Paper #85

Systems of Systems and Emergent System Context

Nirav B. Shah **Dr. Donna H. Rhodes** **Prof. Daniel E. Hastings**
Massachusetts Institute of Technology
77 Massachusetts Ave., 41-205
Cambridge, MA 02139
nbshah@mit.edu rhodes@mit.edu hastings@mit.edu

Abstract

One of the key issues facing architects of systems of systems (SOS) is the emergent nature of the context of the SOS. In the spirit of Herbert Simon, context here means all the stakeholders, objects, processes and other external effects that create the environment for goal attainment of a system. As SOS are systems, they too have contexts. Understanding the relationship between the contexts of the systems that constitute the SOS and SOS's own context is a key step in building the architecture of a SOS. This paper will address SOS context from several perspectives. First, context is defined and examples given to explain its relevance to system architecture. This is followed by a discussion of how the context of an SOS is not simply the union of the contexts of its constituents, but rather emerges as a result of interactions that define the SOS. A key issue that arises from this view of SOS context is that the temporary nature of many SOS forces designers of constituent systems to be particularly adept at simultaneously operating in/planning for current, future and legacy SOS contexts in which they may be participating. The paper concludes with a discussion of different strategies for how constituent designers might handle this situation with recommendations and examples drawn from historical commercial and defense SOS.

Introduction

Since the mid 90's there has been a growing interest in how systems come together to form systems of systems (SOS). These temporary coalitions of independently operated and managed systems can meet unforeseen needs in a timely and cost effective fashion. For example, during the first gulf war a SOS was formed between Patriot missile batteries and Defense Support Program (DSP) satellites. The satellites, meant for strategic missions, provided the Patriots' operators with "over the horizon" early warning of medium-range missile launches prior to the inbound missiles being visible to the Patriot radar (Cunningham 1991). Combining disparate systems was not always so successful. During the lead up to the same conflict, electronic coordination could not be fully established with naval forces resulting in air tasking order needing to be flown from Riyadh out to aircraft carriers (Zinn 2004). There is much attention focused today on deliberately architecting systems so that they support broader objectives of a SOS.

The first step is to identify those characteristics that distinguish the SOS engineering challenge from traditional systems engineering. Several authors have attempted such classification (e.g. Eisner 1991, Maier 1998, Keating 2003, Gideon 2005). Though there are differences in how each author defines the SOS problem, three key characteristics stand out in their definitions:

1. SOS are systems
2. SOS are composed of other systems that are value producing in their own right
3. SOS constituents have some sense of independence after being assembled into the SOS

The first of these points indicates that SOS are systems and as such possesses the characteristics of systems traditionally cited such as stakeholders who build and derive value from their use, an environment in which they operate and, in the case of open systems, interfaces. The second point emphasizes that the constituents systems of a SOS are systems. They too possess the system characteristics outlined above. In joining the SOS the constituents do not lose their identity as systems and remain recognizable. The first two points alone, however, do not differentiate SOS as a class of systems requiring special attention – given these points, SOS are simply systems that happen to be composed of other systems. There are many systems that are composed of other systems. For example, a factory may be composed of several manufacturing cells, a mainframe of several processors, etc. The third point leads to the taxonomic distinction between SOS and the more general problem of systems engineering.

Maier (1998) identifies two types of independence experienced by constituent systems: operational independence (if removed from the SOS, a constituent could operate once again as an independent system) and managerial independence (even when incorporated into the SOS, a constituent continues to deliver value independently). SOS that exhibit these two independence criteria are termed 'collaborative SOS' and, as Maier argues, require a different architectural approach than traditional systems. For example, in the Patrior-DSP case cited earlier, full command and control authority over the DSP satellites could not be handed over to the Patriot operators, as the DSP satellites still needed to perform other missions in addition to helping the Patriots.

