System *of* Systems (SoS)

Our understanding of SoS, just like our understanding of system, must transcend domains.

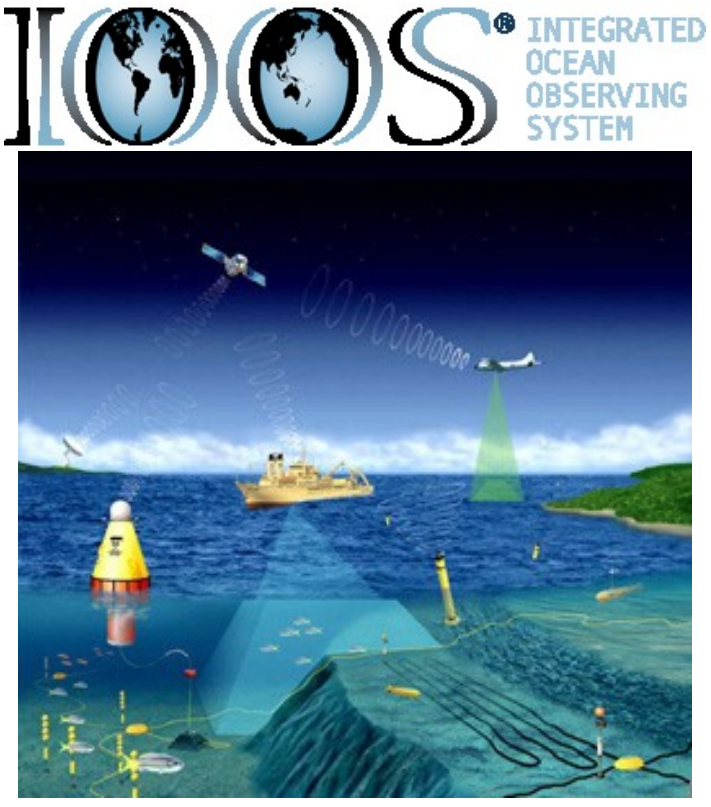
"SoS is a collection of entities and their interrelationships gathered together to form a whole greater than the sum of the parts. Thus, our attention is directed to parts (the entities), relationships (between the parts) and the whole itself which we understand in some sense to be better than the collection of parts and relationships"

"A SoS is a system, but *of* matters, and it forms an antithetical stance to the gathering together for the type of system that subordinates its parts and relationships to meet its purpose."

"In thinking about a SoS we are faced from the outset with two inescapable verities.

1. Some parts of the envisaged SoS, constituent systems, already exist; these are commonly known as legacy systems.
2. There is some new system in view, to which these legacy systems will contribute, and in envisioning this new system there is both a commonality of purposes – of existing systems' purposes and of the to-be SoS purpose; but there is also a differentiation of purpose; in other words the existing purposes can help bring about the new SoS purpose simply because of this commonality."

How to recognize or to realize a System of Systems (SoS) ?



Autonomy

The reason a system exists is to be free to pursue its purpose.
A vehicle has a need and in part the brake serves that need. It serves no purpose of its own. The notion of autonomy appears to have been lost in the systems approach, which is why some systems are known as such when in fact they may in truth be merely parts.

Belonging

The envisioneer of the SoS foresees it partly by seeing how existing systems can play a part in a new adventure; parts but yet systems.
Perhaps these systems need to change, but not beyond recognition. Each will form new relationships, with other autonomous systems. Each will have to be persuaded of the value of all this – to change, to render service, and to collaborate with other systems.

The existence of the SoS will enhance the value of the system's purpose, exalt the role of the system, whose belonging makes achievement of the supra purpose more likely and more effective. But that belonging does mean partness for the autonomous system. This autonomous legacy system now exhibits both partness and wholeness.

Connectivity

We are faced with the need to create connectivity amongst the legacy systems and possibly additions of new systems to the SoS.
The systems themselves have an important role to play in determining the connectivity they wish to form to one another, and to new additions. This is consistent with constituent system autonomy and its choosing to belong, composing as it were from the ground up the rules of engagement by constituent systems in the unfolding design of the SoS, always towards the fulfillment of its mission, the supra purpose for constituent systems.
This determination is not confined to *a priori* design, requiring the degree of prescience normally associated with systems engineering. *Au contraire*, it calls for a dynamic determination of connectivity, with interfaces and links forming and vanishing as the need arises.

