Can We Engineer the Emergent Behavior of A System-of-Systems?

Moderator: J. C Hsu,The Royal Academy of Engineering, Queens University

Panelists: E. Axelband, RAND Corporation
A. M. Madni, Intelligent Systems Technology, Inc.
C. Dagli, Missouri University of Science and Technology
D. McKinney, Lockheed Martin Space Systems Company

Abstract:

Emergent behaviors exist in biological systems, physical systems and human performance. It is an inherited nature of a System-of-Systems (SoS). SoS displays a global complexity that cannot be adequately managed by hierarchical structures and central control; therefore, traditional systems engineering and management approaches are necessary but insufficient for a SoS.

Little is currently known about constructing an interoperable network of systems and the incorporation of emergent behaviors. The purpose of this panel is to explore the possibilities of developing an architecture model including the emergent behavior.

The challenge is how to understand the initiation mechanisms of the emergent behaviors for a particular system architecture model so that the resident beneficial or harmful emergent behaviors can be enhanced or mitigated with selected changes in the model. Is model-based the only feasible approach to develop the architecture model with emergent behavior? If this is the answer, what kind of modeling methodology? Should it be solely based on agent-based modeling or a combination of SysML and agent-based? Is SysML ready to deal with emergent behavior?

For a non-modeling consideration, can we plan for the beneficial or harmful emergent properties? How do we overcome development friction that is bound to arise when there are complex, independent, overlapping governances, for example, the customer requirements for a SoS evolve over time, etc.? There may only be SoS modeling at the level of the government agency, which actually procures but does not build, and not at the level of contractors who build the next generation of technologies.

Biographies:

Moderator:

John C Hsu

Dr. Hsu is a pioneer in developing and establishing systems engineering process for the Boeing Company. He has worked as technical and project managers, Systems Engineering Integration Lead and Senior Staff at The Boeing Company. With more than thirty (30) years of diversified experience in Systems Engineering, Aerospace Engineering, Mechanical Engineering, Nuclear Engineering, software development and engineering management has made John a significant contributor to the aerospace and nuclear industries.

John is also active in academia. He is an Adjunct Professor teaching graduate/senior level systems engineering at California State University Long Beach and systems engineering certificate program at The University of California Irvine, and a Royal Academy of Engineering Visiting Professor in Systems Engineering in UK.

John is also active in professional societies. He is the Chair of INCOSE Net-Centric Operations (NCO) Working Group, past Region II Representative and President of Los Angeles Chapter. He is the Editor of Systems Engineering Journal and immediate past Chair of AIAA (American Institute of Aeronautics and Astronautics) Systems Engineering Technical Committee and an Associate Fellow.

Dr. Hsu earned his Ph.D. in Mechanical and Aerospace Engineering, M.S. in Nuclear Engineering and M.S. in Mechanical Engineering. He is a registered Professional Engineer.

Panelists:

Elliot Axelband

Dr. Elliot Axelband is an INCOSE Fellow. He is currently a Senior Engineer at RAND Corporation. He is an Associate Dean Emeritus and Consultant of USC School of Engineering. He is also Head of Fellows Search Committee, IEEE Aerospace and Electronic Systems Society, Associate Editor, INCOSE Journal, Member of Systems Engineering Steering Committee, NDIA, etc. His past responsibilities included Member, DoD Scientific Advisory Board, Head, Fellows Committee of US Air Force Center for Systems Engineering Advisory Committee, System Engineering Advisory Board, Systems and Software Productivity Consortium, Member, DoD JSF Independent Review Team (Contracted through IDA), Director, Moller International, Research Professor, USC, Dept. of Electrical Engineering-Systems, Director, USC Graduate Program in System Architecture and Engineering, Associate Dean for Research Development, USC School of Engineering, 35 years with Hughes Electronics (Hughes Aircraft Co.), including: - Vice President, Systems Sector, Vice President, Electro Optical and Data Systems Group and General Manager, Tactical Electro-Optical Systems Division, US Air Force, Scientific Advisory Board, Hughes Senior Executive to USC School of Engineering Board of Counselors. He earned his Ph.D. and MSEE from UCLA and has 50 Professional Publications. Numerous other achievements cannot mention here due to time limitation.