This paper focuses on collaborative SOS. The reader should take SOS to mean collaborative SOS unless otherwise stated. One key consequence of this independence is the complex nature of the environment or context in which a SOS operates. SOS being systems, exist within an environment. Their constituents are also systems and so also have a local environment. Given managerial independence, understanding the relationship between the SOS environment and the local environments of the constituents is important to creating successful SOS. This paper seeks to explore that relationship and discuss architectural issues that arise from it.

Defining Context

While the term environment or context is often used in the systems literature, there is disagreement on a precise definition. For example, some consider stakeholders inside the system boundary, while other consider them influencers on the system and therefore part of the context. Some guidance in this regard can be found in some of the early references. Ackoff (1971) defines the environment of a system as:

"The *environment* of a system is a set of elements and their relevant properties, which elements are not part of the system but a change in any of which can produce a change in the state of the system. Thus a system's environment consists of all variables [that] can affect its state. External elements which effect irrelevant properties of a system are not part of its environment"

Ackoff further notes that the environment of a system can be subjective. Since its definition is directly related to the system problem being analyzed the same entity may be part of the system in one case and part of the environment in another.

Like Ackoff, Simon (1996) defines the 'outer environment' of a system as the 'mold'

into which the successful system fits, i.e. the environment, while not part of the systems, through its form and influence specifies the conditions or boundaries within which the system achieves its objectives. Included are the objectives themselves as they must be satisfied for the system to exist – e.g. a plane must be able to fly for it to be plane. By tying the definition of a system's environment to the objectives that its promulgators are trying to meet, Simon removes some of the subjectivity of Ackoff's definition. A key point of contention in the literature is if stakeholders are part of the system or the context. Since this issue often comes down to the semantics of how exactly an author defines the term 'system', there is no universally agreed upon choice. For this paper, stakeholders are considered part of the system only if they directly participate in the value creating action of the system. If they only specify objectives or are affected by the output of the system, they are part of the context. Some authorities (e.g. INCOSE 2004) define the term environment in a narrower sense, i.e. the natural environment, regulatory environment or software environment, etc. The sense of environment being developed here encompasses all those environments in so far as they relevant to goal attainment for the system under study. Taken individually, however, they only provide one piece of the environment as Simon defines it. To prevent confusion this paper will use the term *context* for environment in its broader sense. Based upon Simon and Ackoff, two definitions of context are provided below. When considering a system already in existence, the analytic definition applies, while design of new or modification of existing systems calls for the architectural definition. Note that the two definitions specify the same entities as being part of the context.

Context (analytic): The external entities and conditions that need to be taken into account in order to understand system behavior.

Context (architectural): The mold to which the architecture must conform. It is a result of the objectives and constraints that the designer is trying to satisfy.

An example of system context: A communication satellite's context consists of both physical laws and requirements imposed by stakeholders. On the physical side, there is the space environment and the realities of signal transmission physics, while stakeholders impose requirements on communication capability and regulatory constraints on spectrum usage. If an understanding of the entire product lifecycle is desired, then the acquisition environment (e.g. schedule, budget, etc.) comes into play. All these issues collectively form the context that the spacecraft (program) must fit to be successful.

Since context defines the conditions for system success, understanding it is a key task for the system architect. However, as demonstrated in the example above, context is often complex and multi-faceted.

When one considers SOS, this challenge becomes all the more difficult.

Context of SOS

As specified in the definition, SOS are systems and, as such, have a context. This context specifies, to use Simon's definition, the external conditions and influences that must be dealt with for goal attainment. From the analytic perspective, determining the context of a SOS is a similar problem to determining context of any complex system. Architecture frameworks can be helpful in managing the data collection process and much literature exists on their use (see, for example, Bernus 2003). A more difficult problem arises when one considers context from a design perspective.

As an analogy to the SOS case, to see how context develops when two systems interacts, consider Figure 1.

