The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems

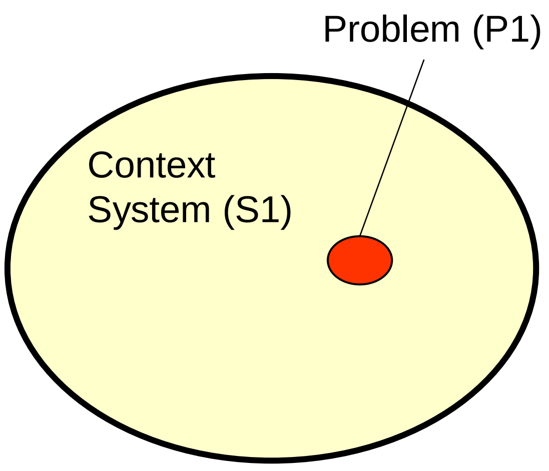
James N Martin
The Aerospace Corporation
M/S CH1-410, 15049 Conference Center Drive, Chantilly, VA 20151
James.Martin@incose.org

Abstract. There are seven different systems that must be acknowledged and understood by those who purport to do systems engineering. The main system to be engineered is the Intervention System that will be designed to solve a real or perceived problem. The Intervention System will be placed in a Context System and must be developed and deployed using a Realization System. The Intervention, when installed in the Context, becomes the Deployed System which is often different in substantial ways from the original intent of the Intervention. This Deployed System will interact with Collaborating Systems to accomplish its own functions. A Sustainment System provides services and materials to keep the Deployed System operational. Finally, there are one or more Competing Systems that may also solve the original problem and will compete for resources with your Deployed System. All seven systems must be properly reckoned with when engineering a system.

Introduction

The Analogy. "*Shichinin No Samurai*," the 1954 film classic directed by Akira Kursawa, is an apt illustration for the plight of the systems engineer. The Seven Samurai were the mighty warriors who became the seven national heroes of a small town. A poor village under attack by bandits recruits seven unemployed samurai to help them defend themselves. The notion of the "seven samurai" described in this paper illustrates the seven systems that are underemployed in the classical practice of systems engineering. When these 7 Samurai are employed with proper consideration and enthusiasm, they will become the seven national heroes of your small town (the system development project).

The Context System. Let us examine the first of seven systems—the Context System (S1). Context is "the set of facts or circumstances that surround a situation or event." (WordWeb) It is the set of "interrelated conditions in which something exists or occurs." (Webster's New Collegiate) Context also goes by the name "Environment" which means the "circumstances, objects, or conditions by which one is surrounded." (ibid) Context originally meant the "weaving together of words" and leads us to the more common connotation of the term: "the parts of a discourse that surround a word or passage and can throw light on its meaning."



Linguistics? Now why would we systems engineers bother with the linguistic aspects of the

Presented at the 2004 Symposium of the International Council on Systems Engineering (INCOSE)
Received "Best Paper" Award

word Context? Precisely because systems engineering is very much about finding the correct words to describe the problem to be solved by the engineering solutions we intend to create. In the words of Jack Ring, the systems engineer's job is to "language the project." [Ring et al 2000]

The Problem System. The Context is where the Problem P1 resides. Aspects of Context can be, and often must be, reverse engineered to discover the constituents of the problem's environment. We must understand the relationships of the constituents to each other and to the problem itself. Is there something in the context that is causing the problem? If we solve the "problem" but do not address the cause(s), will the problem merely evolve into something more dreadful? Is the initial statement of the problem really the problem or merely a symptom of the real problem?

Object Oriented Thinking. Using the object oriented approach, the relevant items in the environment can be identified as "objects." The objects are identified as either types or instances. These object types (or classes) and instances can be depicted using a Class Diagram. An older, but still useful, technique for this contextual analysis is the ERA approach. This involves identifying the relevant Entities, the Relationships between those entities, and the Attributes of each entity or relationship. Below is an illustration of an ERA diagram developed during the context analysis phase of a project to develop the Observing System Architecture for NOAA (National Oceanic and Atmospheric Administration). [Martin 2003]