Experience:
- Senior Engineer, RAND Corporation, since ‘94
- Associate Dean Emeritus, Consultant, USC School of Engineering, since ‘03
- Head, Fellows Search Committee, IEEE Aerospace and Electronic Systems Society, since ‘94
- Manager/Part Owner, Legacy Engineering LLC, since ‘98
- Member, DoD JSF Independent Review Team-2 (Contracted through IDA), since ’04-‘06
- Member, US Air Force Center for Systems Engineering Advisory Committee, ‘01-‘05
- System Engineering Advisory Board, Systems and Software Productivity Consortium, ’05-‘06
- Member, DoD JSF Independent Review Team (Contracted through IDA), 3/04 – 5/05
- Director, Moller International, LLC, 2002-2003
- Director, BuySmart LLC, Delaware, Md., 1999 - 2003
- Director, Kollsman Inc., Manchester NH, 1996 - 2001
- Director, USC Graduate Program in System Architecture and Engineering, 1/’94- 6/’03
- Associate Dean for Research Development, USC School of Engineering, 1/’94 – 6/’01
- 35 years with Hughes Electronics (Hughes Aircraft Co.), including:

- Vice President, Systems Sector ($1B in sales, 8000 employees), '93 - '90
- Vice President, Electro Optical and Data Systems Group and General Manager, Tactical Electro-Optical Systems Division
($ 350M in sales, 350 direct and 2000 indirect employees),
‘90 –‘ 86

- Electro-Optical and Data Systems Group, Missile Systems Group, Avionics Systems Group, Space Systems Division, Weapons Systems Group, ‘86 - ’58.

Azad M. Madni

Dr. Azad Madni is elected Fellows of INCOSE, IEEE, SDPS and IETE. He is the founder and CEO of Intelligent Systems Technology, Inc. His research interests are enterprise/systems architecting, system-of-system engineering, modeling and simulation, human performance enhancement, and cognitive systems engineering. He is the recipient of the SBA’s National Tibbetts Award for excellence in research, technology innovation, and transition. In 2000 he received the Blue Chip Enterprise Award for entrepreneurship from Mass Mutual and U.S. Chamber of Commerce. He is a two-time winner of the Developer of the Year Award from the Software Council of Southern California. He is currently serving as the President of the Society of Design and Process Science. He is the Editor-in-Chief of the Journal of Integrated Design and Process Science, a journal devoted to covering transdisciplinary research and education worldwide. Dr. Madni has been a Visiting Industrial Fellow of Caltech’s Jet Propulsion Laboratory in the Space Microelectronics Center. He is listed in the Marquis’ Who’s Who in Science and Engineering, Who’s Who in Industry and Finance, and Who’s Who in America. He received his Ph.D., M.S., and B.S. in Engineering from UCLA.

Experience: Dr. Madni is the CEO and Chief Scientist of Intelligent Systems Technology, Inc., a high technology R&D company specializing in developing solutions and tools for enterprise/systems architecting, system modeling and simulation, and system-of-systems engineering, etc. He has been a Principal Investigator on more than sixty-five R&D projects sponsored by approximately thirty different Federal R&D organizations including DARPA, AFRL, U.S. Army’s, DoE, and NASA, etc. His company is an affiliate of the USC Center for Systems and Software Engineering. He is currently serving as the President of the Society of Design and Process Science and is the Editor-in-Chief of the Journal of Integrated Design and Process Science, a journal devoted to covering transdisciplinary research and education worldwide. Prior to founding ISTI, he was with Perceptronics, Inc. for seventeen years where he served in various senior level executive and scientific positions, culminating in Executive Vice President for R&D and the Chief Technology Officer of the company. Previous to that, he was a senior systems engineer and simulation manager at Rockwell International on NASA’s Space Shuttle program, and a systems engineer at The Ralph M. Parsons Company on Anti-Ballistic Missile Defense programs.

Cihan Dagli

Dr. Cihan H Dagli is an INCOSE Fellow. He is a Professor of Systems Engineering, Computer Engineering and Engineering Management at the Missouri University of Science and Technology.

He received BS and MS degrees in Industrial Engineering from the Middle East Technical University and a Ph.D. in Applied Operations Research in Large Scale Systems Design and Operation from the University of Birmingham, United Kingdom, where from 1976 to 1979 he was a British Council Fellow. His research interests are in the areas of Systems Architecting and Engineering, System of Systems, Smart Engineering System Design, Computational Intelligence: Neural Networks-Fuzzy Logic-Evolutionary Programming.

He has published more than 300 papers in refereed journals and proceedings, 19 edited books and was a Principal or Co-Principal Investigator of research grants of approximately $ 3 million.

