the environmental system, the product system, and the process system.

Haskins (Haskins 2008) also proposed the use of patterns as a way to facilitate the extension of SE from traditional technological systems to address social and socio-technical systems. Some patterns have been applied and identified in this extended arena, described as patterns of success by Rebovich and DeRosa (Rebovich and DeRosa2012). Stevens (Stevens 2010) also discussed patterns in the engineering of large-scale, complex "mega-systems."

A common SE activity in which patterns are applied is in system design, especially in defining one or more solution options for a system-of-interest. See Synthesizing Possible Solutions for a discussion. The more specific topic of using patterns (and antipatterns, as described below) to understand and exploit emergence is discussed in the Emergence topic.

Patterns of Failure: Antipatterns

System Archetypes

The system dynamics community has developed a collection of what are called system archetypes. The concept was originated by Forrester (Forrester 1969), while Senge (Senge 1990) appears to have introduced the system archetype term. According to Braun (2002), the archetypes describe common patterns of behavior that help answer the question, "Why do we keep seeing the same problems recur over time?" They focus on behavior in organizations and other complex social systems that are repeatedly but unsuccessfully used to solve recurring problems. This is why they are grouped here under antipatterns, even though the system dynamics community does not refer to the archetypes as antipatterns. The table below summarizes the archetypes. There is not a fixed set, or even fixed names for a given archetype. The table shows alternative names for some archetypes.

Table 3. System Archetypes. (SEBoK Original)

Name (Alternates)	Description	Reference**
Counterintuitive Behavior	Forrester identified three "especially dangerous" counter-intuitive behaviors of social systems, which correspond respectively to three of the archetypes discussed below: (1) Low-Leverage Policies: Ineffective Actions; (2) High Leverage Policies: Often Wrongly Applied; and (3) Long-Term vs. Short-Term Trade-offs	F1, F2
Low-Leverage Policies: Ineffective Actions (Policy Resistance)	Most intuitive policy changes in a complex system have very little leverage to create change; this is because the change causes reactions in other parts of the system that counteract the new policy.	F1, F3, M

High Leverage Policies: Often Wrongly Applied (High Leverage, Wrong Direction)	A system problem is often correctable with a small change, but this high-leverage solution is typically counter-intuitive in two ways: (1) the leverage point is difficult to find because it is usually far removed in time and place from where the problem appears, and (2) if the leverage point is identified, the change is typically made in the wrong direction, thereby intensifying the problem.	F1, F3, M
Long-Term vs. Short-Term Trade-offs (Fixes that Fail, Shifting the Burden, Addiction)	Short-term solutions are intuitive, but in complex systems there is nearly always a conflict or tradeoff between short-term and long-term goals. Thus, a quick fix produces immediate positive results, but its unforeseen and unintended long-term consequences worsen the problem. Furthermore, a repeated quick fix approach makes it harder to change to a more fundamental solution approach later.	F1, F3, M, S, B
Drift to Low Performance (Eroding Goals, Collapse of Goals)	There is a strong tendency for complex system goals to drift downward. A gap between current state and goal state creates pressure to lower the goal rather than taking difficult corrective action to reach the goal. Over time the continually lowered goals lead to crisis and possible collapse of the system.	F1, F3, M, B
Official Addiction – Shifting the Burden to the Intervener	The ability of a system to maintain itself deteriorates when an intervener provides help and the system then becomes dependent on the intervener	M, S
Limits to Growth (a.k.a. Limits to Success)	A reinforcing process of accelerating growth (or expansion) will encounter a balancing process as the limit of that system is approached and continuing efforts will produce diminishing returns as one approaches the limits.	S, B
Balancing Process with Delay	Delay in the response of a system to corrective action causes the correcting agent to either over-correct or to give up due to no visible progress.	S
Escalation	Two systems compete for superiority, with each escalating its competitive actions to get ahead, to the point that both systems are harmed.	B
Success to the Successful	Growth leads to decline elsewhere. When two equally capable systems compete for a limited resource, if one system receives more resources, it is more likely to be successful, which results in it’s receiving even more resources, in a reinforcing loop.	S, B
Tragedy of the Commons	A shared resource is depleted as each system abuses it for individual gain, ultimately hurting all who share it.	H, S, B
Growth and Underinvestment	In a situation where capacity investments can overcome limits, if such investments are not made, then growth stalls, which then rationalizes further underinvestment.	S, B
Accidental Adversaries	Two systems destroy their relationship through escalating retaliations for perceived injuries.	B
Attractiveness Principle	In situations where a system faces multiple limiting or impeding factors, the tendency is to consider each factor separately to select which one to address first, rather than a strategy based on the interdependencies among the factors.	B

