



# The Unification of System Requirements Engineering & Systems Architecting

As presented to:  
INCOSE SF Bay Chapter  
10/14/08  
Scott Workinger, Ph.D.  
ScottWorkinger@gmail.com  
(707) 632-5134

(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

1

**These slides are taken from a presentation to the San Francisco Bay Area Chapter of the International Council of Systems Engineering on October 14, 2008. Although the slides are not intended as a standalone representation of the presentation, notes have been added in an attempt to serve as a reminder of the commentary from the meeting for those who attended. It is also important to note that a number of key slides are heavily animated and are only partially effective at conveying their intended meaning in static form.**

**Anyone interested in this and similar topics in Systems Engineering, Test Engineering, Technical Leadership and Knowledge Management is welcome to contact Scott Workinger at (707) 632-5134.**

**Scott Workinger**

**11/11/08**



# Topics

- Context
  - The Problem
  - Systems Engineering Processes
- Requirements Development
  - Allocating Requirements
  - Decision-Making During Requirements Development
- An Example
- The Architectural Process, a Patterns Perspective
  - The Structure of Architectural Patterns
  - Some Widely Used Patterns
  - Networked Computation Pattern
  - Layered Interface Pattern
- The Unification of Systems Engineering & Systems Architecting
- Consequences

(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

2

**This lecture speaks to a common problem in systems engineering organizations, a lack of coordination between the efforts of those individuals responsible for requirements development and maintenance and the individuals responsible for creating the architecture of systems. We will look at this problem from several perspectives:**

- First, we will look at the typical requirements process.**
- Second, we will introduce a notional example which illustrates the problem.**
- Third, we will look at the architectural process using a pattern-based architectural perspective.**
- Finally, we will examine how using an architectural patterns perspective allows us to unify the effort of the requirements development and architectural development efforts.**

**In closing, we will look at the consequences of adopting a unified approach to systems engineering and systems architecting.**



# The Problem



- Classical Approach – Two processes:
  1. Requirements are developed independent of implementation
  2. Architects begin work after they receive requirementsEach process is essentially independent (stovepiped)
- Issues:
  - Requirements writing is a decision-making process
  - Many requirements decisions are architectural
  - Most requirements developers are not architects
  - Architects lose respect for requirements that compromise the quality of architecture unnecessarily
  - Clients are sometimes quite unhappy with the result

(707) 632-5134

Unification of SE & SA

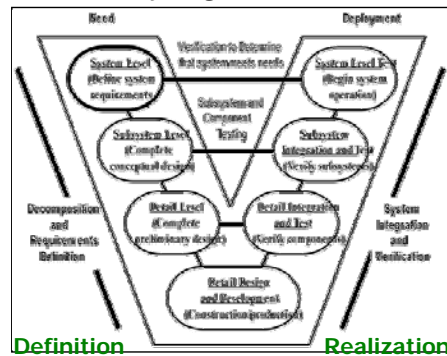
ScottWorkinger@gmail.com

3



# SE Development Process

- Agreement: Describing & Quantifying the need
- Defining Requirements
  - Analyze Missions & Env.
  - Defining Functions
  - Defining Technical Reqs.
  - Requirements Allocation
- Solution Definition
- System Realization
- Systems Engineering Support Functions
  - Analysis
  - Technical Management

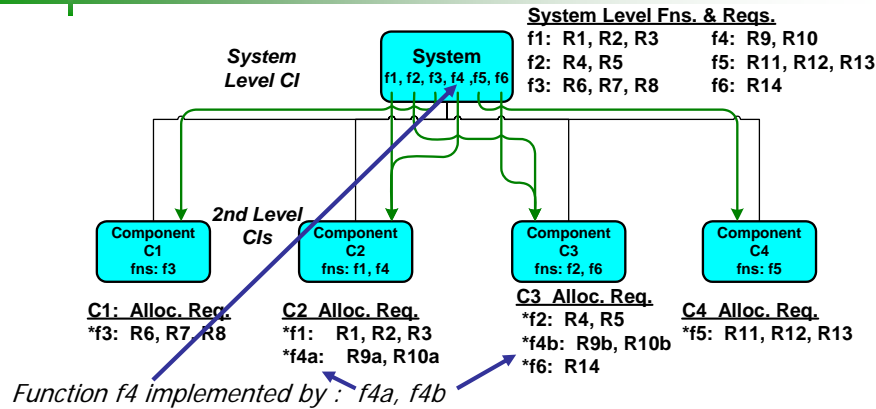


Classic "V" Model

**Note that the "V Model" is not the only possible approach. The V Model is essentially a Top-Down approach to the development of an architecture. Bottom-Up and Mixed Initiative approaches are also available. These other approaches would typically be discussed in a well-rounded systems of systems course. One very powerful Mixed Initiative approach is Architecting for Emergence. The development of the Internet is one successful example of Architecting for Emergence.**