Experience: Dr. Cihan H Dagli is a Professor of Systems Engineering, Computer Engineering and Engineering Management at the Missouri University of Science and Technology. He is the founder and the director of the Missouri S&T’s System Engineering graduate program. He is the Area editor for Intelligent Systems of the International Journal of General Systems, published by Taylor and Francis, and Informa Inc. He contributes to INCOSE NCO Net-Centric Operations

Working group. He has consulted with various companies and international organizations including The Boeing Company, AT&T, John Deere, Motorola, U.S. Army, UNIDO, and OECD. He contributes to INCOSE NCO Net-Centric Operations Working group.

Dorothy McKinney

Ms. Dorothy McKinney is an INCOSE Fellow. She works for Lockheed Martin Space Systems Company as a Senior Fellow. Dorothy helps the company to apply best practices and lessons learned from across Lockheed Martin and industry to meet immediate program needs and reduce risks, with special emphasis on software systems engineering challenges.

A graduate of Prescott College with a B.A. and majors in Systems Sciences and English, Dorothy also has a M.S. in Computer Engineering from Stanford University, and an M.B.A. from Pepperdine University.

Professional society participation includes: INCOSE Fellow, AIAA Associate Fellow and IEEE member. Dorothy has been active in INCOSE since 1992, including delivering tutorials to INCOSE chapters, and serving as Technical Chair of INCOSE 2002 and 2007 Symposia. She is past president of the San Francisco Bay Area chapter, serves on the chapter Board of Directors and also has served on the INCOSE Board.

Experience: Ms. Dorothy McKinney works for Lockheed Martin Space Systems Company as a Senior Fellow. Dorothy helps the company to apply best practices and lessons learned from across Lockheed Martin and industry to meet immediate program needs and reduce risks, with special emphasis on software systems engineering challenges.   Dorothy joined a Lockheed Martin heritage company in 1979 as a lead systems engineer. She has held a series of software, systems and engineering management positions in various parts of the corporation, and has been active in multiple program rescues. From 1985 to 1994, Dorothy was also an Adjunct Professor at San Jose State University, teaching graduate courses such as Engineering Management and Software Project Management. Starting in 2002, Dorothy has been teaching a course on Requirements Engineering over the internet for Portland State University. Other industry experience included three years at ARGOSystems (a Boeing subsidiary), and nine years at SRI International (formerly called Stanford Research Institute).

Elliot Axelband- COPING WITH EMERGENCE

Contribution to a proposed INCOSE 2009 panel organized by John C Hsu

Complex adaptive systems and emergence existed long before their terms of art were coined and their consequences appreciated. However, the greater prevalence today of social networks and net centricity and their impact upon the creation and operation of complex adaptive systems accelerates our interest and the need for improving our understanding of emergence, and both utilizing its positive possibilities and suppressing or confining its negative effects.

We can nonetheless learn from the past, and anticipate issues that the future will bring. Both are addressed in terms of confining negative effects: the past in terms of lessons learned, and the future in terms of the new forces attacking information assurance that accompany net centricity that have the power to disrupt complex adaptive systems by inducing negative emergence. These are listed below along with the lessons and challenges they provide. In all cases, modeling and simulation play a strong role in defining and coping with emergence, and in several cases the immaturity of modeling and simulation to address underlying phenomena contribute to the resulting emergence.

1 - Landing on the Moon; creating a system to suppress inherently difficult-to-predict emergent phenomena

2 - Guided missile flight stability; coping with after the fact-experienced emergence

3 - Electro-Optical Image Generation; coping with after the fact-experienced emergence

4 - JSF Weight Explosion; coping with the absence of accurate models and the consequent after the fact-experienced emergence

5 - VH-71 Requirements Miasma; avoiding self-induced emergence

6 - MANET Networks and FCS; coping with emergence in the absence of theory

7 - Information Assured Networks, the need for robustness and multiple lines of defense, and the resultant uncertainty regarding emergen

The lessons vary from example to example, but these examples are real, and I believe the lessons are enduring.

The first example is a classic case where the environment was unknowable. There was, at the time a soft landing on the moon was contemplated, in 1960, insufficient knowledge of the properties of the moon's surface to use to design the lunar descent profile, the landing gear, and the spacecraft it supported. There was a program planned to improve that knowledge by taking images from a hard landing spacecraft and transmitting them back to the earth before it crashed on the moon, but that program failed in its early launches and did not produce any timely data. All then that could be relied upon to estimate the hardness/sponginess and contours of the lunar surface were radar returns, optical images, and geological suppositions, and all experts agreed that these were inadequate for providing accurate lunar surface estimates. However the program succeeded by undertaking a campaign of experimentation that included extensive modeling and simulation, and used these data to design and implement a robust solution.

