Singleton To Sandwich
Chunking Into Buslets for Better System Development

Robert H. Hodges
Doctoral Student
Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030
robert.h.hodges@lmco.com

Robert J. Cloutier
Associate Professor
Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030
robert.cloutier@stevens.edu

Mary A. Bone
Doctoral Student
Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030
mbone@stevens.edu

Peter Korfiatis
Doctoral Student
Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030
pkorfiat@stevens.edu

Abstract - *With the rising cost of developing increasingly complex software intensive systems, improved strategies to join together legacy systems are desirable. Connecting black boxes not easily customized by the integrator requires architectural innovation and new integration practices. Middleware software solutions turn disparate hardware systems into system of systems service oriented architecture but there still needs to be a common underlying service with a consistent data structure to the overall system. This paper proposes an innovative approach to transposing software subsystems into the dominant flow down system engineering process used in many large programs today, specifically; large legacy bound architectures The paper suggest this can be accomplished by creating a hybrid architecture pattern from two established software patterns, singleton and sandwich, which would allow users to phase in service availability long before dominant system engineering processes are completed.*

Keywords: Service orientation, systems engineering, buslet, chunk.

1 Introduction

Disney brought the World of Color (WoC) to the already existing California Adventure Park in Southern California in 2010. This 25 minute 3D show of light, music, lasers, and water set to classic Disney music have thrilled audiences of all ages and have met the high expectations the Disney label brings. Said to cost about $75 million, some of the subsystems comprising WoC include:

- Nearly one full acre of engineered superstructure, longer than a football field and capable of settling on three levels – one for the performance, one under the water surface and one for maintenance.
- More than 1,200 powerful and programmable fountains.
- A vast underwater grid with more than 18,000 points of control. Each fountain has multiple points of control for lighting, color intensity, water angle, height and more.
- A precise system of flame projections, lasers, and special effects that will flood the senses as guests experience the animation.
- Nearly 30 high-definition projectors. [1]

Traditional content (the legacy) delivered with state of the art fluid simulation tools created one of the most complex symphonies of physical assets ever assembled. Underneath, established technologies such as hydraulics, computer controlled lighting and projection systems were integrated piecemeal with motion picture computer graphics tools. WoC's development included concept, storyboards, pre-visualization, technical R+D, animation, fluid simulations, light, rendering and final integration of all 3D elements. [2] Challenges faced included management of 12 terabytes of legacy data and the complexity of integrating a variety of technical systems. [3]

The data, syncrhonization of equipment, communications, and systems thinking are required to address availability, cost, and performance [4] to design and implement the WoC are not unlike net centric operations in the battlefield.

In today's large scale integration space is comprised of large legacy bound architectures limited by customer

preference for a service oriented (SO) approach which require the phasing in of system service availability long before dominant system engineering processes are completed. The approach described in this paper is aimed at innovating existing system of systems engineering processes to cut development time, bringing better value to the customer and buying down risk to assure high level technical readiness in SO system of systems. The authors propose this can be accomplished through the application and integration of traditional software patterns (the singleton and the sandwich patterns) during systems architecting.

SO implies systems characterized by publishing information about resources available. SO participants can request resources on demand for use but cannot modify them. SO is an approach to building heterogeneous distributed systems. Presented with a water gun, a Disney WoC developer might envision a vapor screen showing animated features, a fireman might envision a tool to put out fires, and a child might envision hours of fun on a hot day. The water gun is the service. Each independent user of the water gun calls upon it as needed for a specific use, but at the end of the day they do not change the essence of the water gun (it takes in water and sprays it out at a higher velocity). What changes is how the water gun is connected to each user's system. Service bus (SB) element is important to mention as enablement for assimilating systems. Red Hat's Len Dimaggio, who offers a SB to link SO applications on the software side states "one of the joys of software development is the ease with which you can create complex stuff out of thin air – or your own imagination. Creating new software may satisfy a need that could not otherwise be met. However, you can take advantage of a standardized way of doing things, so that you don't have to start from scratch every single time". [4] SO is a style of architecture implemented with such attributes as complexity. Accorging to Waldrop [6], the design of a system with SO style is as complex as the stakeholders needs for the system but "each of these [sub]systems is a network of many 'agents' acting in parallel . . . a complex adaptive system has many levels of organization, with agents at any one level serving as the building blocks for agents at a higher level." Understanding where the complexities are and how to minimize the implementation risk is explored.