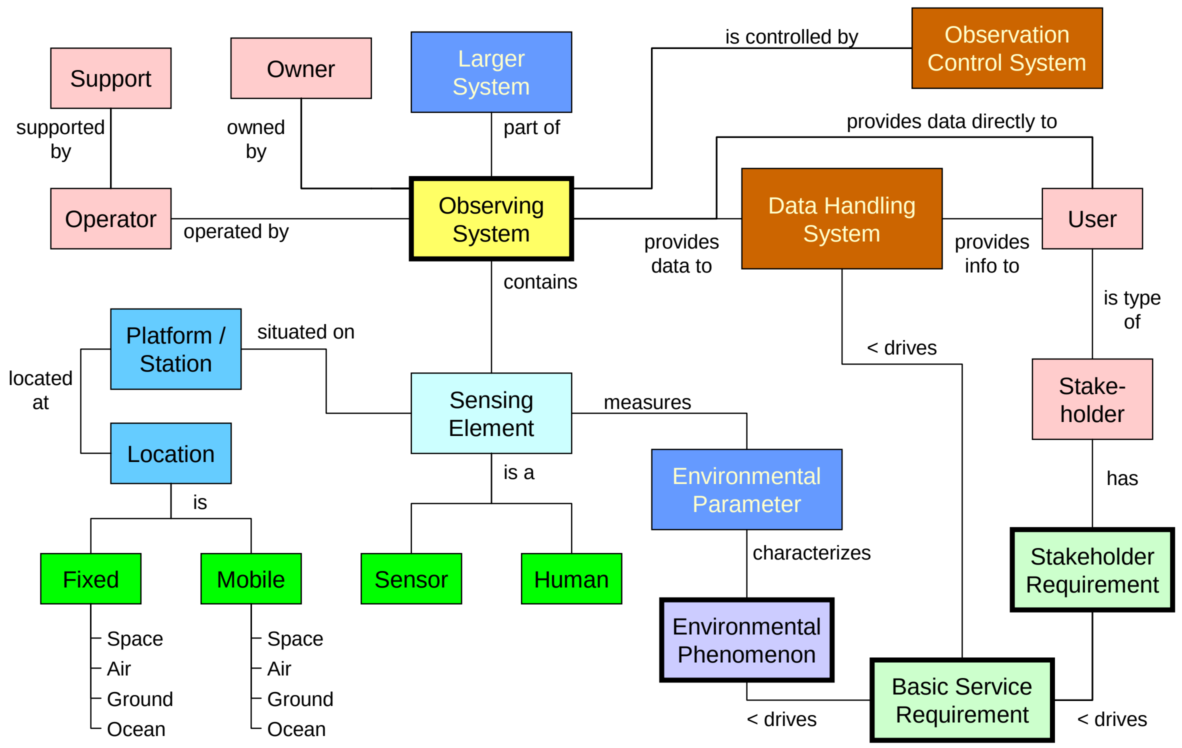


Figure 1. ERA diagram for the NOAA Observing System Architecture

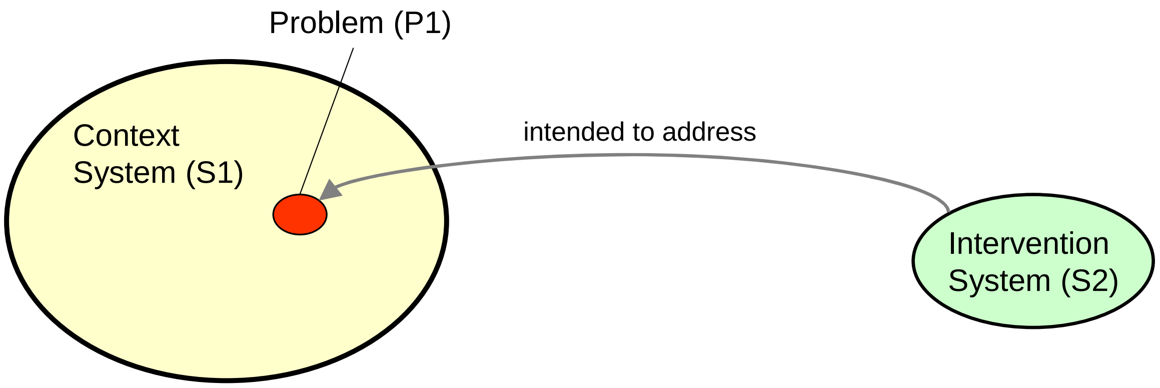
The ERA diagram above does not illustrate the attributes, so it is more correct to call this an ER diagram. Sometimes the attributes of each entity are listed inside each entity box. In the case above only the entity type names are shown. Whether you use a Class Diagram or Entity Relationship Diagram, you are really defining the "scope" of the problem to be solved.

Metamodeling. The basic structure of any problem can be captured in a "metamodel." Often the metamodel you need to use for your problem of interest is already captured in your favorite tool or methodology (eg, UML or IDEF0). The problem P1 for NOAA was to identify the deficiencies and excess capacities of the 100 different observing system types owned or operated by NOAA. The ERA diagram above is a depiction of the metamodel for the NOAA problem situation. A good description of the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a metamodel are given at [metamodel.com]:

> A meta-model is an explicit model of the constructs and rules needed to build specific models within a domain of interest. A valid meta-model is an ontology, but not all ontologies are modeled explicitly as meta-models. A meta-model can be viewed from three different perspectives:
>
> 1. as a set of building blocks and rules used to build models
> 2. as a *model of a domain of interest*, and [emphasis added]
> 3. as an instance of another model.
>
> When comparing meta-models to ontologies, we are talking about meta-models as models (perspective 2).
>
> Note: Meta-modeling as a domain of interest can have its own ontology. For example, the CDIF Family of Standards, which contains the CDIF Meta-meta-model along with rules for modeling and extensibility and transfer format, is such an ontology. When modelers use a modeling tool to construct models, they are making a commitment to use the ontology implemented in the modeling tool. This model making ontology is usually called a meta-model, with "model making" as its domain of interest.