Examples 2 and 3 represent cases where in hind sight simulation and modeling could have been employed to predict a serious system problem, but were not with the result that the problems were discovered in flight test with pre-production proto-types, and were very expensive to fix as was, of itself, the time spent creating and demonstrating the fix. As a result, simulation and modeling were, at that point, no longer the proper avenues of correction and it was much more expeditious to build alternative fixes and flight test them to find one that was satisfactory.

Example 4 is an illustration of a hard to accept but real situation: there are fields for which accurate models do not exist. Accurate weight prediction for aircraft whose features depart significantly from prior versions is far from guaranteed. In this case, two independent and expert groups made state of the art predictions that were in general agreement, but both were significantly in error. The result was a very serious disruption of a complicated and expensive program as an overweight unacceptable condition emerged. This argues for the early identification of areas for which accurate predictive techniques do not exist, and building early prototypes that allow emergence to be experienced at a time when its consequences can be more easily accommodated.

Example 5 makes the case that you are asking for emergence if you enter system development before you stabilize the requirements, i.e., before you understand what you need to build. This is an old story that has been understood in many fields for many years. What is more intriguing is how, knowing this, the very competent participants could have allowed it to happen. The new story to be explained in the paper is that of the social and political underpinning that caused these veterans to do something that was so basically flawed in retrospect.

Example 6 illustrates that beyond lacking accurate models as in example 4, there are field for which there are no models that predict performance and therefore experimentation is the only way to contain emergence.

And example 7 is similar except that it is not a lack of models, but an inability to predict external forces that, in this case, disrupt networks whose operation is necessary for enterprises to succeed. Emergence is contained in the case be continual expert maintenance that is vigilant at detecting emergent properties, and constructing real time corrective compensation.

A consideration of these 7 examples provides insight into the causes and containment of harmful emergence.

Azad M. Madni – TOWARD ARCHITECTING SYSTEM-OF-SYSTEMS: CHALLENGES AND CRITICAL SUCCESS FACTORS

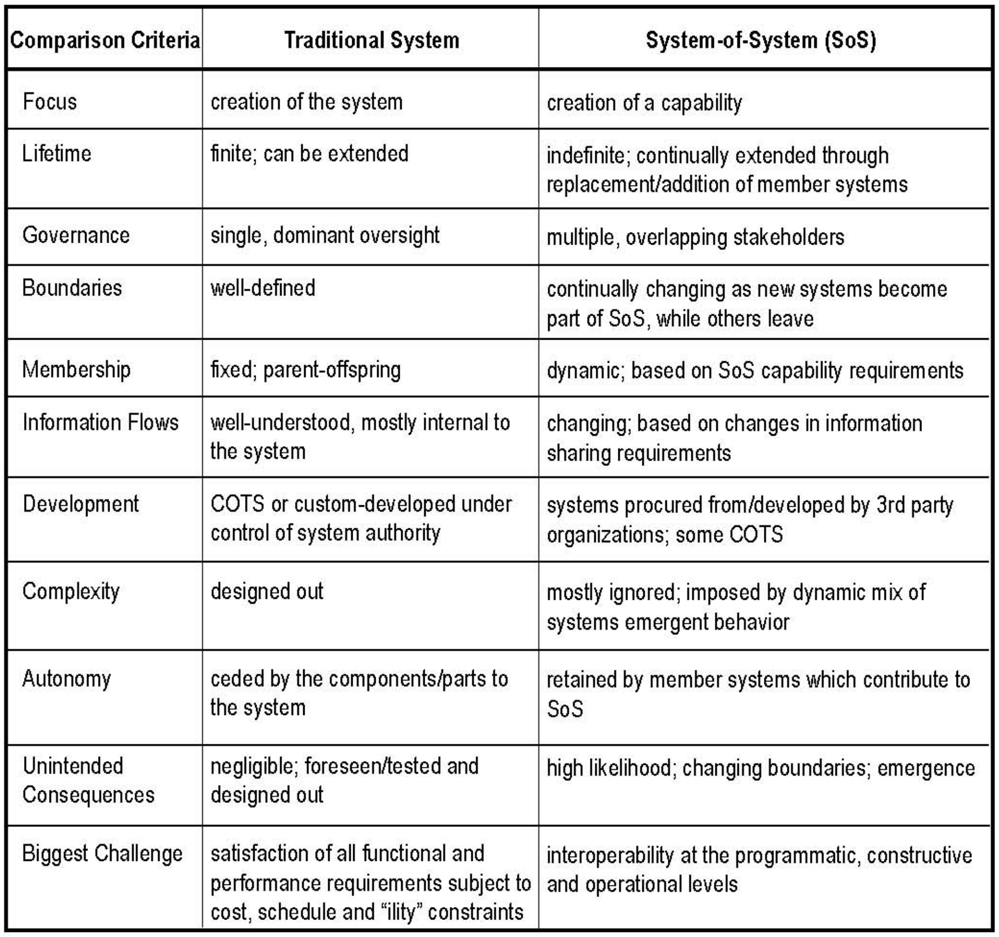
Contribution to a proposed INCOSE 2009 panel organized by John C Hsu