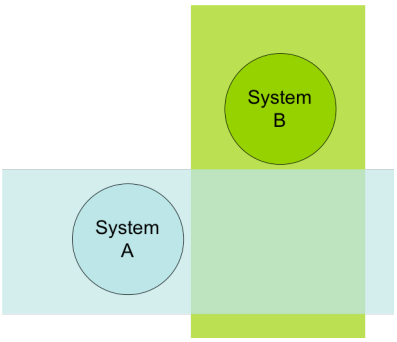

Figure 1: Two Systems operating separately, but sharing context

Two systems, A and B, are generically represented as circles. They are each situated in their respective context indicated by the blue and green rectangles respectively. They are not interacting directly, but do share some context. For example, system A could be the rental property listing service on the website Craigslist[1] and system B, Google's map service[2]. In this example, the shared context includes a common technical foundation, i.e. the Internet and WWW, and some common stakeholders, e.g. users of both services.

These users soon found out that these services can be used together – find a property to ones liking on Craigslist and then map the address in Google Maps to get a sense of neighborhood. The two services being independent however, did not facilitate such collaborative use. One user, Paul Rademacher, noticed that there were tools available to make a Google map overlayed with rental opportunities (Rademacher 2005). This new entity, HousingMaps (www.housingmaps.com), is a SOS as defined above. See Figure 2 for a sample screenshot of listings for downtown Boston. The constituent systems remain quite independent in both an operational and a managerial sense. They are so independent in fact that neither Craigslist nor Google participated directly in Rademacher's endeavor. At the time of development there wasn't even an API available for Google Maps, Rademacher had to rely on underlying web technologies (Google 2005, later, spurred on by effort such as Rademacher's, an API was released). Since HousingMaps is a system, it has a context. This new context is represented in Figure 3.

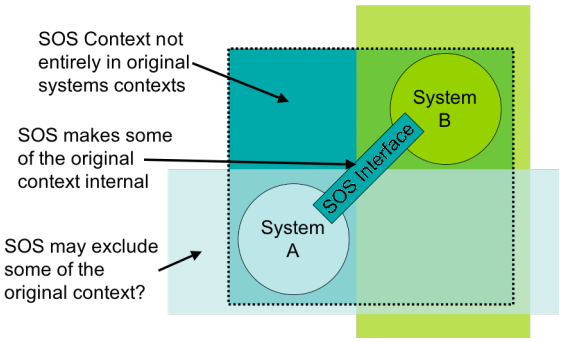

Figure 3: Context of a SOS

Unlike what one might naively be led to believe, the context of the SOS is neither the union nor the intersection of the contexts of its