The Intervention System

Now we must look for a solution to the problem. Let us call this intended "solution" the Intervention System (S2). The Intervention System is intended to address the Problem P1. It is the system to be engineered using the systems engineering process, methods, and tools. This is the central focus for the development project that is established to be a profitable venture for systems development companies. But to ensure that the so-called "requirements" for this system are valid and complete, full and proper consideration must be given to all seven "samurai." These samurai will bring misery to all if left loose to roam at will across the countryside.



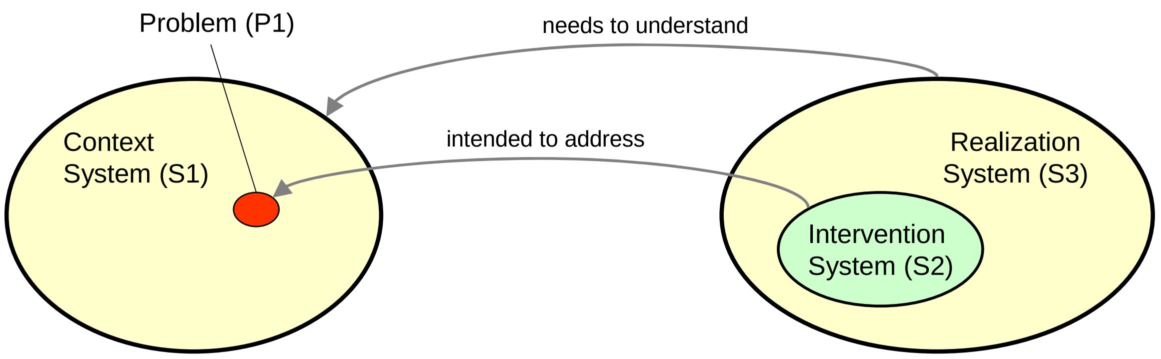
Preventing the Undesirable. Intervention is "*action affecting another's affairs:* an action undertaken in order to change what is happening or might happen in another's affairs, especially in order to prevent something undesirable" (dictionaries.com) Intervention can be seen as a sort of perturbation of the Context, as a form of engagement with the evils of the world. Intervention has two types: intermediation and mediation. Mediation is a form of negotiation to "resolve differences conducted by some impartial party." (WordWeb) Intermediation is acting "between parties with a view to reconciling differences." (ibid)

Achieving Reconciliation. What are these differences to be reconciled? There will be people in the Context that would like the situation to be different, better somehow. The systems engineer should devise an Intervention System that settles the differences between the way things are now, the "as-is" situation, and the desired state of affairs after intervention, the "should-be" situation. It is important to recognize that the systems engineer must be an unprejudiced, third party to this situation. When a systems engineer is "involved" in the situation, it is difficult to be impartial and just when deciding how best to "solve" the problem.

Systems Architecting. The conceptual nature of the Intervention System is often understood through the efforts of "architecting." [Maier and Rechtin 2000] Bear in mind that S2 may include 'mod kits' to the Problem System and the Context System. A depiction of the system architecture is created by development of an architectural model which uses the metamodel's foundational building blocks—the element types and structures discovered in the Context during analysis of S1. Architecture can be thought of as "an arrangement of feature and function that accomplishes some objective." [Ring 2001]

The Realization System

For the Intervention System to come about, it must be brought into being by a Realization System (S3). The Realization System consists of all the resources to be applied in causing the Intervention System to be fully conceived, developed, produced, tested, and deployed.



The Realization System will consist of a wide variety of things, some tangible and some not:

(a) people & organizations
(b) facilities & equipment
(c) materials & supplies
(d) services & utilities
(e) processes & methods
(f) tools & techniques
(g) policies & procedures
(h) data & information
(i) knowledge & wisdom