# Allocating Requirements



- Requirements for lower level CIs are traceable to system level requirements.
- Every lower level function definition and/or allocated requirement implies architectural decisions.
- Function Definition & Requirements Allocation are recursive.

(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

5

## Legend:

fn x – Function number x

R y - Requirement number y

The classical approach to systems management is to use functional decomposition on a system and allocate requirements to lower level components in the design. When implemented correctly, this provides requirements traceability 1) from top to bottom (A requirement can be traced from system level to the lowest level components that support meeting the requirement.) and 2) from bottom to top (A component requirement can be traced to the systems level requirement from which it originates.) An important motivation for this approach is that when implemented properly it preserves the functional integrity of a system under development.

Note that allocating requirements is a recursive process. The recursive nature of this process will be important to us later in this discussion.



## Decision-Making During Requirements Development

- ❑ Scenario Analysis – What are the objects & trajectories?
- ❑ Defining Functions – What are the functions of the system?
- ❑ Defining Technical Reqs. – What are the performance levels required for the functions?
- ❑ Requirements Allocation – How are functions & requirements allocated to lower level subsystems and components?
- ❑ All these decisions have architectural aspects
- ❑ ***Usually, there are alternatives***



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

6

**Some large organizations have standardized function lists. For instance, the US Navy uses such lists for engineering development projects. The strength of such lists is that they help collaborating systems development professionals “get on the same page.” However, there is a potential weakness. They can encourage limitations in the view of systems architecture and place unnatural limitations on the potential performance of designs. Since all of these decisions potentially have architectural significance, making the decisions without the involvement of architects can seriously compromise a design.**

**In the following example, consider how the view of the problem (the objects, functions and processes) limited the design outcome.**



Example (1):

## University Library Info. Retrieval Sys.

- Need: Support Research and Education with Information Retrieval Services. (1995)
- Scenario: **Objects**
  - User walks to library,
  - User looks up books and periodicals in card catalog and science citation index (a book)
  - User retrieves books
  - If book not found:
    - User searches client/server database
    - Requests book shipment from collaborating library
- Functions Include:
  - Looking up sources in card catalog
  - Physical retrieval of books & periodicals



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

7

This is a notional example of a development project undertaken in 1995. The client is a research university that needs a system to support research and education with information retrieval services. Our requirements development professional works with the client to define the need, develop scenarios, analyze functions and develop requirements. The basic scenario is framed in terms of certain objects: A library, books, periodicals, a card catalog and a science citation index. The events in the scenario are abstractions of the outcomes of physical tasks. Functions are defined in terms of looking up sources in a card catalog and the physical retrieval of books and periodicals.

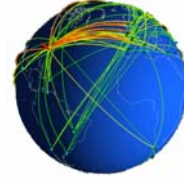
Many would have considered this to be a reasonable approach in 1995. It was within the scope of normal practice. But something is missing...



Example (2):

## Library Information Retrieval System

- What's missing?
  - Internet:
    - Developed in 70's,
    - Became important in 90's
  - Browser: Introduced in 1993
  - Commercial Search Engines: Introduced in 1994
- Only individuals in touch with trends would have been familiar with this alternative in 1995.
- If developers missed this:  
Would the client have been happy?