System of Systems (SoS) modeling, using SysML and a system engineering methodology has been well covered and applied to a coalition maritime domain. [5] This paper proposes an approach, based on practioner's experience, for integrating complex systems using patterns. Service Oriented Architecture (SOA) has been previously examined in literature using SE approaches. Lewis and Smith explicitly lists an example of a SOA SoS beginning with a pilot project. [6] Calinescu and Kwiatkowska developed a SoS framework including the SOA implementation of reconfigurable policy engines as web services. [7] Sloane discussed the ubiquity of SOA and its SoS complex interactions in DoD net-centric perspective [8] and a SOA-enhanced US National Healthcare Information Network in a complex SoS information network. [9]

2 Singleton to Sandwich - Patterns in Play

The concept proposed in this approach is to leverage established architectural patterns and not "invent" a new pattern without basis. It is informed by the author's observations over the past decade of successful and unsuccessful SO influenced systems development. "A dominant design incorporates a range of basic choices about the design that are not revisited in every subsequent design. . . once any dominant design is established, the initial set of components is refined and elaborated, and progress takes the shape of improvements in the components within the framework of a stable architecture." [10] The goal of this approach is to propose improvements on how SO systems are integrated. The observations are independent of computing languages and hardware specifications. Taking the enterprise as a whole, SO architectural style principles can apply to systems development. "An enterprise architecture for an organization combines and relates all architectures describing particular aspects of that organization." [11]

The singleton architecture pattern is a dominant design instantiation as described by Henderson [12]. However, it is discussed here in a new light. The singleton architecture pattern has roots in the software discipline but can be applied equally well to system of systems design by restricting the implementation of a subsystem to a singular instantiation void of global state. This serves to act as a good isolator of single purpose. For a SO architecture type, systems of similar functionality that are constrained by legacy tradition as was the case with the existing lighting system around the California Adventure Park where WoC was to be installed, the singleton can be used to wrap the legacy systems and expose them through software elements as services. Singleton architecture patterns can lead to replication for system-wide resources in the form of factory architecture patterns and command architecture patterns.

The other architecture pattern to be examined in this approach is the sandwich architecture pattern. The name derives from imagining a SO architecture type system with subsystems such as WoC water hydraulics control made available on a bus (the lower piece of bread) to a middle layer for connection and orchestration of business events (this is the meat portion of the sandwich) topped by a layer

devoted to making services or systems available to consumers (the upper piece of bread). Logically in software, the lower layer is often shown as a SB while the middle layer is typically shown in as a mix of business logic built as components to insulate the service consumers from the deployment of the service infrastructure. The middle layer is where complexity engineering can be applied with increasing success. Figure 1, the sandwich pattern implies two SB layers but these are logical layers, not physical instantiations.

Figure 1: Sandwich Architecture Pattern

Chunking is an approach to breaking something up (software code, information, etc.) or partinioning it in a way that the pieces are still related in some meaningful way. For instance, the social security number is a 9 digit number. However, it is normally chunked in a pattern of three digits, two digits, and four digits in the form of 123-45-6789 to facilitate memorization and recall. The combining of singleton with sandwich architecture patterns using SO chunking and buslet style SB components are at the heart of the premise that large legacy bound architectures can phase in service availability long before dominant system engineering processes are completed. Major middleware makers of the software version of a SB called enterprise service bus (ESB) such as IBM, Oracle, and Red Hat share a similar implementation premise: Through configuration control one ESB can proxy one wrapped SO web service at a time then later the ESBs can be locked together. So-called "SO chunking" is the act of wrapping individual SO type system elements controlling specific hardware assets then connecting them to an isolated ESB not connected to any other service endpoints. Once connected, simple business logic and user experience is binded to the chunked subsystem for control and function prototyping and testing. The application (i.e. driver) is then able to communicate as an exposed service on a very limited singleton unit. Coining a new term, this "buslet" will have minimal connection, eccentric data constructs and no global state. The "buslet" will be able to make its services available to the enterprise. This approach leverages existing architecture patterns to create "buslets" capable of stepwise integration maximizing bottom up system integration. In doing so, a number of buslets emerge as separate ESB pieces which if implemented correctly can be connected into one SB. The case where legacy systems are large pieces of a service oriented system of systems (SoS), a new system engineering process emerges.

