13th INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
August 2001,Glasgow, Scotland, UK

TOWARDS A THEORY OF COMPLICATEDNESS: FRAMEWORK FOR COMPLEX SYSTEMS ANALYSIS AND DESIGN

Victor Tang, Vesa Salminen

Keywords: theory of technical systems, complexity management, adaptive design

1. Introduction

Global, dynamic, and competitive business environment has increased the complexity in product, service, operational processes and human side. Much engineering effort goes into reducing systems complexity. We argue that the real issue is reducing complicatedness. This is an important distinction. Complexity can be a desirable property of systems provided it is *architected complexity* that reduces *complicatedness.* Complexity and complicatedness are not synonyms. Complexity is an inherent property of systems; complicatedness is a derived function of complexity. We introduce the notion of complicatedness of complex systems, present equations for each and show they are separate and distinct properties. To make these ideas actionable, we present a design methodology to address complicatedness. We show examples and discuss how our equations reflect the fundamental behavior of complex systems and how our equations are consistent with our intuition and system design experience. We discuss validation experiments with global firms and address potential areas for further research. We close with a discussion of the implications for systems design engineers. As engineers, we believe our strongest contributions are to the analysis, design, and managerial practice of complex systems analysis and design.

We illustrate the difference between complexity and complicatedness. Relative to a manual transmission, a car's automatic transmission has more parts and more intricate linkages. It is more *complex.* To drivers, it is unquestionably less *complicated*, but to mechanics who have to fix it, it is more *complicated.* This illustrates a fundamental fact about systems; *decision units* act on systems to manage their behavior. Complexity is an inherent property of systems. Complicatedness is a derived property that characterizes an execution unit's ability to manage a complex system. A system of complexity level, $\mathbf{C_a}$**,** may present different degrees of complicatedness, **K,** to distinct execution units **E** and **F**; $\mathbf{K_E}=\mathbf{K_E}(\mathbf{C_a}) \neq \mathbf{K_F}=\mathbf{K_F}(\mathbf{C_a})$.

We summarize relevant literature on systems complexity in Figure 1. Columns [1] to [15] are keyed to the references. Rows identify key areas of research results; e.g., metrics, complicatedness, etc. We make four observations about the locus of results. One, there is a dearth of quantitative frameworks or metrics. There is no research on complicatedness and complexity as distinct properties of systems. Two, research seems to cluster around engineering management and physical products. The focus is on modularization and interactions with a bias to linear systems and qualitative metrics. Three, there are efforts on methodologies and tools, but theory, foundations and software have a demonstrably lesser presence. Ferdinand's work in software systems complexity is a happy exception [1]. It is analytical, rigorous and elegant. Three, services and enterprise solutions are barely addressed.

This is a serious omission given the high proportion of services in industrialized economies. Fourth, although layering of abstract systems and reintegration have a long history; the literature is skewed to decomposition rather than integration.

Reference	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
Main Focus of Investigation															
Foundations & Theory	■	□	□	□	□	□	□	■	■	□	■	■	□	□	□
Linear Systems Engineering	■	□	■	□	■	□	□	■	□	■	■	■	□	□	■
Non-linear Systems Engineering	□	■	□	□	□	■	□	□	■	□	□	□	■	■	□
Systems Architecture/Structure	■	□	□	□	■	■	■	■	■	■	■	■	■	■	■
Management	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Design Methodology	□	□	□	□	■	□	■	■	□	■	■	■	■	■	■
Design Tools	□	□	□	□	□	□	■	□	□	□	□	□	□	□	■
Complicatedness	■	□	□	□	□	□	□	□	□	□	□	□	■	□	□
Strategy to Address Complexity															
Modularization & decomposition	■	□	■	□	■	■	■	■	□	■	■	■	■	■	■
Interactions and Dependencies	■	■	■	□	■	■	■	■	■	■	■	■	■	■	■
Layering & Abstraction	■	□	□	□	□	■	□	□	□	■	□	□	■	□	□
Integration	□	□	□	□	□	□	■	□	□	□	□	□	■	□	□
Complexity Metrics															
Quantitative	■	□	□	□	□	□	□	■	□	□	□	□	□	□	■
Qualitative	□	■	■	■	■	■	■	□	■	■	■	■	■	■	■
Domain															
Physical Products & Systems	□	□	■	□	■	■	■	■	■	■	■	■	■	■	■
Software Products & Systems	■	□	□	□	■	□	□	□	□	□	□	□	■	□	□
Services	□	□	□	□	□	■	□	□	□	□	□	□	■	□	□
Enterprise Solutions	□	□	□	□	□	□	□	□	□	□	□	□	■	□	□
Social & Organizational	□	■	□	■	■	□	□	□	□	□	□	□	□	■	■
Applications															
Engineering	□	□	■	□	□	■	■	■	■	■	■	■	■	■	■
Organizational Theory	□	■	□	■	■	□	□	□	□	□	□	□	□	■	□
Quality Management	■	□	□	□	□	□	□	□	□	□	■	□	□	□	□