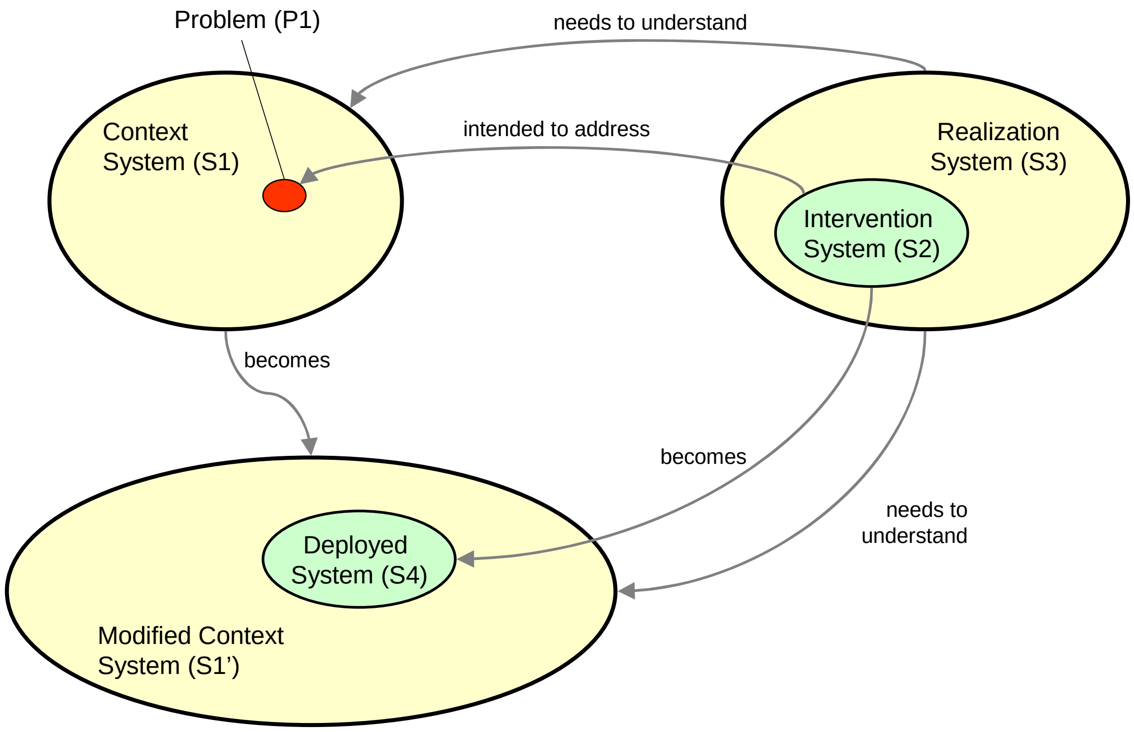
(j) and so on

All of these things interact in complex ways to bring about a solution to the real or perceived problem. The Realization System needs to "understand" the Context and the Problem contained therein. How can this be? How can a system have understanding? Well, people and organizations have understanding and they are an intimate part of the Realization System. Understanding is also captured in policies and procedures, and in knowledge and wisdom. This is the reason that knowledge management has become so important for better execution of the systems engineering process. A good way to model and understand the Realization System is through knowledge modeling. [Lillehagen et al. 2003]

Enterprise Architecture. Often this Realization System is known as an Enterprise. An Enterprise is a purposeful or industrious undertaking (especially one that requires effort or boldness). It usually involves many organizations that contribute their resources to the "owning" organization of that enterprise. The organizational resources can be either tangible (eg, funding and people) or intangible (eg, goodwill and enthusiasm). Enterprise architecting is a relatively new field of endeavor but is gaining popularity as the complexity of current ventures (and adventures) becomes more recognized. A good description of enterprise modeling can be found in [Vernadat 1996].

The Deployed System

Even though we have the best of intentions, the system we design, develop, and build will often morph into something else once it is transitioned to its final destination. This Deployed System (S4) is intended to be the same as S2, but variability often occurs due to malicious intent, inadvertent errors, performance degradation, deployment pressures, interaction between the new system and its environment (S1/S4 coupling), and so on.



Modified Context. The new system will often change the original Context into a Modified

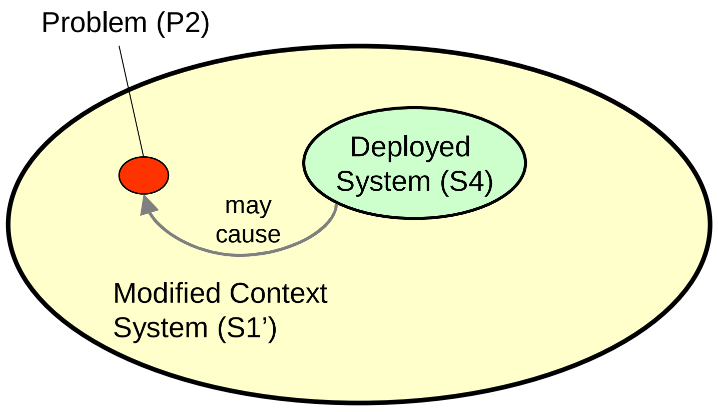
Context (S1') in ways that are sometimes beneficial, but more often than we would like this change is to the detriment of those we were trying to help. Furthermore, several years may have passed since the original analysis of the Context was conducted and when the Intervention System was ready to deploy. The world changes without asking our permission. The original "customer" has often moved on. The people we interviewed to assess the situation may have already solved their problem through other means.