** **B**—(Braun 2002); **F1**—(Forrester 1969); **F2**—(Forrester 1995); **F3**—(Forrester 2009); **H**—(Hardin 1968); **M**—(Meadows 1982); **S**—(Senge 1990).

Relations among system archetypes were defined by Goodman and Kleiner (Goodman and Kleiner 1993/1994) and republished in Senge et al. (Senge et al. 1994).

Software and Other Antipatterns

Antipatterns have been identified and collected in the software community in areas that include: Architecture, development, project management, user interface, organization, analysis, software design, programming, methodology, and configuration management (AntiPatterns Catalog 2012, Wikibooks 2012). A brief statement of three of them follows; the first two are organization and the third is software design.

- Escalation of commitment - Failing to revoke a decision when it proves wrong.
- Moral hazard - Insulating a decision-maker from the consequences of his or her decision.
- Big ball of mud - A system with no recognizable structure,

A link between the software community and the system archetypes is represented in a project at the Software Engineering Institute (SEI) (2012), which is exploring the system archetypes in the context of identifying recurring software acquisition problems as “acquisition archetypes.” They refer to both types of archetypes as patterns of failure.

Another set of antipatterns in the general systems arena has been compiled by Troncale (Troncale 2010; Troncale 2011) in his systems pathologies project. Sample pathology types or patterns include:

- Cyberpathologies - Systems-level malfunctions in feedback architectures.
- Nexopathologies - Systems-level malfunctions in network architectures or dynamics.
- Heteropathologies - systems-level malfunctions in hierarchical, modular structure & dynamics.

Some treatments of antipatterns, including Senge (Senge 1990) and SEI (SEI 2012), also provide some advice on dealing with or preventing the antipattern.

Patterns and Maturity

Patterns may be used as an indicator of the maturity of a domain of inquiry, such as systems science or systems engineering. In a mature and relatively stable domain, the problems and solutions are generally understood and their similarities are captured in a variety of what are here called patterns. A couple of observations can be made in this regard on the maturity of systems science in support of systems engineering.

In the arenas of physical systems and technical systems, systems science is relatively mature; many system patterns of both natural physical systems and engineered technical systems are reasonably well defined and understood.

In the arena of more complex systems, including social systems, systems science is somewhat less mature. Solution patterns in that arena are more challenging. A pessimistic view of the possibility of science developing solutions to social problems was expressed by Rittel and Webber (Rittel and Webber 1973) in their classic paper on wicked problems: “The search for scientific bases for confronting problems of social policy is bound to fail, because . . . they are ‘wicked’ problems, whereas science has developed to deal with ‘tame’ problems.” A more optimistic stance toward social problems has characterized the system dynamics community. They have been pointing out for over 40 years the problems with conventional solutions to social problems, in the form of the system archetypes and associated feedback loop models. That was an important first step. Nevertheless, they have had difficulty achieving the second step; producing social patterns that can be applied to solve those problems. The antipatterns characterize problems, but the patterns for solving those

problems are elusive.