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

8

**So, our hypothetical requirements developer missed the relevance of the Internet in his requirements development process. Assuming that this would be a multi-year project, by 1999 when fielding was scheduled for the system, would the client have been happy?**

**Consider, are we being fair to our hypothetical requirements developer by suggesting that he or she would have missed the Internet? Any given individual might have been aware of developing Internet technology. But let's look at why a requirements developer in a stovepiped systems engineering organization would be likely to miss the Internet as an alternative in this situation...**



Example (3):

## Cause of the Problem

- Why might our stovepiped requirements developer be likely to miss networked computing options in 1995?
- Answer: The Role of Requirements Developers in a Stovepiped Organization
  - Requirements developers are not architects.
  - It's not their business to know how systems are put together.
  - It's not their primary business to stay current on new or existing architectural structures.
  - Iteration might have caught the issue, but iteration is inefficient and expensive
- The requirements developers were not part of the architects' community of practice



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

9

**The development of Communities of Practice within organizations is a major trend in the field of Knowledge Management. It's also important to keep in mind that Systems Engineers have a natural role in Knowledge Management within technical organizations. Specifically, as the integrative technical discipline, Systems Engineering has the role of integrating diverse communities of practice. Such integration is an essential aspect of creating a high-performing, integrated system in an efficient way. No other discipline has the necessary scope of responsibility, the knowledge, the necessary skills, or the view needed to achieve the integration of technical communities of practice.**

**When integrating Communities of Practice, what better place to start than within the Systems Engineering organization itself?**



# What is Architecture?

- Is it a science?
  - It uses science
- Is it an art?
  - Good architects are artists
  - The 'Art' of Systems Architecting*  
- Maier & Rechtin
- Is it a practice?
  - Body of Knowledge
  - Community of Practice
- What do architects know?
  - *Process & Technique*
  - *Structure*
    - *A Pattern Language* – Christopher Alexander
    - *Design Patterns* – Gamma, Helm, Johnson, Vlissides
    - *Systems Architecting Patterns* – Emerging...



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

10

**A good architect creates delight!**

**Systems architecting patterns have been in use for many years. What is emerging, today, is the formalization of such patterns. Formalizing the patterns helps us to talk about patterns, compare them, and analyze their strengths and weaknesses.**



## Architectural Patterns



“...a solid and reliable invariant which relates **context**, **problem**, and **solution** in an unchanging way.”

- Christopher Alexander

When a design works, the essence of the concept, that which makes it work, is its architectural pattern.

(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

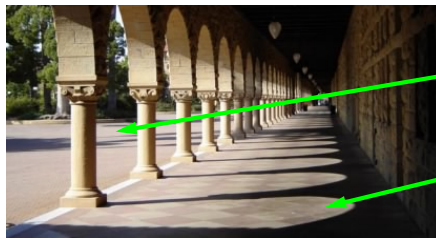
11

**The concept of an architectural pattern was first formalized in Civil Architecture, but has been adapted to other fields. The concept is based upon the notion that certain patterns of organizing systems recur because they are effective and powerful. Whether formalized or not, such patterns constitute a lot of knowledge that separates an experienced architect from a beginner. “Patterns stay alive because the people that are using them are also testing them.” – Christopher Alexander. A pattern language is a set of powerful tools. “It is the structure of the underlying language that is doing most of the hard work.” When a design works, the essence of the concept, that which makes it work, is its dominant architectural pattern and the lower level patterns that complete it.**

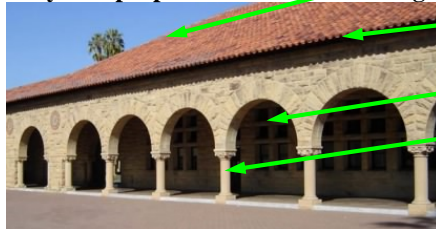
**Note that this slide and several others following were adapted from the course: “Systems of Systems, Technical Leadership in a Networked World.”**



## The Arcade Pattern (2 Views)



“Arcades – covered walkways at the edge of buildings, which are partly inside, partly outside – play a vital role in the way that people interact with buildings”

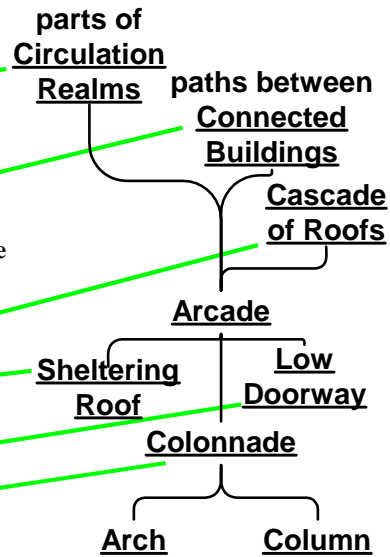


(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

12




A classical example of a pattern from Civil Architecture is the “Arcade Pattern.” An arcade pattern occurs within the context of Circulation Paths within a complex of buildings, paths between connected buildings, and a “Cascade of Roofs” within a structure. The problem that an arcade solves is to make a building more accessible in a psychological sense. The pattern is completed by patterns such as the “Sheltering Roof,” “Low Doorway,” and “Colonnade.” Colonnades can be completed by arches and columns. Well-designed arcades tend to see a lot of pedestrian traffic.