■ Indicates a strong element in the publication. □ Indicates a lesser or absent element.

Figure 1. Systems Complexity Summary of the Literature

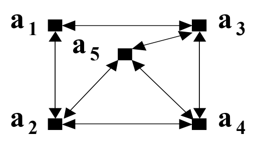
2. Complexity

Overwhelmingly, the literature considers a system with a large number of elements as complex. Very few address the linkages among the elements and no one, to our knowledge, considers their bandwidth. All these factors are inherent characteristics of systems. Therefore, we argue that the number of elements, the number of *interactions* among them and the *bandwidth* of these interactions determine complexity of the system. As any of these increases, we expect complexity to increase. For example, a system $\mathbf{N}=\{\mathbf{n}_i\}_{i=1,2,\ldots,p}$ with binary interactions among the elements. Complexity, $\mathbf{C_N}$, of this system does not exceed p^2, we denote this by $\mathbf{C_N}=\mathbf{O}(p^2)$. System $\mathbf{M}=\{\mathbf{m}_j\}_{j=1,2,\ldots,p}$ can have complexity $\mathbf{C_M}=\mathbf{O}(p^k)$ where $k>2$. When $\mathbf{M}$ admits $\{\mathbf{m}_j \mathbf{x} \mathbf{m}_r\}_{jr}$ and $\{\mathbf{m}_j \mathbf{x} \mathbf{m}_r \mathbf{x} \mathbf{m}_s\}_{jrs}$ interactions, $\mathbf{C_M}=\mathbf{O}(p^3)$. If $\mathbf{M}$ admits $\{\mathbf{m}_j \mathbf{x} \mathbf{m}_r \mathbf{x} \mathbf{m}_s \mathbf{x} \mathbf{m}_t\}_{jrst}$ interactions, $\mathbf{C_M}=\mathbf{O}(p^4)$. This characterization of complex systems admits systems with feedback loops of arbitrary nesting and depth, and high bandwidth interactions among system elements. Complexity is a monotonically increasing function as the size of the system size, number of interactions increases, and bandwidth of interactions increase. In the limit, complexity$\rightarrow\infty$. We define complexity by $\mathbf{C}=\mathbf{X}^n\boldsymbol{\Sigma}_b\,\mathbf{B}^b$

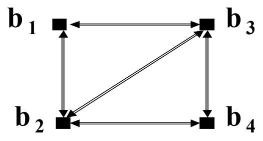
where $\quad\mathbf{X}$ is an integer denoting the number of elements $\{x_e\}_{e=1,\ldots,p}$
n is the integer indicated in the relation $O(p^n)$

and $\quad\mathbf{B_1}=\Sigma_{ij}\lambda_{ij}\beta_{ij}$
λ_{ij} is the number of linkages between x_i and x_j
β_{ij} is the bandwidth of the linkages between x_i and x_j
$\mathbf{B_2}=\Sigma_k\lambda_k^{ij}\ \beta_k^{ij}$
λ_k^{ij} is the number of linkages between x_k and (x_i,x_j)
β_k^{ij} is the bandwidth of the linkages between x_k and (x_i,x_j) and in general,
$\mathbf{B_n}=\Sigma_n\lambda_p^{ijk\ldots n-1}\ \beta_n^{ijkl\ldots n-1}$
$\lambda_n^{ijkl\ldots n-1}$ number of linkages among x_k and $(x_i,x_j),(x_i,x_j,x_k),\ldots,(x_i,x_j,x_k,x_{k,\ldots},x_{n-1})$
$\beta_n^{ijkl\ldots n-1}$ linkage bandwidth among x_k and $(x_i,x_j),(x_i,x_j,x_k),\ldots,(x_i,x_j,x_k,x_{k,\ldots},x_{n-1})$

B is a measure of the information capacity among the elements of the system. Note that the monotinicity properties are not violated. In Figure 2 we give an example.