Despite the difficulties, however, social systems do exhibit regularities, and social problems are often solved to some degree. The social sciences and complex systems community have limited sets of patterns, such as common types of organization structures, common macro-economic models, and even patterns of insurgency and counter-insurgency. The challenge for systems science is to capture those regularities and the salient features of those solutions more broadly, and make them explicit and available in the form of mature patterns. Then perhaps social problems can be solved on a more regular basis. As systems engineering expands its scope from the traditional emphasis on technical aspects of systems to the interplay of the social and technical aspects of socio-technical systems, such progress in systems science is becoming even more important to the practice of systems engineering.

References

Works Cited

Alexander, C. 1979. *The Timeless Way of Building*. New York: Oxford University Press.

Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. 1977. *A Pattern Language: Towns – Buildings – Construction*. New York: Oxford University Press.

ATIS. 2008. *ATIS Telecom Glossary 2007*. Washington, D.C.: Alliance for Telecommunications Industry Solutions. Accessed December 3 2014 at ATIS http://www.atis.org/glossary/definition.aspx?id=3516.

Bagnulo, A. and T. Addison. 2010. *State of the Art Report on Patterns in Systems Engineering and Capability Engineering.* Contract Report 2010-012 by CGI Group for Defence R&D Canada – Valcartier. March 2010.

Bloom, J. 2005. "The application of chaos, complexity, and emergent (meta)patterns to research in teacher education." *Proceedings of the 2004 Complexity Science and Educational Research Conference* (pp. 155-191), Sep 30–Oct 3 • Chaffey's Locks, Canada. http://www.complexityandeducation.ca.

Boccara, N. 2004. *Modeling Complex Systems*. New York, NY: Springer-Verlag.

Braun, T. 2002. "The System Archetypes." Accessed December 3 at http://www.albany.edu/faculty/gpr/PAD724/724WebArticles/sys_archetypes.pdf

Brown, W., R. Malveau, H. McCormick, and T. Mowbray. 1998. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons.

Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. 1996. *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, U.K.: John Wiley.

Cloutier, R. 2005. "Toward the Application of Patterns to Systems Engineering." 'Proceedings of the Conference on Systems Engineering Research (CSER) 2005, March 23-25, Hoboken, NJ, USA.

Forrester, J. 1969. *Urban Dynamics*. Waltham, MA: Pegasus Communications.

Forrester, J. 1995. "Counterintuitive Behavior of Social Systems." *Technology Review.* 73(3), Jan. 1971: 52-68.

Forrester, J. 2009. Learning through System Dynamics as Preparation for the 21st Century.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.

Goodman, G. and A. Kleiner. 1993/1994. “Using the Archetype Family Tree as a Diagnostic Tool.” *The Systems Thinker.* December 1993/January 1994.

Gregory, S. 1966. "Design and the design method," in S. Gregory (ed.). *The Design Method*. London, Engladh: Butterworth.

Hardin, G. 1968. "The Tragedy of the Commons." *Science.* 162(13 December 1968):1243-1248. DOI: 10.1126/science.162.3859.1243.

Haskins, C. 2005. "Application of Patterns and Pattern Languages to Systems Engineering." *Proceedings of the 15th Annual INCOSE International Symposium*. Rochester, NY, July 10-13, 2005.

Haskins, C. 2008. "Using patterns to transition systems engineering from a technological to social context." *Systems Engineering*, 11(2), May 2008: 147-155.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL: Auerbach/CRC Press.

Kappraff, J. (1991). *Connections: The geometric bridge between art and science*. New York, NY: McGraw-Hill.

Koenig, A. 1995. "Patterns and Antipatterns". *Journal of Object-Oriented Programming.* 8(1)March/April 1995: 46–48.

Koestler, A. 1967. *The Ghost in the Machine*. New York, NY: Macmillan.

Lehmann, M. and L. Belady. 1985. *Program Evolution*. London, England: Academic Press.

Meadows, D. 1982. Whole Earth Models and Systems. *The Co-Evolution Quarterly.* Summer 1982: 98-108.

Odum, H.1994. *Ecological and General Systems: An Introduction to Systems Ecology (Revised Edition).* Boulder, CO: University Press of Colorado.