In the two pictures, an example of an arcade is seen from inside and outside. The “Circulation Realm” is a plaza area at the center of Stanford University. Note that a colonnade can be completed by arches and columns. However, a beam and column approach is a viable alternative. The arcades at Stanford see a lot of pedestrian traffic.

In this example, we’re looking at 5 levels of patterns within the architecture.



## Format of an Architectural Pattern

- Name: *Arcade*
- Picture: (multiple views) 
- Context:
  - parts of *Circulation Realms*
  - paths between *Connected Buildings*
  - *Cascade of Roofs*
- Problem: Lack of connection between the territory inside the building and outside the building <or>
- Problem – Satisfy the Requirement(s): The territory inside the building and outside the building shall be connected.
- Solutions: Whenever paths run along the edge of buildings, build arcades, above all, to connect up the buildings to one another, so that a person can walk from place to place under the cover of arcades
- Completing Patterns:
  - *Sheltering Roof*     ■ *Low Doorway*
  - *Colonnade*

Adapted from: A Pattern Language  
- Christopher Alexander

(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

13

The format of an architectural pattern includes:

- A name for the pattern,
- A picture or schematic of the pattern in one or more relevant system views,
- The application Context for the Pattern – This is a list of patterns that the current pattern might be used to help complete.
- A statement of the problem that the pattern solves. - This is usually the longest part of the pattern description. It often contains examples.
- The Solutions – This is a description of how the solution solves the problem. If the pattern can solve more than one problem, then this section may have several parts. Typically this section will mention the chief parameters for the pattern, how the values of the parameters are chosen, and the typical range of the parameters.
- Completing Patterns – These are patterns that may be used to complete the design.

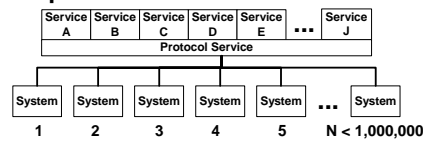
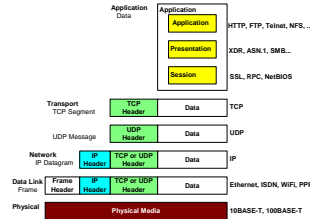
It's significant that patterns tend to form a hierarchy. The order in which a series of patterns is applied can significantly affect the resulting architecture.

A key point that we will examine further is that there are two alternative ways of formulating the problem. Classically, Christopher Alexander would have phrased the problem as indicated in the first "Problem" statement. However, from a systems engineering point of view, it is useful to pose a problem as the satisfaction of a list of requirements, as in the second "Problem" statement. (See above.)



## A Sampling of Patterns in Large Systems

- Networked Computation
- The System of Systems
- Interface Patterns
  - Layered Interface Pattern
  - Interface Constitution
- Stateless Server
- Integrated Operations Space
- Open System
- COTS
- Reachback



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

14

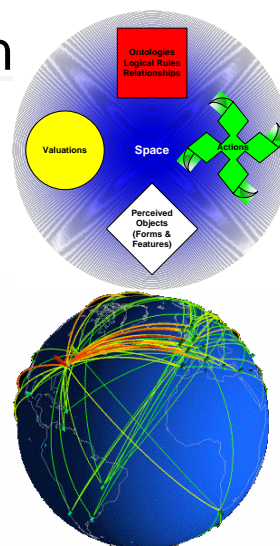
There are many important patterns used in various aspects of systems architecture. This is a somewhat ad hoc selection of patterns used in large systems, today. We will look at several of them briefly, as examples in the coming slides. Many of these are discussed more fully in the Systems of Systems course from which some of these slides are adapted. Some of them are also discussed in the lecture, "Systems of Systems, Where's the Beef?" scheduled for January, 2009 at the San Francisco Bay Area Chapter of INCOSE.