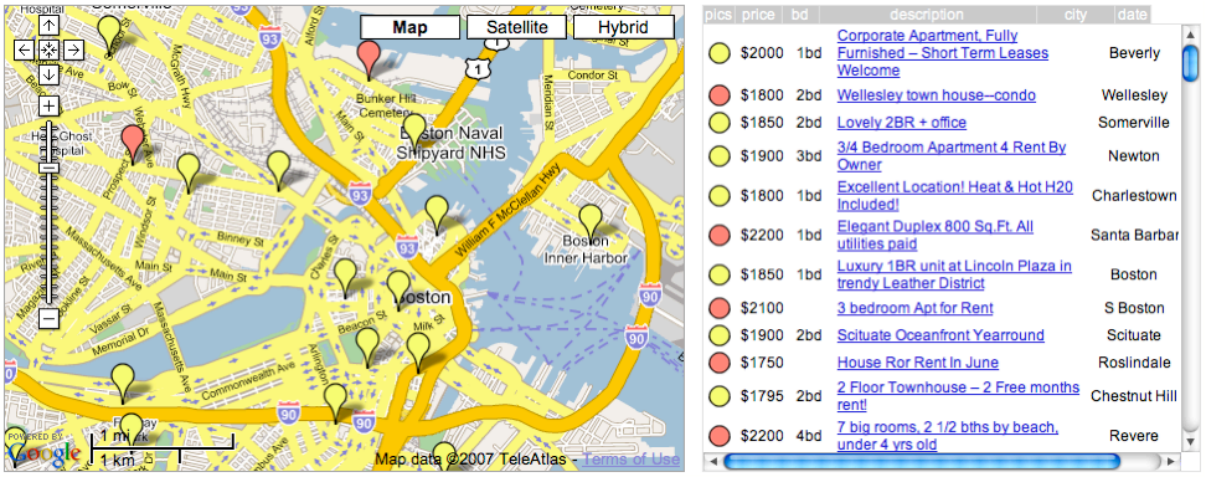

Figure 2: HousingMaps showing Craigslist rental listings on Google Maps

constituents. Rather, it includes entities from both constituents' contexts and also introduces new element/constraints that do not apply when considering the constituents in isolation. Applying this diagram to the HousingMaps example, one first notices that the website acts as an interface between the two constituent systems[3]. In doing so a new context is created. The SOS context inherit the Internet architecture and related constraints from its constituents, and also has new constraints levied by the technologies used by Rademacher in developing the website. For example, the scripting engine used to develop HousingMaps is not relevant when considering the constituents in isolation, but become very important with looking at the SOS. This would lie in the teal area in Figure 3. Conversely, some portion of each of the constituent's context is not relevant to the SOS. For example, Craigslist must abide by government fair housing regulations – such regulations do not impact HousingMaps. Given the example of HousingMaps, it seems that the context of the SOS can be complex and is not easily determined by looking at the contexts of the constituents alone.

Unfortunately this situation is not limited to the example given above. Rather it will be typical for SOS because they are systems. As systems they exhibit emergence in the sense of Simon (1996), i.e. they exhibit behavior that is distinct from the union of the behavior of the constituents. Simply bringing together several systems does not form a SOS. One must have them interact to produce value that could not be obtained without said interaction to form a *system* and thus a SOS. Since this new value producing behavior did not exist prior to forming the SOS it follows that any context components that are related only to it are irrelevant when considering the constituents in isolation. One of those contextual components is the need or requirement for the SOS behavior. This need is not part of the constituents' context. Therefore, the SOS's context contains at least this element that does not exist in the constituents' contexts and will probably contain other elements that arise from the SOS need.

The situation becomes even more complex when one allows for constituents to participate in multiple SOS. Both the constituents in the HousingMaps example participate in other SOS. In fact meeting the needs of HousingMaps is a minor objective of those constituents. Contextual constraints may arise from these other SOS engagement and impact the SOS under study. The quick rise of sites such as HousingMaps placed great strain upon Google's infrastructure and they were forced to put limits on information provided to these external sites in order to maintain quality of service (Google 2006a). This constraint on geo-location services arose from neither HousingMaps's nor Google's actions alone, but was a consequence of the overall popularity of the services Google Maps provided. Figure 4 extends the previous context diagrams to include this effect:

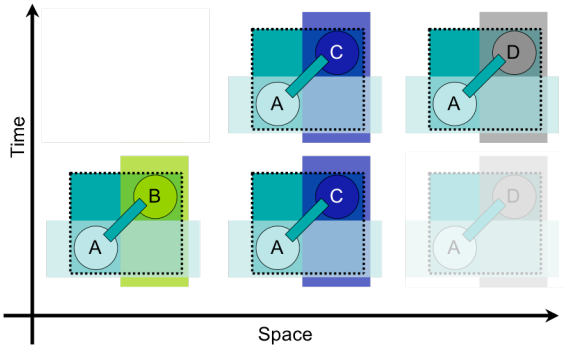

Figure 4: Possible cases for participation of a constituent in multiple SOS

Figure 4 shows the many possible SOS in which system 'A' could be participating. The horizontal axis indicates that at the same time, system A may be participating in multiple SOS (the AB and AC SOS). In addition, there

[3] One can also interpret this as a three entity SOS: Craigslist + Google Maps + the HousingMaps backend and UI. In this view, Google and Craigslist are interfacing with HousingMaps and not with each other.

is potential for future participation is the as yet unrealized AD SOS. The architects of each of these SOS may need to aware of one or more of system A's other SOS since they may impact their own architecture. For example, if the same system A (as opposed to copy of A) is simultaneously participating AB and AC, the pressure on systems A's resources from the AC interaction on AB ability to make use of system A's resources is an important part of the context of AB. Contextual issues may also spring up when one considers SOS evolution over time. In Figure 4, the AD SOS may be hobbled by the legacy interface A need to maintain with C.