Rebovich, G. and J. DeRosa 2012. "Patterns of Success in Systems Engineering of IT-Intensive Government Systems." *Procedia Computer Science.* 8(2012): 303 – 308.

Rittel, H. and M. Webber. 1973. "Dilemmas in a general theory of planning." *Policy Sciences.* 4:155–169.

Schindel, W. 2005. "Pattern-based systems engineering: An extension of model-based systems engineering." INCOSE TIES tutorial presented at 2005 INCOSE Symposium, 10-15 July 2005, Rochester, NY.

Schindel, W. and V. Smith. 2002. *Results of applying a families-of-systems approach to systems engineering of product line families.* Technical Report 2002-01-3086. SAE International.

SEI 2012. Patterns of Failure: System Archetypes. Accessed December 3 2014 at SEI http://www.sei.cmu.edu/acquisition/research/pofsa.cfm

Senge, P. 1990. *The Fifth Discipline: Discipline: The Art and Practice of the Learning Organization*. New York, NY: Currency Doubleday.

Senge, P., A. Kleiner, C. Roberts and R. Ross. 1994. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization*. New York, NY: Currency Doubleday.

Shaw, M. and D. Garlan. 1996. *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice Hall.

Simpson, J. and M. Simpson. 2006. "Foundational Systems Engineering Patterns for a SE Pattern

Language." *Proceedings of the 16th Annual INCOSE Symposium, July, 2006, Orlando, FL.*

Stevens, R. 2011. *Engineering Mega-Systems: The Challenge of Systems Engineering in the Information Age*. Boca Raton, FL: Auerbach/Taylor & Francis.

Troncale, L. 2010. "Would a Rigorous Knowledge Base in “Systems Pathology” Add to the S.E. Portfolio?" Presented at 2010 LA Mini-Conference, 16 October 2010, Loyola Marymount University, Los Angeles, CA.

Troncale, L. 2011. “Would A Rigorous Knowledge Base in Systems Pathology Add Significantly to the SE Portfolio?" Proceedings of the Conference on Systems Engineering Research (CSER), April 14-16, Redondo Beach, CA.

Volk, T., and J.W. Bloom. 2007. "The use of metapatterns for research into complex systems of teaching, learning, and schooling. Part I: Metapatterns in nature and culture." *Complicity: An International Journal of Complexity and Education*, 4(1): 25—43.

Wikibooks. 2012a. "AntiPatterns." http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Architecture/Anti-Patterns.

Wikipedia. 2012b. "Software design pattern." http://en.wikipedia.org/wiki/Software_design_pattern

Primary References

Alexander, C. 1979. *The Timeless Way of Building*. New York, NY: Oxford University Press.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Bloom, J. 2005. "The application of chaos, complexity, and emergent (meta)patterns to research in teacher education." *Proceedings of the 2004 Complexity Science and Educational Research Conference* (pp. 155-191), Sep 30–Oct 3, Chaffey’s Locks, Canada. http://www.complexityandeducation.ca.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Additional References

Principia Cybernetica. 1996. "Cybernetics and Systems Theory." Accessed 21 April 2013. Available at: http://pespmc1.vub.ac.be/CYBSYSTH.html

Erl, T. 2009. *SOA: Design Patterns*. Upper Saddle River, NJ: Prentice Hall.

Erl, T. 2008. *SOA: Principles of Service Design*. Upper Saddle River, NJ: Prentice Hall.

Francois, F. (ed.). 2004. *International Encyclopedia of Systems and Cybernetics*, 2nd ed.. Munich, Germany: K. G. Saur Verlag.

Meyers, R. (ed.). 2009. *Encyclopedia of Complexity and Systems Science.* New York, NY: Springer.

Midgley, G. (ed.). 2003. *Systems Thinking.* Thousand Oaks, CA: Sage Publications Ltd.

Principia Cybernetica Web. 2013. "Web Dictionary of Cybernetics and Systems." Accessed 21 April 2013. Available at: http://pespmc1.vub.ac.be/ASC/indexASC.html

---