$A=\{a_1,a_2,a_3,a_4,a_5\}$
β_{ij}=bandwidth between $\{a_i,a_j\}$
Let $\beta_{ij}=1$ bit
$B=\Sigma_i\ \Sigma_j\ \lambda_{ij}\ \beta ij=7$
$C=X^nB^n=(5)^2(7)^2=1225$



$B=\{b_1,b_2,b_3,b_4\}$
β_{ij}=bandwidth between $\{b_i,b_j\}$
Let $\beta_{ij}=2$ bits
$B=\Sigma_i\ \Sigma_j\ \lambda_{ij}\ \beta_{ij=}10$
$C=X^nB^{n=}(4)^2(10)^2=1600$

Figure 2. Complexity Example of Two Systems

3. Complicatedness

Complicatedness is the degree to which a decision unit for the system is able to manage the level of complexity presented by the system. The decision unit can be another system or a person. Complicatedness is a function of complexity, **K**=**K**(**C**). Let us explore the properties we expect from a complicatedness function. We expect monotonicity of complexity is imposed on complicatedness, but do not expect that they are identical. Clearly at **C**=0, **K**=0.

Consider **K** when $\mathbf{C}\rightarrow\infty$. Intuitively, there is a level of complexity at which the decision unit can barely cope with the system. The system is becoming unmanageable. For example, most people can visualize a graph, **g**=**g**(x,y) of $\mathbf{C}_g=\mathbf{O}(p^2)$, but it is harder for h=h(x,y,z) with $\mathbf{C}_h=\mathbf{O}(p^3)$. Few can visualize a surface four variables, although complexity has only reached $\mathbf{O}(p^4)$. Consider, equally incomprehensible systems **A** and **B** where $\mathbf{C_A}=\mathbf{O}(p^{100})$ and

$\mathbf{C_B}$=$\mathbf{O}$($p^{100,000}$) respectively; $\mathbf{K_A} \gtrsim \mathbf{K_B}$ although $\mathbf{O}(p^{100,000}) >> \mathbf{O}(p^{10,000})$. Therefore, when $\mathbf{C}=0$, $\mathbf{K}=0$ and when $\mathbf{C} \rightarrow \infty$, $\mathbf{K} \rightarrow \mathbf{K_{max}}$ asymptotically.

Systems are designed to operate and be managed around at an optimal point of complexity, say **C***. For **C<C***, although complexity increases, it is well within the interval of manageability. At **C=C*** the system complexity is optimal for the decision unit. For **C>C***, complexity is increasing and the decision unit can manage the system with decelerating effectiveness. Mathematically, $d\mathbf{K}/d\mathbf{C}>0$ in the open interval $(0, \infty)$. At **C=C***, $d\mathbf{K}/d\mathbf{C}=0$ and $d^2\mathbf{K}/d\mathbf{C}^2=0$. Complicatedness has reached an inflection point. So that for **C>C***, $d^2\mathbf{K}/d\mathbf{C}^2<0$, i.e., complicatedness is reaching saturation. The decision unit's ability to manage complexity has reached diminishing returns. For **C<C***, $d^2\mathbf{K}/d\mathbf{C}^{2>}0$, complexity is growing faster than complicatedness. Because the logistic function is one of the simplest mathematical expressions that has all the above properties, figure 3. We adopt it to express complicatedness. $\mathbf{K}(\mathbf{C})=\mathbf{K_{max}}/(1+e^{-\alpha C})$

where e is the transcendental number e=3.2718 2818 284…
α is a constant specific to the decision unit
C is the complexity of the system