The most important claim in this approach is that each buslet designer, while needing to be mindful of particular domains and persistent patterns from allocated system capabilities, may move forward early on with the ardent task of bottoms up integration.

Singleton buslets with SO chunking figure prominently into the lower portion of the sandwich. At issue is the resolve to combine buslets for the purpose of making services available to the middle portion of the sandwich. "Complexity rules of engagement subscribe to the notion that systems with SO are complex systems in themselves – a reflection of the problem and the solution domains" [13] implying complexity engineering may be required to assemble the variables, attributes and data structures. That is a subject for further research.

At the same time singleton buslets are being built by the system implementation team, a top down systems engineering process is taking shape. Figure 2 shows the starting singleton structure for chunking in the software domain. The legacy application is glued into a SB using one of many methods, and is supported by a SO infrastructure (e.g. lightweight data bases, system emulators, hardware), governed by a few policies, driven by a limited workflow (enough to show functionality) and driven by a simple user interface.

Figure 2. Singleton chunked SOA

Singleton buslets are not unlike singleton design patterns for software development. In programming, the purpose of singleton is to ensure an object will only have a

single instance. This results in a single point for access from anywhere.

Once the buslets are formed, a network of service availability can be discovered and connected using common standards and protocols. As the systems engineering process evolves, business logic and system user interfaces can be programmed to tie mission operations together using the buslets as building blocks. This results in the chunked sandwich approach.

2.1 Consumer to Service Availability

Contributors to better life cycle performance include ROI, risk reduction, and reuse over program phases. Risk may be reduced when the tough integration problems are solved up front. "A dominant design incorporates a range of basic choices about the architecture that are not revisited in every subsequent architecture. The initial set of components is refined and elaborated and progress takes the shape of improvements in the components within the framework of a stable architecture." [14] Stevens Institute [15] suggests a process for systems engineering development shown in Figure 3 however any number of processes can be adapted to a particular domain. Note the Stevens approach enables persistent patterns for phased development so the phases avoid stagnation during refactoring.

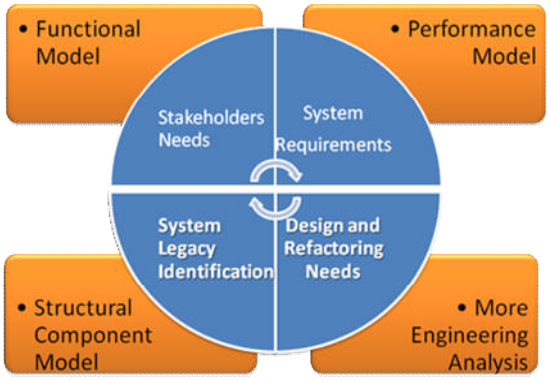

Figure 3. Systems Engineering Process

The relative improvement of ROI as the risk is reduced and parts of the system begin to build can be characterized by the chart shown in Figure 4. The ROI of a large scale integration project is difficult to estimate during the early phases simply because the long lead time of the various reviews. As the chunks build, more reuse is possible, the marginal risk reduces, and the understandability factor begins to dominate. The ROI takes a dramatic upturn once the systems engineering process value is realized. Implicit knowledge is gained at each phase of the project over the life cycle of the development. More research is needed to specifically quantify Figure 4.