The architecting, design, and management of System-of-Systems (SoS) is complicated by the fact that a SoS exhibits certain unique properties such as changing boundaries, lack of central control, and emergence. Exacerbating the problem is the fact that there is no standard definition of a SoS. Therefore, at the outset, I offer the following definition of a SoS to begin a dialogue:

“A SoS is a complex ensemble of independent systems developed and introduced over different time frames by multiple independent authorities to provide multiple, interdependent capabilities in support of multiple missions. The capability of a SoS typically exceeds the sum of the capabilities of the member systems.”

With this definition, we can begin to distinguish between a traditional system and a SoS.

Table 1. Traditional System Versus SoS [Madni, 2006; Madni, 2007a]



Comparison Criteria	Traditional System	System-of-System (SoS)
Focus	creation of the system	creation of a capability
Lifetime	finite; can be extended	indefinite; continually extended through replacement/addition of member systems
Governance	single, dominant oversight	multiple, overlapping stakeholders
Boundaries	well-defined	continually changing as new systems become part of SoS, while others leave
Membership	fixed; parent-offspring	dynamic; based on SoS capability requirements
Information Flows	well-understood, mostly internal to the system	changing; based on changes in information sharing requirements
Development	COTS or custom-developed under control of system authority	systems procured from/developed by 3rd party organizations; some COTS
Complexity	designed out	mostly ignored; imposed by dynamic mix of systems emergent behavior
Autonomy	ceded by the components/parts to the system	retained by member systems which contribute to SoS
Unintended Consequences	negligible; foreseen/tested and designed out	high likelihood; changing boundaries; emergence
Biggest Challenge	satisfaction of all functional and performance requirements subject to cost, schedule and “ility” constraints	interoperability at the programmatic, constructive and operational levels

From the foregoing definition and comparison of SoS with a traditional system, certain SoS architecting challenges become apparent: (a) overcoming development friction that is bound to arise when there are complex, overlapping governances; (b) getting stakeholders to develop shared interests when, in fact, they are associated with multiple, independent concurrent development with overlapping governances; (c) maintaining coherence at the SoS level when independently planned programs continue to pull in different directions; (d) achieving robustness despite inability to perform critical tradeoffs (remember, complexity renders certain tradeoffs incalculable); and (e) maintaining interoperability in the face of dynamically changing, uncertain information requirements [Salasin and Madni, 2007].

An example of a SoS that exhibits these characteristics is the National Air Transportation System (NATS). The NATS is a geographically-distributed, networked enterprise. In such an enterprise, the different parts of the enterprise tend to optimize their respective objectives which can often be in conflict with each other. The challenge is determining how to maintain global coherence while responding to changes and opportunities at both the global and local levels of the enterprise. In the case of NATS, its architecture is driven by the goals of international commerce. It comprises multiple systems such as the Air Traffic Control System (ATCS), the Airlines, Airport Operations, and, of course, the Consumer Complex. Each system is concerned with maximizing its own objectives. For example: the ATCS is concerned with flight safety and maximizing the use of airspace; the airlines are concerned with maximizing their bottom line; the airport operations are concerned with conserving costs while providing acceptable service; and, the consumers are interested in getting best value (i.e., a combination of cost, timeliness, and experience) from the rest of the enterprise. Since changes can occur in any of these systems and cross-cutting functions, NATS needs to be able to respond to such changes by marshalling the required resources in the physical world and on the net and bringing them to bear at the point of need.