Unintended Consequences. Notice that the Realization System also needs to understand the Modified Context. The systems engineers must be cognizant of how their proposed solution might change the original Context, and perhaps even become worse than the original problematic situation. Never forget the Law of Unintended Consequences. [Norton]

> The law of unintended consequences, often cited but rarely defined, is that actions of people—and especially of government—always have effects that are unanticipated or "unintended." Economists and other social scientists have heeded its power for centuries; for just as long, politicians and popular opinion have largely ignored it.
>
> The concept of unintended consequences is one of the building blocks of economics. Adam Smith's "invisible hand," the most famous metaphor in social science, is an example of a positive unintended consequence. Smith maintained that each individual, seeking only his own gain, "is led by an invisible hand to promote an end which was no part of his intention," that end being the public interest. "It is not from the benevolence of the butcher, or the baker, that we expect our dinner," Smith wrote, "but from regard to their own self interest."

This Law clearly applies in the economic domain, but is equally applicable, if not more so, in the domain of systems engineering. We must heed this Law if we are to be successful in engineering systems that are appropriate for Context Systems that are complex and adaptive. [Holland 1998] The best situation is where the proposed solution is adaptive to changes in the environment to compensate for environmental changes. [Holland 1995]

A New Problem. Not only has the original Context been modified, but our newly deployed system often causes a new Problem (P2). More work for the unemployed, you say. Yes, but your company might go out of business due to litigation or bankruptcy before you have a chance to rid the streets of the homeless.

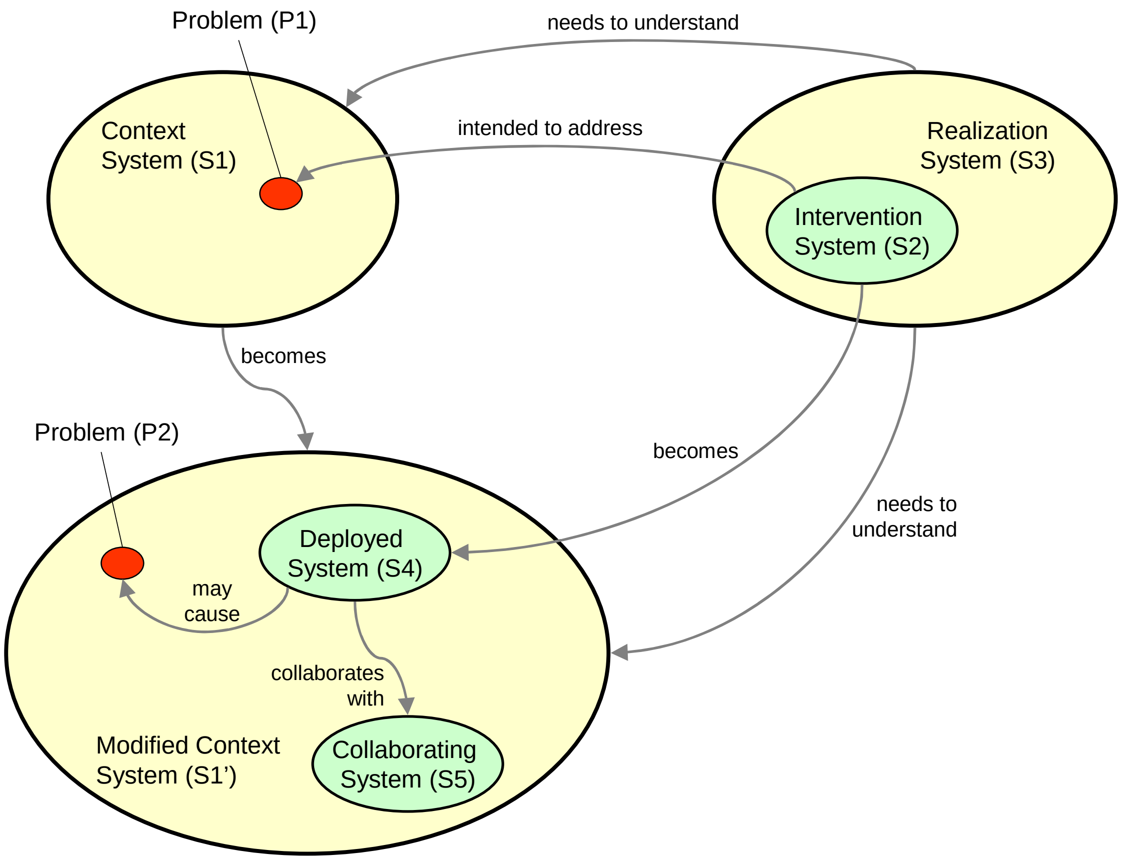


One reason for the change in Context is that it contains people. People are highly complex and adaptive. Therefore you can expect your system "solution" to be used improperly, controverted, damaged (sometimes even unintentionally), bypassed, and so on. People are good at finding things to do with your system that were not part of your original intent. Hence, be forewarned—

your solutions can sometimes cause more problems than they solve. As system development progresses, it is essential to be cognizant of mutations in the Context and adjust the development goals accordingly.