In summary, determining the context of a SOS is a non-trivial exercise for the SOS architect and will require consideration of issues beyond those identified in the context of the constituents. The context of the SOS emerges as a result of the interaction between the constituents. Other SOS in which the constituents participate may also have an impact. Given this challenge, what then is the SOS architect to do? Fully answering that question will require more research into the mechanisms of how the SOS context emerges, however historical examples can provide some guidance to the architect.

Creating context: SOS Integration

Given the widespread interest in SOS there have been surprisingly few in-depth case studies of SOS. However, a few good case studies do exist and can provide the architect with guidance on how to manage the complex context of the SOS. One particular study of note is Krygiel's (1999) examination of the Army's Task Force XXI Advanced War Fighting Experiment (AWE). The AWE was an attempt by the Army to determine how a networked brigade would function. The intent was to bring together the latest in information technology with restructured tactics and then gauge their effectiveness through a series of war games. The integrated system did have the characteristics Maier proscribes for a SOS, ensuring that the individual systems could still operate independently.

Much work went into planning the integration of the many dozen systems involved in the AWE. Architectures were developed, protocols were documented and constituent level test plans were implemented. Despite all these efforts, the initial attempt at integrating the SOS went quite poorly. The many systems simply did not interoperate correctly given demands being place on them. Systems that worked in isolated test failed to scale when tried in the full exercise.

Despite the best intentions of the developers of the individual systems, integration simply did not work. They had demonstrated that it is very difficult to, *a priori*, understand the shared context formed when an SOS is assembled without actually forming the SOS. Their difficulties were compounded by the fact that several of the systems being integrated were new or were operating in new ways. This limited their knowledge of constituent contexts and thereby made understanding and staying within the limits of the SOS all the more challenging. To resolve this impasse, the Task Force XXI team gave authority to their 'Central Technical Support facility' to manage the integration of the SOS. The leader of the CTSF was given the authority to arbitrate between the needs of the various constituent system stakeholders with the intent of producing a cohesive whole. The result was transition from a haphazard integration effort in which the SOS context was constantly shifting as systems tried to integrate, to an orderly integration process driven by SOS needs. This authority also reduced the length of decision making cycles and allowed rapid spiral development of the SOS capability.

The Task Force XXI story represents one approach to managing the complexity of SOS context – strong central authority to arbitrate and prioritize system evolution and spiral development to allow for rapid learning and elu-

cidation of the SOS context. The downside of this approach, however, is that the SOS assembled is the one conceived by the integrating authority that may or may not be the best solution.

The integration experience summarized above is quite different from that which produced HousingMaps. In the HousingMaps case, the constituents were not even aware of there participation until integration was well on its way. More importantly, the existence of robust interfaces or 'hooks' in the constituents' architecture meant that the constituents did initially not *need* to know about the SOS integration. However, once the SOS hampered the constituents' ability to meet other objectives, corrective action became necessary. Had Rademacher taken an approach more like Task Force XXI, such difficulty could have been avoided. Resource usage could have negotiated upfront instead of dealt with once the crisis was upon the constituents. In the end finding the right heuristics for managing the emergence of SOS context will require further study. Google has moved to model more like Task Force XXI in the next version of their API terms of use requesting that if "your site gets more than 500,000 page views per day, we ask that you talk to us before you launch so that we can prepare in advance to handle your traffic." (Google 2006b) Such a system ensures that smaller sites can grow without the need of Google's attention, yet allows Google to work proactively with larger sites.