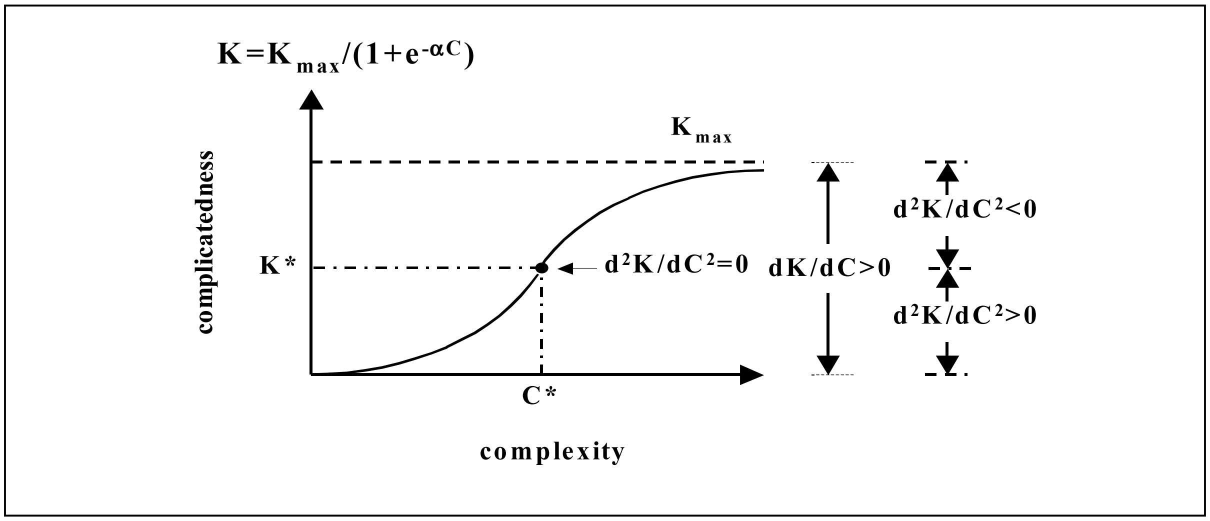


Figure 3. Complexity and Complicatedness

Without loss of generality, we set $\mathbf{K_{max}}=1$ to indicate abject complicatedness. There are other functions that can be used; such as, the Gompertz curve, Weibull distribution, log-reciprocal function, etc. The major differences are the location of the inflection point, the growth pattern before and after the inflection point, and the symmetry around the inflection point.

4. Examples of Uncomplicated Complex Systems

Earlier we presented the automobile transmission as a complex system that is uncomplicated. Neural networks are more interesting as a systems engineering example. Typically they are applied to situations where there are an intractable number of data points to analyze in order to set a course of action. To solve this difficulty, the neural network is layered, Figure 4. The complexity has increased relative to the input vector. Many new elements, new interactions and their bandwidth have all increased the initial complexity. But *architected complexity* has reduced an intractably complicated input vector to an output vector that now makes the system manageable. This approach has proven effective for engineering paper machines [16]. This is a non-trivial example. The purchase price of paper machine ranges around \$50 M. The mill generates about 10^9 data points, which are processed in real-time by adaptive and distributed neural networks embedded in the machine.