Introduction:

# Networked Computation

- A New Dominant Paradigm
- Aspects of New Paradigm
  - First Many-to-Many Mass Medium
  - Expanded Size of Systems  $10^3 - 10^6$ 
    - Issue with scaling SE →
    - Systems of Systems Engineering
  - Effect: Functionally Smarter (About some things)
  - Increased Pace of Activity
- What's it becoming?



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

15

***As a paradigm, Networked Computation has changed the world. Networked computing can connect systems together on a vast scale. The Internet, itself, is a System of Systems. In this and subsequent slides we will be briefly examining some of the fundamental structures of the Internet as examples of useful patterns in SoS architecture. We will be looking at some of the structure that makes the Internet work.***

***The expansion of the Internet itself is an example of unplanned emergent behavior. Most of the Internet services we use today were not envisioned when Vinton Cerf and his colleagues invented the basic technology for the internet in the early 70's. The power of the Internet is perhaps our foremost example of the triumph of good architecture to produce powerful emergent results. A few of the many aspects of the Internet include:***

- ***First Many-to-Many Mass Medium – Arguable as important as the printing press***
- ***In 10 years the scale of systems we've been asked to build has increased, from a thousand-fold to a million-fold***
- ***The Internet has created a scaling issue for Systems Engineering practice***
- ***This is origin of the Systems of Systems Concept. Without networked computing, the term "System of Systems" would probably never have been coined.***
- ***Psychologists measure a person's ability to learn and call it intelligence. The Internet makes it easier to learn. Example: Ability to do web searches and to use tools like Wikipedia makes it possible to find and understand relevant information more quickly.***
- ***Increased Pace of operations***

***Improved communication mean that we are interacting more strongly on a grand scale with vast creative opportunities. That's the good news. The bad news is that this has led to global conflict. In some sense, these changes started with new systems architecture patterns. Ready or not, we've entered the age of the systems engineer. Welcome!***



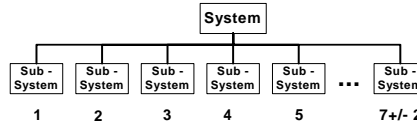
Pattern Example:

## Layered Interface Pattern

- Traditional Structures use *Modest Fanout Pattern*

- *Problems for Large Systems:*

- *Too many levels*
- *Managing complexity*
- *Scaling*

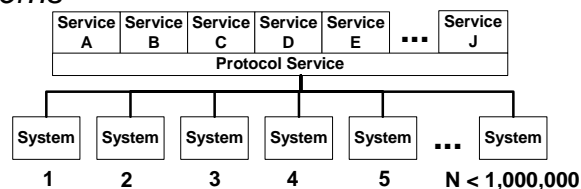


- *Solution: Layered Interface Pattern*

- *Plus Completing Patterns*

- *Examples*

- *Internet*
- *FORCEnet*



(707) 632-5134

Unification of SE & SA

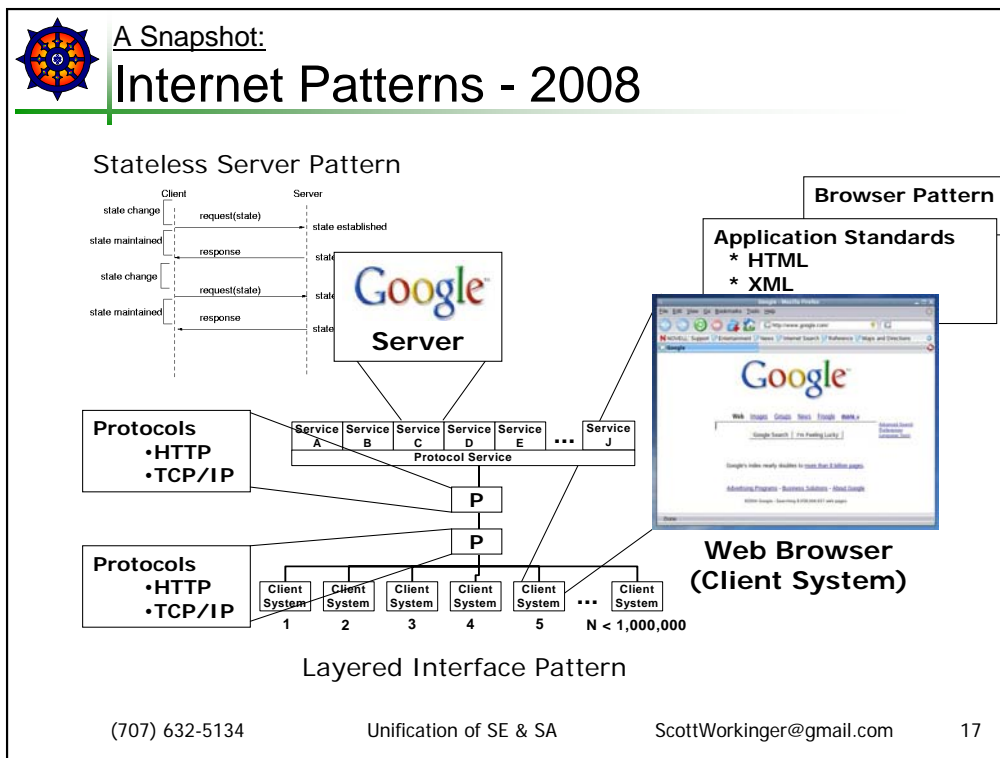
ScottWorkinger@gmail.com

16

Traditionally one of the architectural patterns used to manage complexity was the Modest Fanout Pattern. In the Modest Fanout Pattern, approximately  $7 \pm 2$  modules are used on each architectural level in a design. Note the origin of the number  $7 \pm 2$ . It actually comes from the fact that a human's short term memory can hold approximately that number of items. So, there's nothing magical about the number 7 in our effort to organize complexity. It's simply one approach. The problem is that in really large systems, the modest fanout approach can lead to a lot of levels and these levels don't necessarily have any functional significance.

The Layered Interface Pattern is an alternative approach used to manage complexity. In this approach services are organized in layers. Services communicate with their clients through a protocol service using standard protocols, sometimes called an "Interface Constitution." The Internet uses this approach. The Layered Interface Pattern is typically one of the highest level patterns in a system where it is applied.

In a few minutes we'll discuss some of the completing patterns that are also required in combination with this one.



This is a snapshot of some of the most important patterns used in the Internet, today. The Internet is still growing. New patterns are continuing to emerge. Note also, that this figure does not include hardware patterns, such as the Router Pattern.

Interfaces are often lower level “completing patterns” in an architecture. For instance, the TCP/IP protocol suite is a crucial pattern in the structure of the Internet. It was invented in the early 1970’s and its presence was one of the key facilitators of the rapid growth of the Internet 25 years later. The Internet is the Largest System of Systems on the Planet. At the highest level, the most commonly used features of the Internet make use of the architectural patterns known as

- the layered interface pattern,
- the stateless server, and
- the browser.

The browser, today’s most ubiquitous application relies upon web page representation standards that allow many people to use similar browsers. These operate at an “Application Level.”

At a lower level, the TCP/IP protocol suite describes the basic interfaces that form the foundation on which higher level functions operate. The interface protocols are what make all of the connections work. This is an example of the “Constitution Strategy” for interfaces in a System of Systems.

Essentially, the Internet began with TCP/IP, arguably the most successful interface (or more properly, set of interfaces) in history. These patterns have changed our world. Note also that there are relatively few of them. We can get a lot of the most important ones into a single figure.

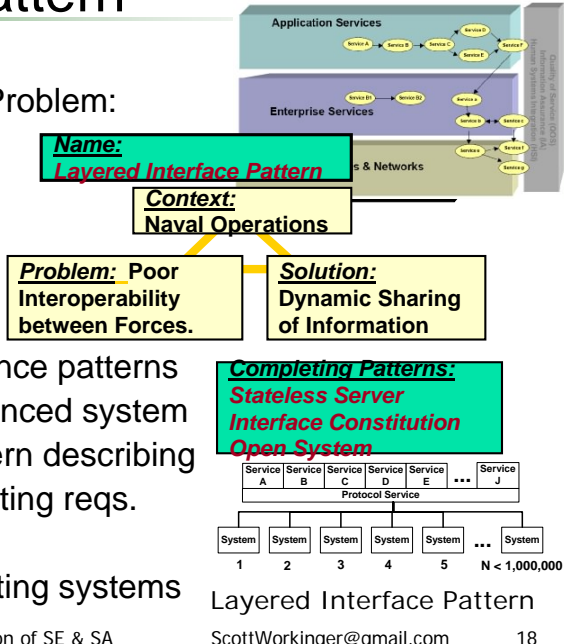


Creating Architecture:

# Applying a Pattern

- Gather & Summarize – Key Aspects of Design Problem:
  - Needs Statement
  - Requirements
  - Existing Footprint
  - Existing Patterns
  - Hotspots
- Gather the tools: Reference patterns
- Visualize: the new enhanced system
- Pick: Highest level pattern describing the revised system, meeting reqs.
- Apply the pattern
  - Bind it to the pre-existing systems

FORCenet applying Layered Interface Pattern



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

18

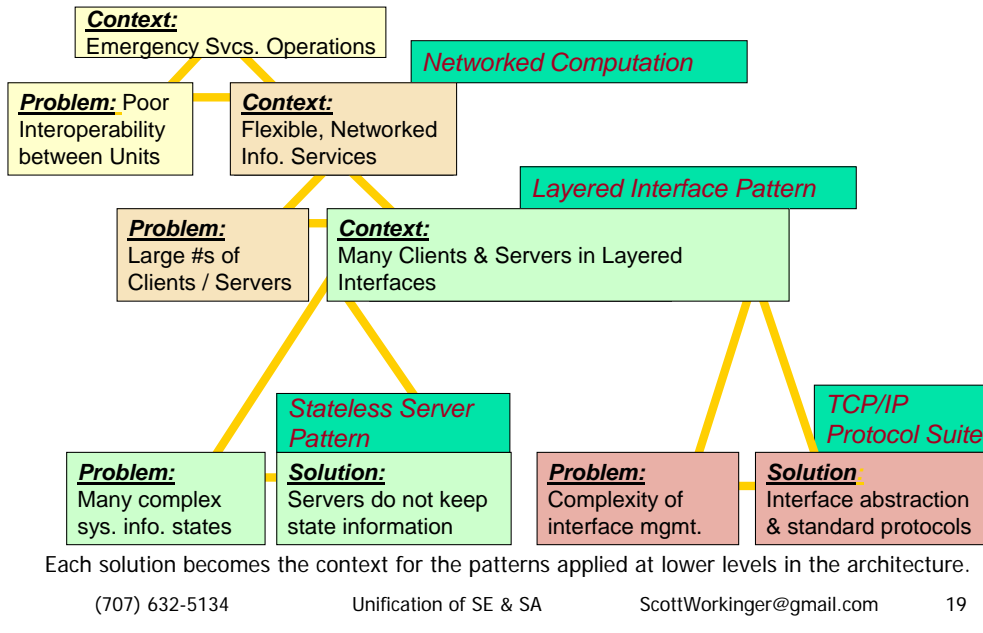
<Approximate rendering of heavily animated slide>

In the pattern-based approach to architecture, the architectural process begins with preparation to choose the primary (or highest level) pattern in the architecture. There should be a clear statement of the design problem, based upon the previous stages of the process that generated a needs statement and a revised set of requirements for the new capability. Initially, all the requirements are allocated to the top level of the system. In the case of capability engineering/reengineering for an SoS, the architectural patterns used for the existing pattern are a reference point for how the existing systems were designed. (Typically, one doesn't get to build an SoS from scratch. Capability engineering/reengineering is the norm. However, the steps are similar for beginning an architecture from scratch.) The footprint of the desired capabilities show which systems are most relevant to the capability engineering effort. A study of the design rationale for the existing systems will often indicate that original design assumptions have changed. (Changing scope is often an issue in capability engineering efforts.) The hotspots are candidate intervention spots. Key activities for applying patterns in a System of Systems include:

- Gather whatever reference material may be helpful in understanding the relevant design patterns for the existing system and possible alternatives.
- Identify reference patterns that are candidates for use in the architecture.
- Use the available views of the existing system to study the capability footprints. At this point it's good to spend a little time with whatever creative techniques work for you visualizing the new capability-enhanced system.
- Pick the highest level pattern which best meets the requirements for the capability-enhanced system.
- Applying the new pattern, binding it into the existing structure, using the footprint as a starting point and giving particular attention to the hotspots.
- Note the higher level patterns inside which the chosen pattern functions. These patterns may suggest collaborative working relationships with other systems engineers.
- List lower level patterns that are candidates for completing the applied pattern. (If you have complete documentation for the applied pattern, it should come with a list of lower level pattern candidates to complete the pattern you're applying.)