The Collaborating System

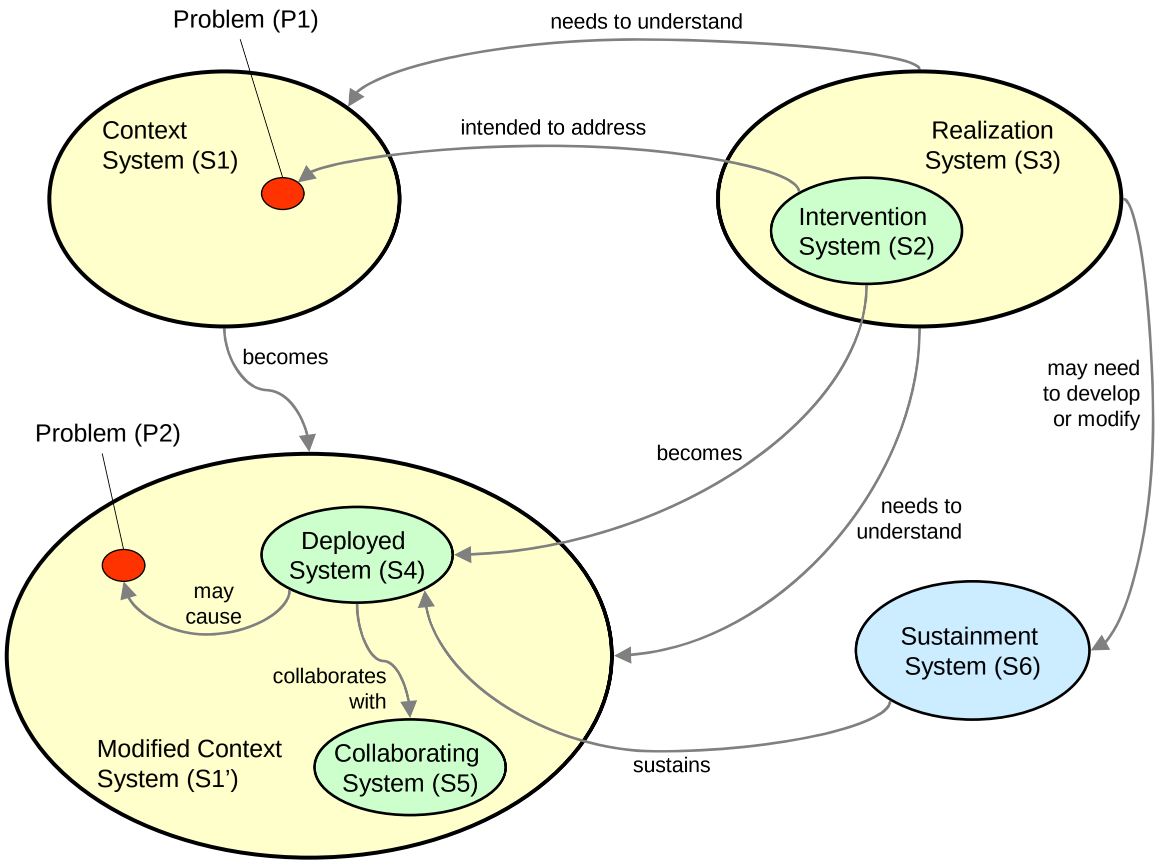
When we designed our Intervention System, we may have realized that we had access to certain resources that could solve only part of the problem. What to do? We made agreements with industry partners, or we decided to make our system modular so that it fits into someone else's platform. We may have decided to incorporate standard interfaces so our system will work with other systems in a synergistic fashion. This can be a win-win situation. But there are times when this can backfire due to "emergent" properties that are undesirable. Why, our system worked with that other system in our integration lab—why doesn't it work out in the field?



Unintended Collaborations. Don't forget that the Collaborating Systems also interact with the Context (the Modified Context, really) and these changes in the environment could affect how your system interacts with its intended collaborators. And then there are the Collaborating Systems that you never intended to interact with. Someone else can come along and "plug in" to your system. This could be great since it could make your system much more valuable to the customer, more indispensable. Or this could be bad since this new Collaborating System could be performing some of the functions of your systems (those that are perhaps not quite as efficient or effective as they could be).