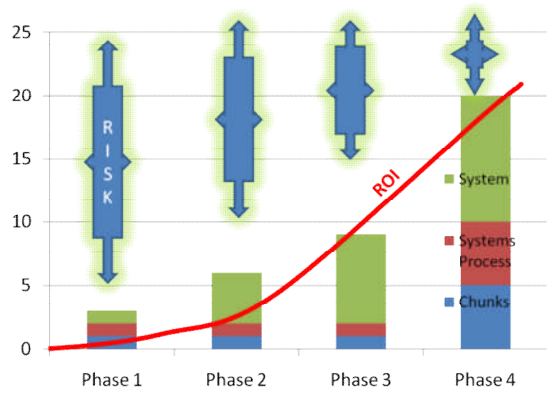

Figure 4 Life Cycle Chunking Effects

The classical testing solution to singleton architecture pattern is to employ mutual exclusion principles. Unit testing will be limited to ensuring the buslets have available services. Without strong test cases built up from unit tests and allocated Key Performance Parameters (KPPs), the buslet development will quickly lose focus.

2.2 Challenges

This approach is not without challenge. Introducing global state into a buslet as the services are integrated can be unwelcome. Removing buslet locking to allow multi-threaded context could be difficult and not conducive to parallel processing. Refactoring the buslet as a means of meeting KPPs could undermine the time saved. In the Java programming language and in some systems development, an executing program can examine itself and manipulate internal properties of the program to reduce the need for new refactoring.

Generalization of software architecture patterns for systems use is fraught with issues including the relative level of abstraction versus implementation blueprints. "The architecture of a system. . . requires a higher level of abstraction than necessary for the software that may be a part of the system." [16]

2.3 Implementing the Chunked Sandwich Architecture Pattern

Disney treated WoC as a system of services, not as an isolated show on the list of events. Disney worked diligently with a number of innovative contractors to ensure that all stages of the program were handled seamlessly and without incidence. The resulting "magic" is an excellent visitor experience through the chunking and integration of their legacy film content into a newly designed fluid dynamic system which was integrated into their legacy theme park.

Motion Theory supervisor John Fragomeni was the lead on the computer graphics (CG) for WoC. He recalls that the new levels of fluid dynamics were required to "attain the level of complexity and detail required to match the sheer scale of the live show. The show has a complex mixture of water tricks, ranging from traditional fountains, background mist projection screens, and shafts of water shot 150-200 feet in the air under extreme pressure, which produced these fine mist streams that dissipate very quickly". [17] Each element of the physical system was exposed to the CG as a service, enabling pipelines (or workflows) to manipulate the services in various ways during the duration of the show. To form the animated characters out of water the water fountains needed to be invisible compared to the projected lifelike icons. Simulations, layered and rendered together, created the final overall look for each character that was then composed through business logic into segments of the show. The role of Motion Theory is akin to the middle and upper portions of the sandwich architecture pattern while the projection, fountains, lights, etc were chunked into singleton buslets for ease of binding as services. This created the SO system. 150 simulations and a volume of over 12 terabytes of data later, the modeled, rigged, matched animated characters morphed into 3D feature film.

3 Conclusions and Further Research

This paper explores the leveraging of existing architecture patterns through better integration strategies to reduce the time to market for systems based on SO Architectures. The approach is to transpose system of systems into the dominant flow down system engineering process used in many large programs today, specifically; large legacy bound architectures The paper discusses how this might be accomplished by creating a hybrid architecture pattern from two established system architecture patterns: specifically, the singleton and sandwich, which would allow users to phase in service availability long before dominant system engineering processes are completed.

Key to understanding how a chunked legacy system can integrate seamlessly is the nascent research of complexity. "The advancement and availability of robust event-driven platforms provide an unique opportunity for architects within mission-critical DoD systems to associate high-volume temporal data with traditional operational IT systems from the edge of the network at the points of force projection and threat capture back to centralized command locations." [18]

Equally important is to consider building systems on standard commercial baseline and not custom technology. Data integration, some 12 terabytes in the 3D fluid dynamics model of WoC, is important to consider.

"Addressing all possible security issues is complex," [19] according to Carlos Gutiererez. "Different issues need to be considered from different perspectives. " Gutierez led a team to developed a model of SO security that should be investigated further.