# Pattern Application is Recursive



**Context:** Emergency service operations, supporting coordination of fire, police and emergency medical services. With multiple jurisdictions and types of agencies, this is a potentially complex problem

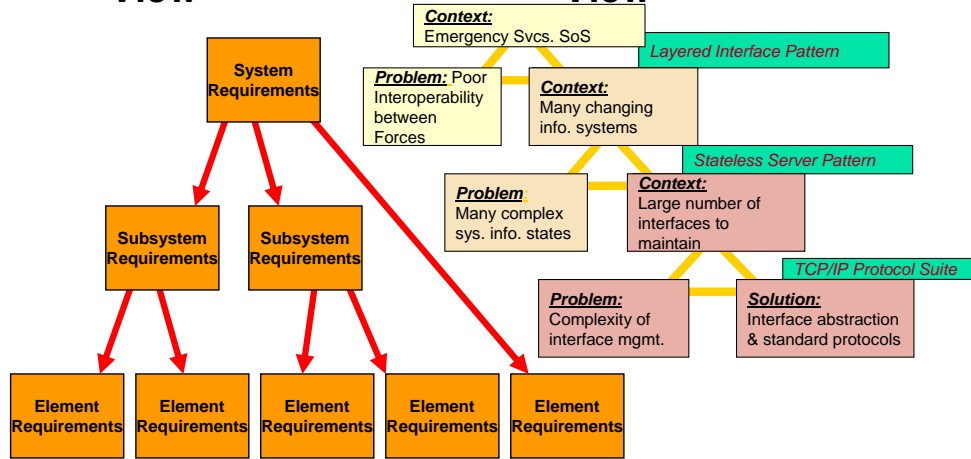
Note that we've encountered another recursive process. You'll recall that we also looked at the allocation of requirements as a recursive process.



The Unification:

## Systems Engineering & Systems Architecting

### Systems Engineering View      Systems Architecting View



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

20

<Approximate rendering of heavily animated slide>

***One of the classic perspectives on systems engineering is that it's mostly focused upon requirements management and hence distinct from systems architecting. But this is misleading. In practice, the process of allocating requirements cannot occur in isolation. When allocating requirements to lower levels in a design, there must be decisions about how the design is organized, in effect, its architectural structure. From an architectural perspective, whenever one applies a pattern, effectively one formulates lower level architectural problems and creates the need for completing patterns at lower levels within the architecture. Each of these lower level architectural problem necessarily has its own requirements which derive from the higher level structure of the architecture. In other words, developing the requirements for solving the lower level architectural problems essentially requirements allocation. So, the systems engineering view and the architecting view are simply two views of the same problem. Moreover, the distinction is quite artificial.***

***One cannot do competent systems engineering requirements development without systems architecting. One cannot do competent systems architecting without systems engineering requirements development. Language aside, systems engineering and systems architecting are inherently two aspects of the same thing and the processes are so closely interrelated that they are not naturally separable. An organization cannot be good at either process without engaging the other.***



Unification:

## System Reqs. Development & Architecting

- A Meeting of the Minds:
  - Architects: Get involved w/ requirements development early & stay involved
  - Requirements Developers: More attuned to Architectural Process
  - Architects & Requirements Developers in same extended Community of Practice
- Requirements become more powerful tools for architects
- More efficient & effective engineering process
- Better quality architecture
- Better value for clients
- Future: Requirements & architectural patterns maintained in integrated databases



(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

21

**In contrast to stovepiping, a unified process of systems engineering and systems architecting is characterized by a meeting of the minds, sound working relationships between the requirements developers and the architects throughout the development process. This should start early. Every proposal team with ambitious goals should have architectural representation. How can one be sure that a quote is competitive without architectural insight?**

**When this presentation was first delivered, the author predicted that requirements databases would some day include architectural patterns. In the discussion following the presentation, one of the master architects in the room, an individual who works for a major aerospace company, observed that her organization is already pursuing this option with the vendor of one of the most popular requirements databases.**

**First rate systems engineering organizations unify their requirements development and architectural processes.**



## Further Information



Scott Workinger, Ph.D.  
(707) 632-5134  
P.O. Box 30  
Jenner, California  
ScottWorkinger@gmail.com

(707) 632-5134

Unification of SE & SA

ScottWorkinger@gmail.com

22