The Sustainment System

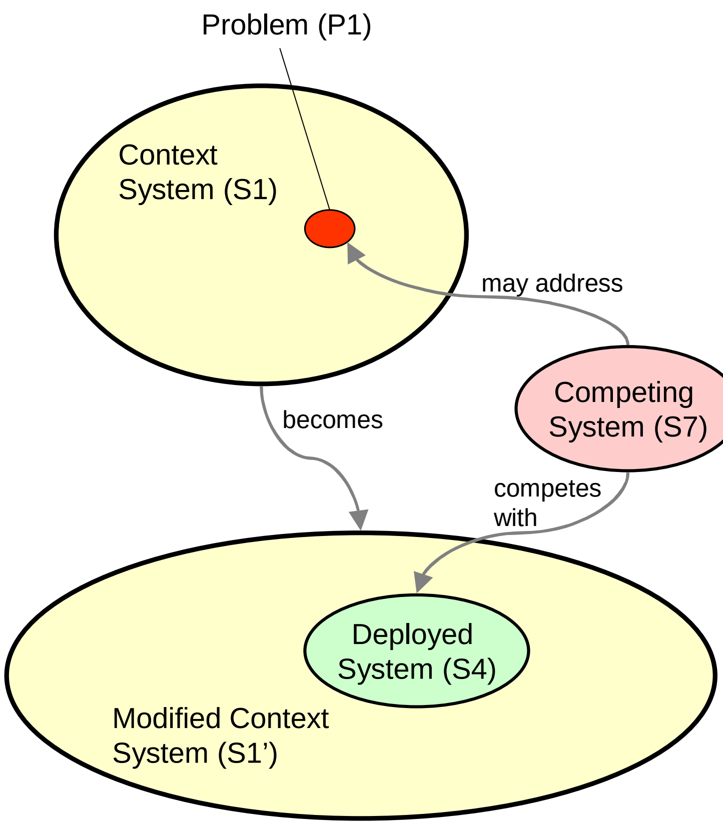
Now we come to the Sustainment System (S6) that provides the necessities and support such as fuel, energy, spare parts, training, customer hotline, maintenance, waste removal, refurbishment, retirement, and so on. It is quite important for the Intervention System to take into account the capabilities and limitations of the Sustainment System. In many cases, the Realization System may need to modify (or even develop parts of) the Sustainment System.



The Sustainment System is often thought to be under the purview of the logistics support engineer. Logistics is a relatively mature discipline that can address most of the concerns related to sustainment. (See [Blanchard 1998] for a good summary of logistics support tools and techniques.) But the systems engineering team needs to work with logistics early in the game to ensure these issues are addressed before "unsupportable" features and functions are captured in the solution concept. The sustainment costs are typically ten to twenty times the cost of development. Therefore, it is worthwhile to spend considerable effort in understanding the sustainment issues before proceeding too far along the path of system development.

The Competing System

Now if life were not complicated enough already as a systems engineer, we must also deal with the Competing System(s) (S7) that may also address all or parts of the original Problem P1. It may provide similar or identical features and functions as your proposed System solution. The Competing System also competes for resources used by the Deployed System. Furthermore, you need to avoid being blindsided by concurrent developments or advances in technologies that might render the Deployed System obsolete.