Diversity

For Life to survive for a long time, it must conserve or generate a great deal of species diversity. The larger the variety of actions available to a control system, the larger the variety of perturbations it is able to compensate.
The other side of this coin is the principle of parsimony. This states that one should not make more assumptions than the minimum needed. By doing that, developing a model will become much easier, and there is less chance of introducing inconsistencies, ambiguities and redundancies.
Does diversity apply to the parts, relationships or the whole? Ex: neurons, synapses or a brain.
SoS should, out of necessity, be incredibly diverse in its capability as a system compared to the rather limited functionality of a constituent system, limited by design.
It seems that there is a fundamental distinction to be made between requirements-driven design for a conventional system based on its defined scope, and a capabilities-based SoS that must exhibit a huge variety of functions, on an as-needed basis, in order to respond to rampant uncertainty, persistent surprise, and disruptive innovation.

Emergence

How does emergence give us differentiation between a system and a SoS?
In a system, emergence is deliberately and intentionally designed in. What's more unintended consequences, i.e., unpleasant or painful emergent behavior are tested out, as far as possible. Instead, a SoS has emergent capability designed into it by virtue of the other factors: preservation of constituent systems autonomy, choosing to belong, enriched connectivity, and commitment to diversity of SoS manifestations and behavior. The challenge for the SoS designer is to know, or learn how, as the SoS progresses through its series of stable states, to create a climate in which emergence can flourish, and an agility to quickly detect and destroy unintended behaviors, much like the human body deals with unwanted invasions.

The distinction between a system and SoS lies in the meaning and significance of 'gathering together', teasingly hidden in the meaning of of. What the two have in common is being gathered together which is why it is proper to refer to a SoS as a system. However a SoS is much more because its parts, acting as autonomous systems, forming their own connections and rejoicing in their diversity, lead to enhanced emergence, something that fulfills capability demands that set a SoS apart.

Element	System	System of Systems
Autonomy	Autonomy is ceded by parts in order to grant autonomy to the system	Autonomy is exercised by constituent systems in order to fulfill the purpose of the SoS
Belonging	Parts are akin to family members; they did not choose themselves but came from parents. Belonging of parts is in their nature.	Constituent systems choose to belong on a cost/benefits basis; also in order to cause greater fulfillment of their own purposes, and because of belief in the SoS supra purpose.
Connectivity	Prescient design, along with parts, with high connectivity hidden in elements, and minimum connectivity among major subsystems.	Dynamically supplied by constituent systems with every possibility of myriad connections between constituent systems, possibly via a net-centric architecture, to enhance SoS capability.
Diversity	Managed i.e. reduced or minimized by modular hierarchy; parts' diversity encapsulated to create a known discrete module whose nature is to project simplicity into the next level of the hierarchy	Increased diversity in SoS capability achieved by released autonomy, committed belonging, and open connectivity
Emergence	Foreseen, both good and bad behavior, and designed in or tested out as appropriate	Enhanced by deliberately not being foreseen, though its crucial importance is, and by creating an emergence capability climate, that will support early detection and elimination of bad behaviors.

Service Orientation and Systems of Systems

In an ideal service-oriented SoS environment, constituent systems would register their capabilities in a service-oriented, standardized capabilities registry. SoS architects, integrators, and developers would query this registry to find capabilities that meet desired functional and quality attribute requirements. Each capability in the registry would contain associated metadata to be able to use the capability.

Mark Maier (1998) identifies five SoS characteristics that are useful in distinguishing such systems from very large and complex but monolithic systems:
• Operational independence of the constituent systems;
• Managerial independence of constituent systems;
• Evolutionary development;
• Emergent behavior; and
• Geographic distribution.

Maier also defines four types of SoS based on their management structure:
• Directed, constituent systems are integrated and built to fulfill specific purposes;
• Acknowledged, SoS have recognized objectives, a designated manager, and resources;
• Collaborative, constituent systems voluntarily agree to fulfill central purposes; and
• Virtual, have no central authority or centrally agreed purpose.