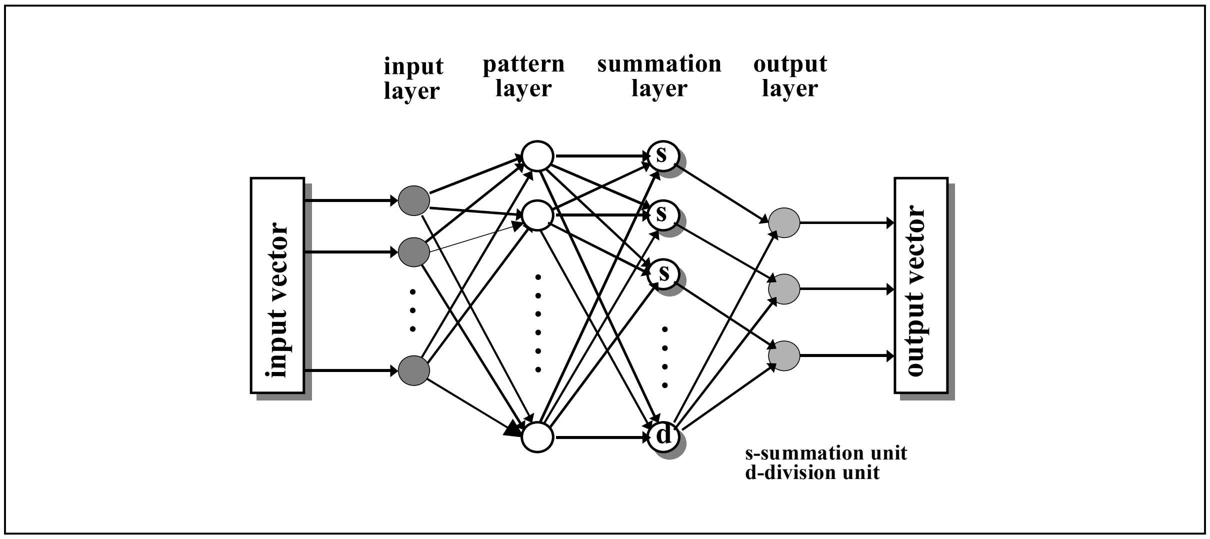


Figure 4. Use of Neural Network as Architected Complexity

The telecomm infrastructure is one of the most massive systems in the world. On demand, it interoperates an immense array of networks, products and computers. The system complexity is enormous, yet we routinely make transcontinental telephone calls and download music and pictures from the Web. *Architected complexity* has made telecomm networks manageable. Engineers created the OSI Reference Model by partitioning the network functions into distinct layers. This architectural innovation creates, at each level, a distinct presentation of the network that is more abstract at each successive layer. Each layer presents to decision units a specific *system image* of the network that is vastly less complicated. Layering system images is a widely adopted doctrine in computers; e.g. programming languages. With the first computers, applications programming was very difficult. Programmers had to embed arcane hardware details into their algorithms. High Level Languages were invented to present an abstract, but domain specific, system image for programming. A layer of software hid and encapsulated, transparently to the programmer, all machine specificities. *Architected complexity* is a very effective complexity management strategy; it reduces complicatedness.

5. Examples of Complicated Complexity

It suffices to present three examples. The typical VCR control panel is a classic example of complex and complicated design. Another example is PC software or “bloatware.” So many application packages are functionally so extravagant that the average person can learn only a fraction of their functionality. Cellular phones are in danger of becoming examples of complex and complicated products.

6. Calibrating Complicatedness

Consider a car's transmission. The automatic transmission presents the well known *system image* of **A**={P,R,N,D1,D2,D3}, $\lambda_{ij}=24$ with $\beta_{ij}=1$; thus $\mathbf{C_A}=6^2(24)(1)=864$. The manual transmission presents a *system image* of **M**={P,R,N,D1,D2,D3,F} where F is the foot clutch. It needs to be engaged and disengaged, so F's interaction bandwidth is 2. $\lambda_{ij}=10$ with $\beta_{ij}=1$, and $\lambda_{mn}=14$ with $\beta_{mn}=2$, thus $\mathbf{C_M}=7^2[10+(14)2]^2=38416$. For the novice driver, $\mathbf{C^*}\approx\mathbf{C_A}=864$. At $C\approx 40000$, we can say that $\mathbf{K_{max}}=1$. Therefore, we can create instruments to determine the analytic form of the complicatedness function. For a system with complexity C, and a decision unit **K,** we design an instrument to perform these functions:

[1] determine the optimal complexity, **C*** that **K** can manage optimally
[2] in the (C,K) space, at **C*** set **K*=**1/2
[3] solve for α using equation $1/2=1/(1+e^{-\alpha C^*})$, recall that and $\mathbf{K_{max}}=1$
[4] get $\mathbf{K}(\mathbf{C})=1/(1+e^{-\alpha C})$.

7. Engineering Complex but Uncomplicated Systems

There is good and bad cholesterol. Similarly, there is *architected* and *unarchitected complexity*. The former reduces complicatedness; the latter does not. There are two important principles in *architected complexity:* partition the system into modules, reintegrate them while maintaining system integrity. Many decomposition schemes address the first principle. Karnaugh maps for digital circuits, Djysktra architectures for computers, Design Structure Matrix for mechanical products [15], etc. They are effective tools, but when the decomposition creates a large number of new components and interactions, the result can now become intolerably complicated and make reintegration impractical. Reintegration is less visible in research, although widely practiced by engineers.

Consider system $\mathbf{M}=\{m_j\}_j$, $C_M>K_{\mathbf{M}}^*$ for **M**'s decision unit, figure 5.