Today, there is little guidance on how to architect a SoS. This is not surprising because architecting a SoS is complicated by several factors. First, SoSs are dynamic entities in that systems are added, modified, or removed as mission requirements change. While it may be possible to address a subset of these changes a priori, several changes need to be handled on the fly. Second, the technological infrastructure needs to change with technological advances and changes in Quality of Service (QoS) requirements. These considerations imply that a SoS architecture needs to not only enable the evolution of the SoS, but also be evolvable itself! Third, we need to be able to assess the potential impact of not completely knowing a priori the architecture of a SoS on the quality attributes and performance of the SoS [Madni, 2008; Madni, 2007b]. In fact, is it even possible to evaluate the quality attributes under such conditions?

Fortunately, it is possible to identify a few critical success factors to get rolling. First, we need to do as much up-front engineering as possible (evolvability is costly and difficult to infuse later). Second, we need to avoid "SoS architecting myopia" by mapping mission capability requirements on to SoS architectural nodes to assess coverage and to identify and fill gaps. Third, we need to focus on the most challenging and the most likely operational scenarios in defining architectural requirements. Fourth, we need to define semantics and models to share best practices. Fifth, we need to employ an iterative process in SoS architecting to continually increase our understanding and reduce risks. Sixth, we need to experiment with "guided" emergence [Madni, 2006] by creating conditions and policies (e.g., incentives/disincentives) that help produce the desired SoS capabilities and behaviors.

An example of "guided emergence" is that of city planning. A city evolves through the collective action of multiple individuals/agents acting locally over time. A city emerges and changes over time through loosely coordinated and regulated action of multiple individuals/agents. An evolving city exhibits coherence without central control through mechanisms that regulate local action [SEI Report, 2006]. Mechanisms include city regulations, communications, distribution and emergency services. These serve as incentives and disincentives. A city is built gradually in parts by people, companies and communities to serve their own purpose. A city grows and thrives based on cultural and economic necessities. A city is always in a state of perpetual change (construction, repairs, demolitions, operations). If one were to substitute the term "SoS" for "city," one can find immediate parallels and develop insights into how an SoS can be architected, designed and managed.

In closing, these thoughts are meant to stimulate a dialogue in the systems engineering community. This is only the beginning of our shared journey in understanding emergent properties of a SoS.

References:

1. Madni, A.M., and Moini, A. "Viewing Enterprises as Systems-of-Systems (SoS): Implications for SoS Research," *Journal of Integrated Design and Process Science*, Vol. 11, No. 2, June 2007a, pp. 3-13.

2. Salasin, J. and Madni, A.M. "Metrics for Service Oriented Architecture (SOA) Systems: What Developers Should Know," *Journal of Integrated Design and Process Science*, Vol. 11, No. 2, pp. 55-71, 2007.
3. Madni, A.M. Architecture Tradeoff Analysis: A Disciplined Approach to Balancing Quality Requirements, *Ground System Architectures Workshop (GSAW)*, Architecture-Centric Evolution (ACE) of Software-Intensive Systems Presentation, March 27, 2007b.
4. Madni, A.M., "Architecture Follies: Common Misconceptions and Erroneous Assumptions," Fellows' Insight, INCOSE *INSIGHT*, Vol. 11, No. 1, pp. 33-34, January 2008.
5. Madni, A.M. System-of-Systems Architecting: Critical Success Factors, *USC-CSSE Convocation, Executive Workshop,* 2006.
6. Madni, A.M. "Agile Systems Architecting: Placing Agility Where it Counts," *Conference on Systems Engineering Research (CSER)*, 2008.
7. SEI's Report on Ultra Large-scale Systems, 2006.

Cihan Dagli – SYSTEMS-OF-SYSTEMS ARCHITECTING

Contribution to a proposed INCOSE 2009 panel organized by John C Hsu

Architecting is the process of structuring the components of a system, their interrelationships and their evolution over time. It is related to the structure properties of a system. Successful architecture development is important as it plays a dominating role in integration of component systems. However, classical system architecting is changing as we are increasingly becoming a networked society. This is true in industry, individuals, and all forms of government. Society is growing increasingly dependent on these networks. It is possible to combine these systems and make them trans-national; it thus provides an opportunity to respond to the dynamically changing needs imposed by global events. Consequently, this creates a need for systems architectures that will be in effect for the duration of the event, possibly necessitating the need to develop new systems architecture for the next mission or event. This fact is important, as it complicates the systems architecting activities. Hence, architecture becomes a dominating but confusing concept in capability development. These systems are generally referred as System of Systems (SoS), which is a collaborative meta-level system structure where independent complex systems are integrated to provide increased functionality and performance capabilities.