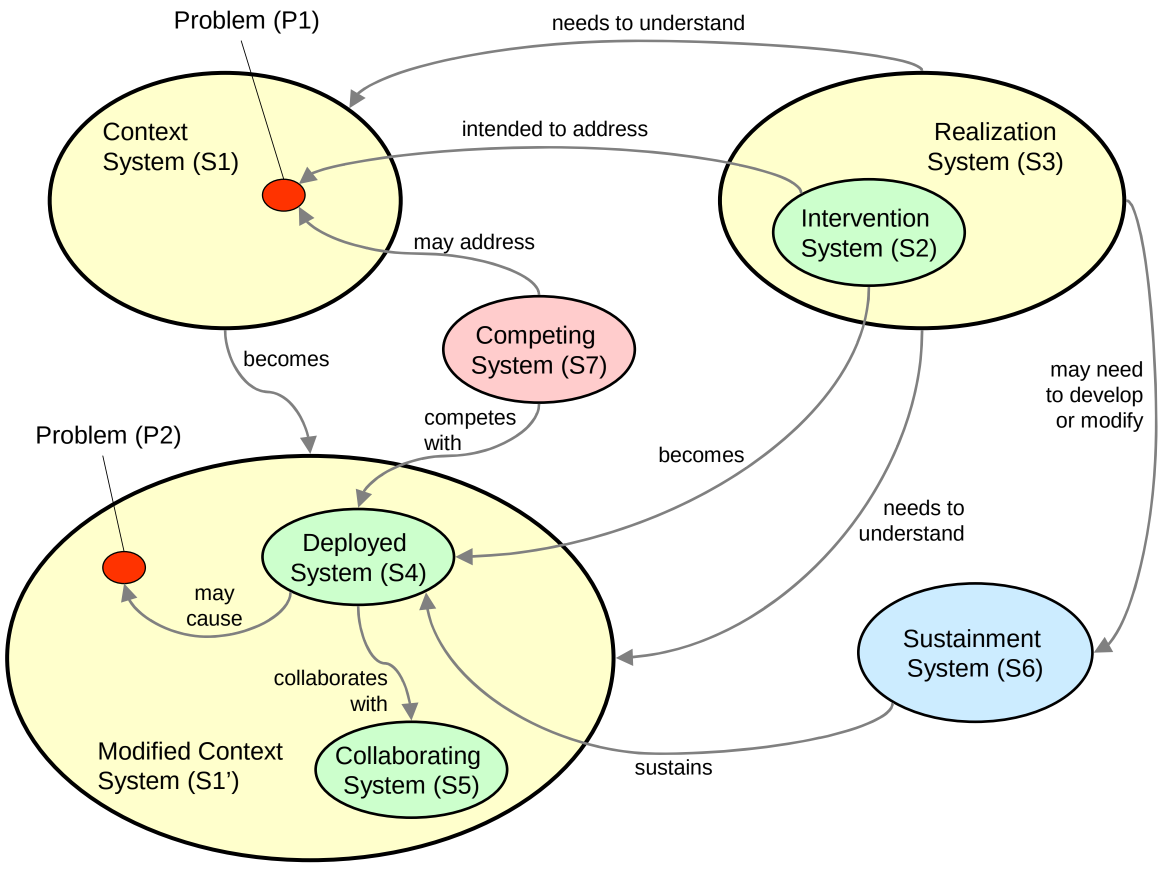
Summary

We can now summarize the interactions between these seven samurai systems:

(1) Context System (S1) contains a Problem (P1)
(2) Intervention System (S2) is intended to address P1
(3) Realization System (S3) brings S2 into being
(4) S2 is a constituent of S3
(5) S3 needs to understand S1
(6) S3 needs to understand the Modified Context System (S1')
(7) S3 may need to develop or modify the Sustainment System (S6)
(8) Intervention System (S2) becomes Deployed System (S4)
(9) S1 becomes the Modified Context System (S1')
(10) S4 is contained in S1'

(11) S4 collaborates with one or more Collaborating Systems (S5)
(12) S4 is sustained by Sustainment System (S6)
(13) S4 may cause new Problem (P2)
(14) Competing System(s) (S7) may address the original Problem (P1)
(15) S7 competes with S4 for resources and for the attention of users and operators

Holistic Systems Thinking. By understanding these fifteen interactions, we now have a better chance of understanding the "whole picture." We need to model all aspects of the entire situation to ensure our system solution is indeed the best way to solve the problem. The essential holistic view is illustrated below.



Is it any wonder why Systems Engineering (SE) is so difficult? For decades we have not explicitly acknowledged nor understood the various systems that must be addressed when engineering a solution for a complex, adaptive situation.

This new paradigm of the "Seven Samurai" must be considered in the application of SE process, methods, tools, and standards if we expect SE to address the increasingly complex problems of the 21st century.

References

Blanchard, Benjamin S., *Logistics Engineering and Management*, Prentice Hall, 1998.

Holland, John N., *Hidden Order: How Adaptation Builds Complexity*, Addison Wesley, Reading MA, 1995.

Holland, John N., *Emergence: From Chaos to Order*, Addison Wesley, Reading MA, 1998.

Lillehagen, Frank and Solheim, Helge G., "The foundations of AKM technology." Concurrent Engineering Conference, 2003.

Maier, Mark W. and Rechtin, Eberhardt, *The Art of Systems Architecting*. CRC Press, 2000.

Martin, James, "On the Use of Knowledge Modeling Tools and Techniques to Characterize the NOAA Observing System Architecture." *Proceedings of the INCOSE Symposium*, 2003.

Norton, Rob, "Unintended Consequences," The Library of Economics and Liberty. http://www.econlib.org/library/Enc/UnintendedConsequences.html

Ring, Jack, and A. Wayne Wymore, "Concept of Operations (ConOps) of a Systems Engineering Education Community," *Proceedings of the INCOSE Symposium*, 2000.

Ring, Jack, "Discovering the Architecture for Product X," *Proceedings of the INCOSE Symposium*, 2001.

Vernadat, Francois, *Enterprise Modeling and Integration: Principles and Applications*. Kluwer Academic Publishers, 1996.

Biography

James Martin is a systems architect and engineer at The Aerospace Corporation developing solutions for information systems and space systems. Mr. Martin led the working group responsible for developing ANSI/EIA 632, a US national standard that defines the processes for engineering a system. He previously worked for Raytheon Systems Company as a lead systems engineer and architect on airborne and satellite communications networks. He has also worked at AT&T Bell Labs on wireless telecommunications products and underwater fiber optic transmission products. His book, *Systems Engineering Guidebook*, was published by CRC Press. Mr. Martin is an INCOSE Fellow and leader of the Standards Technical Committee.