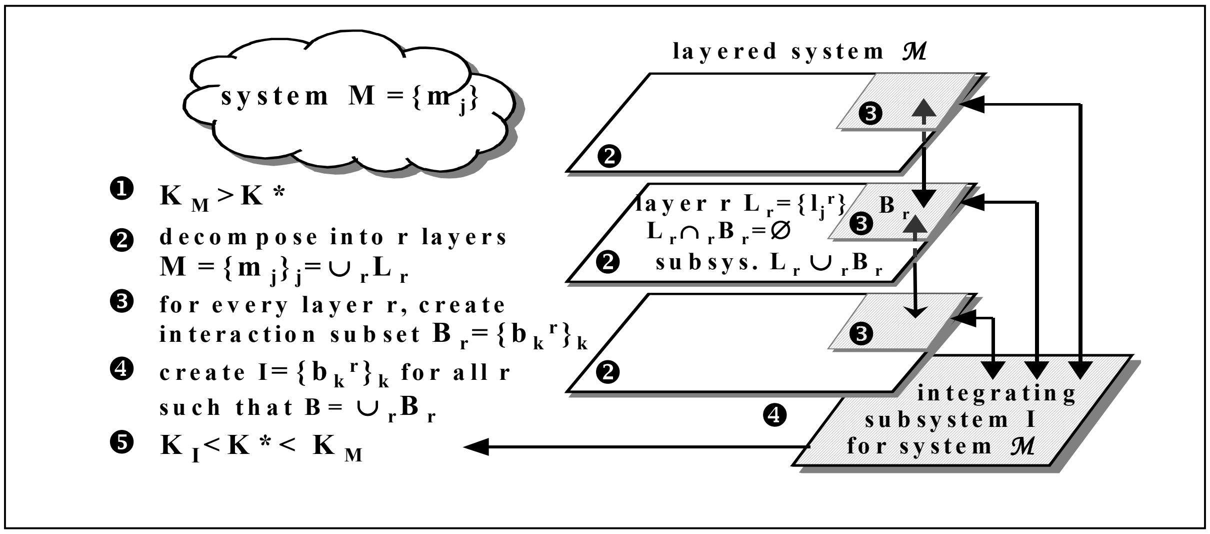

layered system $\mathcal{M}$
system $M=\{m_j\}$
❶ $K_M > K^*$
❷ decompose into r layers $M=\{m_j\}_j=\cup_r L_r$
❸ for every layer r, create interaction subset $B_r=\{b_k^r\}_k$
❹ create $I=\{b_k^r\}_k$ for all r such that $B=\cup_r B_r$
❺ $K_I < K^* < K_M$
layer r $L_r=\{l_j^r\}$ B_r
$L_r \cap_r B_r=\varnothing$
subsys. $L_r \cup_r B_r$
❷
❸
❹
integrating subsystem I for system $\mathcal{M}$

Figure 5. Architected Complexity to reduce Complicatedness

The goal is to architect complexity so that **M** is transformed into $\mathcal{M}$, such that $K_{\mathcal{M}}<K_{\mathbf{M}^*}$, although $C_{\mathcal{M}}>C_{M^*}$. Partition **M** into layers, $\mathbf{L}_r=\{\mathbf{l}_j^r\}_j$ such that $\mathbf{M}=\cup_r\mathbf{L}_r$, i.e., all the elements of

M appear in some specific layer. Functional decomposition is an engineering application of this principle. For a paper mill, these can be the mechanical, control, and process domains [13], or for a computer, the arithmetic unit, main memory, the I/O units, etc. Design the layers so that there are only intra level interactions among the elements of a layer. Create $\mathbf{B}_r=\{\mathbf{b}_k^r\}_k$ for ever layer $\mathbf{L}_r$, so that $\mathbf{L}_r \cap_r \mathbf{B}_r=\emptyset$. Design **B** so that only elements of **B** communicate with each other. $\mathbf{B}=\cup_r \mathbf{B}_r$ is $\mathcal{M}$'s communications subsystem. For $\mathcal{M}$, design a system integration unit $\mathbf{T}=\{t_x\}_x$, which on one side interfaces with **B** and the other with the decision unit. Note that T presents the decision unit with an *image* of the system $\mathcal{M}$. This is a hallmark of a good architecture. Good design always presents a less complicated *system image* to a decision unit.