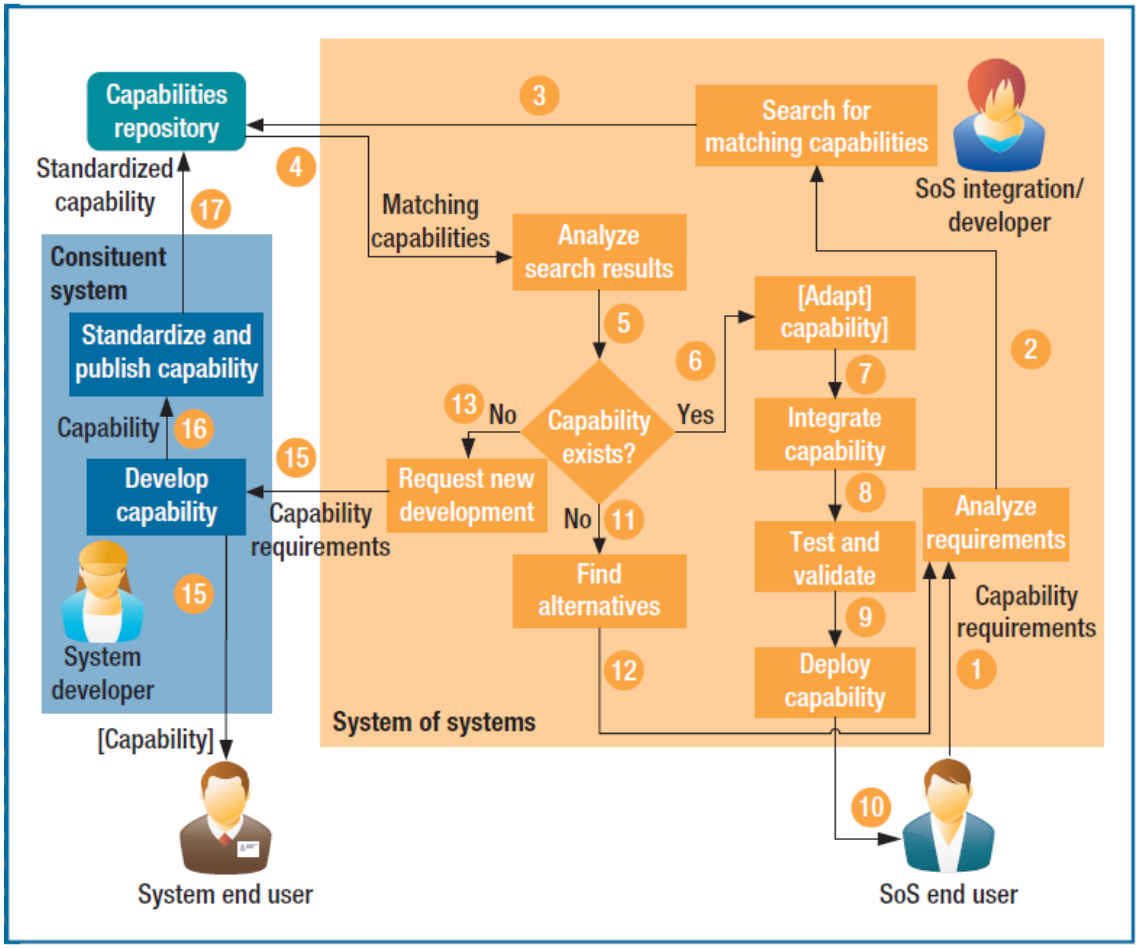


Figure: **Lewis et al., 2011**

1. An SoS end user has a requirement for new capabilities.

2. The SoS developer/integrator analyzes the new capability requirements.
3. The SoS developer/integrator searches the capabilities repository for matching capabilities.
4. The SoS developer/integrator analyzes the returned matching capabilities.
5. The capability requirements might match an existing capability (to a certain extent) or not. If there isn't a match, the SoS developer/integrator might have the option of finding other alternatives or requesting new development.

If the capability exists in the capabilities repository

6. If there isn't a 100 percent match between requirements and capabilities, the SoS developer/integrator might adapt the existing capability by developing code to deal with the mismatches.
7. The SoS developer/integrator integrates the new capability into the SoS.
8. The SoS developer/integrator tests and validates the new capability and its effect on the SoS.
9. The SoS developer/integrator deploys the new capability.
10. The new capability is delivered to the SoS end user.

If the capability doesn't exist in the capabilities repository and the SoS developer/integrator can't afford to wait for development of new capabilities

11. The SoS developer/integrator looks for alternatives either by repeating the search with different criteria or by presenting the SoS end user available capabilities and seeing if the end user is willing to relax capability requirements in exchange for faster delivery of capabilities.
12. Process starts again at Step 2.

If the capability doesn't exist in the capabilities repository, and the SoS developer /integrator can wait for development of new capabilities (this is possible for directed and acknowledged SoS environments but might not be possible in collaborative and virtual ones)

13. The SoS developer/integrator sends requirements to the system developer.
14. The system developer develops capabilities that match the requirements.
15. The system developer might decide to make the new capability available to system end users.
16. The system developer standardizes and publishes the new capability in the capabilities repository. Depending on the type of relationship between the SoS developer/integrator and the system developer, the SoS developer/integrator could require certain tests and compliance requirements before publishing the capability.
17. After the new capability is published, it's now available to the SoS developer/integrator who would restart the process at Step 3.

MAIER M. W., 1998. "*Architecting Principles for Systems-of-Systems*", Systems Engineering, 1(4): pp.267-284, Wiley Online Library.

LEWIS G., MORRIS E., SIMANTA S., SMITH D., 2011. "*Service Orientation and Systems of Systems*". IEEE Software, 28(1): pp. 58-63, IEEE Computer Society.