The loss of any part of the system will degrade the performance or the capabilities of the whole. They need to evolve in time to accommodate changes in requirements and technology. Hence, systems engineers need to monitor and evolve adapt systems architectures in a timely manner. This eliminates the classical concept that is used in the past, namely, that architectures are static.

These systems evolve by adding components, and as in the case of electrical utilities, creating a potential for hidden robustness, e.g. load sharing across electric utilities, and also giving rise to a potential for cascading failures as well; as characterized by the August 14, 2003 blackout in Northeast U.S. Individual systems within the SoS may be developed to satisfy the peculiar needs of a given group, the information they share being so important that the loss of a single system may deprive other systems of the data needed to achieve even minimal capabilities.

Unfortunately, the current body of knowledge in systems research is not sufficient for effective design and operation of these types of systems. There is a need to push the boundaries of technology and systems engineering and systems architecting research both in industry and research universities to meet the challenges imposed by new demands. There is an increased uncertainty about system requirements coupled with continuous changes in technology and organization structures. Diverse spectrums of missions and operations require the development of system architectures that can adapt and evolve. .

Different complex system of systems can be identified by analyzing the system attributes such as interdependent, independent, distributed, cooperative, competitive, and adaptive. Recent system definitions can be based on these attributes. For example, it is possible to define a Family of Systems (FoS) as a set or arrangements of *independent* systems that can be arranged or interconnected in various ways to provide capabilities. The mix of systems can be tailored to provide desired capabilities, dependent on the situation. Although these systems can be providing useful capabilities independently, in collaboration they can more fully satisfy a more complex and challenging capability. We can also define intelligent enterprise systems in terms of *cooperative*, *competitive* and *adaptive* systems that evolve to respond to changing business conditions. System of systems can be defined in terms of *interdependence* attribute where a set or arrangements of interdependent systems are connected to provide a given capability. While individual systems within the SoS may be developed to satisfy the peculiar needs of a given user group, the information they share is so important that the loss of a single system may deprive other systems of the data need to achieve even minimal capabilities.

Complexity Theory is a beneficial approach to define and understand the identity of a system. It helps in understanding how complex systems are affected from their environments and how a system learns by proposing alternative ways for improvement. It also answers the question that why some good predictions and solutions can be obstructed by dynamic nature of the environment.

1. *Long term planning is impossible:* There are non-linear relationships among components of complex systems. Therefore, long-term planning is impossible. Systems of systems are composed of complex systems and a meta-system behavior cannot be derived by analyzing the behavior of the component sub-systems.
2. *Dramatic change can occur unexpectedly:* Complexity Theory claims that small perturbations can also cause huge changes on the overall system behavior. Changes are inevitable and impact of changes is not always obvious. This property is the reason for cascading failures in System of Systems. Since there is strong interdependency among systems, a small change can cause a chain reaction and result in cascading failures.
3. *Complex systems exhibit patterns and short-term predictability*: Long-term forecasting is impossible but short-term forecasting and describing the behavioral model of systems is possible. Therefore, next time period behavior of systems can be predicted when reasonable specifications of conditions at one time period are given. System of System testing and validation is based on this characteristic. Architecture performance evaluations focus on short term forecasting of system architecture behavior.
4. *Organizations can be turned to be more innovative and adaptive:* Complexity Theory suggests that emergent order and self organization provide a robust solution for organic networks to be successful in competitive and rapidly changing environmental conditions. The evolutionary characteristic of the SoS architecting results in emergent capabilities that individual systems are not capable of achieving. System architects can benefit from this property of by designing SoS components that can self adapt and self organize to changing environmental conditions. In the talk these concepts will be discussed and importance of systems architecting will be emphasized in creating the complex engineering system of this century.

Dorothy McKinney: How to Engineer the Emergent Behavior of A System-of-Systems

Contribution to a proposed INCOSE 2009 panel organized by John C Hsu

Developing, refining and controlling System-of-Systems (SoS) present many challenges. It is instructive to consider how traditional systems engineering and management approaches could be extended and augmented to address the complexities of a System-of-Systems. In considering different approaches, it would be optimal to enable participants in System-of-Systems development and use to change their perspective from the more deterministic view we have traditionally had of system development ("We can decide what the system will do and how it will do it") to a more opportunistic view ("We need to ready ourselves to 'catch the wave' and 'surf the ocean of possibilities' to surmount, or even harness, the 'waves' of emergent behavior we encounter in our attempt to use the System of Systems for our intended purposes"). In other words, we have the dual challenges of:

- adding new approaches and techniques to deal with the complexities of System of System development and
- changing the culture of our organizations to be able to handle the human and organizational complexity and changes required to make effective use of System-of-Systems potential.