8. Areas of Potential Research

Complexity should be studied further with industry experiments. Begin by selecting a set of simple systems with unambiguously matched by an identifiable class of decision units. Then calculate the systems' complexity and derive complicatedness functions for the set of decision units. These experiments would serve as case studies for the behavior of the complicatedness function and help determine whether our simple logistics function serves its purpose well. If not, different analytic functions should be tried as suggested earlier. In addition, experiments in an organizational setting should also be studied. The decision units are people with specific complicatedness functions. An executive who must negotiate with a large number of departments is such an example. In this case the system elements are the department heads, the bandwidth of interactions is determined by the specific nature of the negotiations. The extent to which organizational structures, communication styles and boundary objects are effective *architected complexity* are fruitful areas of investigation [17].

9. Conclusions

Separating complicatedness and complexity improves the clarity by which systems can be described and analyzed. In this way we can clearly separate what is an inherent property of the system, complexity, from a derived attribute, which is complicatedness. The mathematical expressions we formulate capture additional properties of systems that have heretofore remained largely unaddressed. We are able to derive results that give us valuable insight into the behavior of systems. These insights are useful in the analysis and design of very large complex systems, and also move us towards a theory of complicatedness.

References

[1] Ferdinand, A.E., Systems, Software and Quality Engineering. New York: Van Nostrand Reinhold, 1993.

[2] Kelly, S. and Allison, M-A, The Complexity Advantage. Business Week Books, McGraw-Hill, 1998.

[3] Khurana, A., Managing Complex Production Processes. Sloan Management Review, Winter 1999, pp.85-97.

[4] Bar-Yam, Y., New England Complex Systems Institute (NECSI) home page: http://www.necsi.org/

[5] Levitt, R., Thomsen, J., Christiansen, T., Kunz, J., Jin, Y., Nass, C., Simulating Project Work Processes and Organizations: Towards a Micro-Contingency

Theory of Organizational Design. Management Science, Vol. 45, No.11, November 1999, pp.1479-1495.

[6] Salminen, V. and Pillai, B., Non-linear Dynamics of Interacting Mechanisms in Distributed Product Development. International Conference on Complex Systems (ICCS), May 21-26, 2000, Nashua, NH.

[7] Salminen, V., Yassine, A., Riitahuhta, A., Strategic Management of Complexity in Distributed Product Development, NordDesign 2000, 24-25th August 2000.

[8] Suh, N.P., A Theory of Complexity, Periodicity, and the Design Axioms, Research in Engineering Design, Vol. 11, 1999, pp. 116-131.

[9] Holland, J., Hidden Order: How Adaptation Builds Complexity. Addison Wesley publishing Company, Reading, MA, 1995.

[10] Baldwin, C., Clark, K., Managing in an Age of Modularity. Harvard Business Review, September-October 1997.

[11] Boppe, C., 16.880- Systems Engineering. MIT-report 1998.

[12] Warfield, J.N., A Structure-Based Science of Complexity: Transforming Complexity Into Understanding. Kluver Publishing, Amsterdam, 2000.

[13] Tang, V., Salminen, V., Pillai, B, Extending DSM to Very Complex Systems. Design Structure Matrix World wide-conference, 18-19 September 2000, MIT.

[14] Gharajedaghi, J., System Thinking: Managing Chaos and Complexity. Butterworth- Heineman Publishing, 1999.

[15] Eppinger, S.D. Innovation at the Speed of Information, Harvard Business Review, vol. 79, no. 1, pp. 149-158, January 2001.

[16] Pillai, B., Adaptation of Neural Network and Fuzzy Logic for the Wet-End-Control and Process Management of Paper or Board Machines: A Tool Making Approach. Acta Polytechnica Scandinavica, Mech.Engineering Series No. 138.

[17] Carlile, P., Lucas, B., Cross-Boundary Work as Cross-Boundary Development. Case Study. ICRMOT Working Paper. MIT Sloan School of Management.

Contact Address
Victor Tang and Vesa Salminen
Massachusetts Institute of Technology, Center for Innovation in Product Development, CIPD
30 Memorial Drive, E60-236,
Cambridge, MA 02139 USA

Tel: +1.617.457.2741
Fax: +1.617.258.0485
Email: victang@mit.edu, vesas@mit.edu