Other potential topics for consideration: 1) Build a prototype and analyze the life cycle impact of this proposed architectural approach. 2) Better understanding the strain of new innovation technology investment on established architecture disciplines.

References

[1] Disney, Inc., "Innovative Entertainment Technology Drives 'World of Color' – the Next Milestone in Expansion of Disney's California Adventure," http://www.disneylandnews.com/press+releases/world+of+color.htm, July 2009.

[2] Motion Theory, http://www.motiontheory.com/work/disney-_world-of-color, July 2009.

[3] *Ibid.*, Motion Theory.

[4] Hodges, Robert H., "2010 Lockheed Martin Net Centric Operations," *Lockheed Martin Connect 2010*, pp. 5-6, October 2010.

[5] Len DiMaggio, "Adpapters for and ESB," Red Hat Magazine, http://magazine.redhat.com/2008/05/22/adapters-for-an-esb/, p. 1, May 2008.

[6] Waldrop, M. Mitchell, *Complexity, The Emerging Science at the Edge of Order and Chaos*, Simon and Schuster, New York, New York, 1992.

[7] Huynh, Thomas V., Osmundson, John S., "A Systems Engineering Methodology for Analyzing Systems of Systems Using the Systems Modeling Language (SysML)," *2nd Annual Conference on System of Systems Engineering Center of Excellence*, p. 7, 25 July 2006.

[8] Lewis, Grace A., Smith, Dennis B., "Four Pillars of Service-Oriented Architecture," *CrossTalk Journal of Defense Software Engineering*, p. 10, September 2007.

[9] Calinescu, Radu, Kwiatkowska, Marta," Software Engineering Techniques for the Development of Systems of Systems", *15th Monterey Workshop on Foundation of Computer Software* , p. 3, September 2008.

[10] Sloane, Elliot, Way, Thomas, Gehlot, Vijay and Beck, Robert, "Using hybrid SoSE approaches for modeling and validating large scale Service Oriented Architecture (SOA) System of Systems as nextgeneration global military informatics platforms with Colored Petri Nets (CPN) and Extend/MESA, " *Proceedings of the Second IEEE International Systems of Systems Engineering Conference*, p. 1-2, April 2007.

[11] Sloane, Elliot, Way, Thomas, Gehlot, Vijay and Beck, Robert, "Using SoSE Modeling and Simulation Approaches to Evaluate the Potential Security, Performance, and Limitations of a Next Generation US National Healthcare Information Network (NHIN-2): Simulating a Service Oriented Architecture (SOA) to create an extensible, context aware, dynamic discovery framework for robust, secure, flexible, safe, and reliable healthcare information management," *Proceedings of the 1st IEEE Systems Conference*, p. 3-4, April 2007.

[12] Henderson, Rebecca M., Clark, Kim B., "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," Cornell University, p. 14,1990.

[13] Stojanovic, Zoran, and Ajantha Dahanayake, "Chapter VII - Service-Oriented Enterprise Architecture," *Service-Oriented Software System Engineering: Challenges and Practices*, IGI Global, 2005.

[14] *Ibid*., Henderson, p. 15.

[15] *Ibid*., Waldrop.

[16] *Ibid*., Henderson, p. 15.

[17] Cloutier, Robert. SD750 lecture notes, 2011.

[18] Cloutier, Robert, *Applicability of Patterns to to Architecting Complex Systems,* VDM Verlag Dr. Mueller Aktiengesellschaft & Co. KG, Saarbruecken, Germany, 2008.

[19] Motion Theory, "Disney 'World of Color'," http://www.realflow.com/casestudies/cs_33worldcolor.php

[20] Bostrom, Peter, Chow, Linus, "Business Process Management, Service-Oriented Architecture, and Web 2.0: Business Transformation or Train Wreck?", August 2008.

[21] Gutierrez, Carlos et al, Web Service Security Development and Architecture: Theoretical and Practical Issues, IGI Global, Hershey, Pennsylvania, 2010.