The table below identifies some of the non-traditional challenges we face in engineering a System-of-Systems, and dealing with the new demands imposed by emergent behavior. To address these challenges, we can try to control events long enough to get a System-of-Systems working to meet formal functional requirements and quality of service targets. This is the "get it right the first time" approach. Or we can try to get a System-of-Systems functioning, and then refine its behavior to move closer to desired functional capabilities and quality of service. This is the "get it working first, then try to evolve it in desired directions" approach.

The latter approach may be more practical when resources are very limited and/or requirements are changing more quickly than System-of-Systems capabilities can be completed and delivered. If we take this latter refinement approach, we need significantly different mechanisms for accounting to stakeholders than the conventional plan-versus-actual progress reporting. Opportunistic refinement of System-of-Systems capabilities probably requires much more of a continuing "sale to stakeholders" than does a conventional plan-and-perform-to-plan approach. This kind of "salesmanship" has not historically been a key requirement for effective systems engineering, and may pose real personal growth challenges for many individuals in our profession. The table below does not attempt to address how the needed new mechanisms could be "sold" to stakeholder communities as credible and reliable, but that is clearly part of the work we have to do to be able to more effectively engineering System-of-Systems in the future.

Challenges	**Insights Needed**	**Leverage Needs**
Re-composing stakeholder groups over time	Identifying affected parties	Multiple mechanisms for engaging stakeholders
Balancing and re-balancing competing system priorities given changing stakeholder needs over time	Opportunities as well as risks posed by emergent behavior	Marketplace mechanisms to allow dynamic trade-offs in both SoS usage and incremental development of new/changed capabilities
Shifting locus of control	Costs/benefits (in term of SoS performance) of ceding control to constituent systems versus exerting control through interfaces or constraints	How interfaces can be used to constraint and control constituent systems
Information control	When to change between "need to share" and "need to know" for different elements of information and data	Mechanisms to handle changing information control and sharing over time – for the same and different info/data elements
Diffuse control over inclusion of different constituent systems in the SoS	When users have achieved practical inclusion of new system elements in the SoS (or changed SoS boundaries)	Mechanisms for enabling users of the System-of-Systems to understand which capabilities have which levels of maturity, credibility and V&V

One of the toughest decisions for systems engineers working on System-of-Systems refinement is where to focus the biggest amount of effort and energy:

- on refining the System-of-Systems to eliminate or compensate for the negative emergent behaviors discovered
- on working with stakeholders to enable them to benefit from the positive emergent behaviors discovered, and re-prioritize the changes to be made in the next refinement of the System-of-Systems

This decision is analogous to the logger's choice between sawing down the next tree, and stopping to sharpen his saw. When we are implementing a plan-then-execute effort in an attempt to "get it right the first time" it is fairly clear when we should focus on "sharpening the saw":

- ➢ When development is proceeding well enough that there is "spare time" to "sharpen the saw" or
- ➢ When development falls far enough behind schedule that it becomes painfully apparent that the saw must be sharpened to make adequate future progress.

But when we are implementing a refinement-towards-desired-capabilities approach, it will be much more difficult for all of the stakeholders to tell when it is time to work with stakeholders to help them benefit from positive emergent behaviors and re-prioritize for the next refinement ("sharpen the saw"). In the world of Agile Software development, this decision is solved by choosing a standard interval (the duration of an agile scrum, typically a timeframe such 4 or 6 weeks), and holding each period of refinement to that interval. The first few days of each interval are used to re-prioritize, and the remainder of the interval is used to design and implement as many of the top priority refinements as possible.

It is not clear that such an approach is feasible for refinement of a System-of-Systems, since there are so many more stakeholders, and they are typically diverse enough that getting them to speak with one voice cannot be accomplished within a few days. So we, as a profession, will need to invent the systems engineering System-of-Systems equivalent to the Agile Software process if we want to use this refinement-towards-desired-capabilities approach.

If we use a combination of SysML modeling and agent-based modeling to improve out insight into the System-of-Systems, we can probably use these models to capture the characteristics of emergent behavior as it is discovered. We may also be able to use these models to show various stakeholders the results of different possible trade-offs in System-of-Systems development. Only trial and error are likely to show whether use of these models will also enable us to reach consensus among stakeholders about when it is time to press ahead and refine the System-of-Systems to deal with emergent behaviors, and when it is time to focus on improving stakeholder use of current system capabilities and re-prioritization of the next refinements needed.