Summary

This paper has defined and developed the concept of system context when applied to SOS. The context of an SOS is complex entity the results from the interaction between the SOS constituents and may not be readily understood by looking at just those constituents. Environments in which constituents are forming multiple SOS further complicate the situation. While more research is needed to understand how SOS context comes to be, the literature seems to indicate that centralized coordination may be a good way to keep integration complexity at a manageable level. Such control, however, will limit innovation and prevent SOS like HousingMaps from developing. Finding the right balance between control of the integration process and openness to allow for creativity of the constituents is a major challenge for SOS architects.

References

Ackoff, R.L., "Towards a System of Systems Concepts", *Management Science*, v.17:11, July 1971

Bernus, P., Nemes, L., Schmidt, G., *Handbook on Enterprise Architecture*, Springer, 2003

Cunningham, J.H., "The Role of Satellites in the Gulf War", *The Journal of Practical Applications in Space*, v.2:3, Spring 1991

Eisner, H, Marciniak, J, McMillan, R, "Computer-Aided System of Systems (S2) Engineering", *IEEE SMC*, 1991

INCOSE, Glossary in *System Engineering Handbook*, Version 2a, June 2004

Maier, M.W., "Architecting principles for systems-of-systems", *Systems Engineering*, v.1:4, 1998

Keating, C., Rogers, R., Unal, R., Dryer, D., Sousa-Poza, A., Safford, R., Peterson, W., Rabadi, G., "System of Systems Engineering", *Engineering Management Journal*, v.15:3, Sep. 2003

Krygiel, A., *Behind the Wizard's Curtain*, CCRP, 1999, see chapters 3-4 for a description description of the case and chapter 5 for lessons learned from the integration experience.

Gideon, J.M., Dagli, C.H., Miller, A., "Taxonomy of Systems-of-Systems", *CSER*, 2005

Google, "Maps/Craigslist mashup", *Google Code Blog,* 2005, http://google-code-featured.blogspot.com/2005/04/mapscraigslist-mashup.html

Google, “Google Maps API Terms of Use”, 2006a
http://www.google.com/apis/maps/terms.html
Google, “Google Maps API Version 2”, 2006b
http://googlemapsapi.blogspot.com/2006/04/google-maps-api-version-2.html
Rademacher, P., *HousingMaps*, 2005
http://www.housingmaps.com/about.html
Simon, H.A., *The Sciences of the Artificial*, 3rd Edition, MIT Press, 1996
Zinn, A., *The Use of Integrated Architectures to Support Agent Based Simulation*, Master’s thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2004.

Biography

Nirav Shah is a graduate student at MIT pursuing a Ph.D in Aeronautics and Astronautics. A member of the V-STARS research cluster and supported by MIT's Lean Aerospace Initiative, his research focuses on architecture of systems of systems. In particular, he is studying the impact of coupling in the physical, functional, and organizational spaces on system evolution. His work experiences include positions at Los Alamos National Laboratory and with Booz Allen Hamilton. Nirav received an S.B. (2001) degree and an S.M. (2004), both in Aeronautics and Astronautics, from MIT.

Donna Rhodes holds a Ph.D. in Systems Science from the T.J. Watson School of Engineering at SUNY Binghamton. Her research interests are focused on systems engineering, systems management, and enterprise architecting. Dr. Rhodes has 20 years of experience in the aerospace, defense systems, systems integration, and commercial product industries. Prior to joining MIT, she held senior level management positions at IBM Federal Systems, Lockheed Martin, and Lucent Technologies in the areas of systems engineering and enterprise transformation. Dr. Rhodes is a Past-President and Fellow of the International Council on Systems Engineering (INCOSE).

Daniel Hastings is a Professor of Aeronautics and Astronautics and Engineering Systems at MIT. Dr. Hastings has taught courses and seminars in plasma physics, rocket propulsion, advanced space power and propulsion systems, aerospace policy, technology and policy, and space systems engineering. He served as chief scientist to the U.S. Air Force from 1997 to 1999 and as director of MIT’s Engineering Systems Division from 2004 to 2005. He is a member of the National Science Board, the International Academy of Astronautics, the Applied Physics Lab Science and Technology Advisory Panel, and the Air Force Scientific Advisory Board.