Making the Case for SOA

Arlene F. Minkiewicz

Introduction

A Service Oriented Architecture (SOA) is a computing environment in which applications are composed, rather than developed, through a set of standard interfaces. A SOA takes advantage of networking capabilities to compose applications through the coupling of services in a way that is independent of architecture, programming language, development platform, and vendor.

SOA creates an environment in which the logistics of deploying business applications is separated from the applications themselves. A SOA consists of various layers of capability, separating those capabilities common to all applications – security, data handling, transaction processing, etc. – from those applications that implement the rules specific to a business' core competency. Once a SOA infrastructure is in place, applications can be composed using services that implement the business rules. This loose coupling of the business logic simplifies making changes to these processes because the infrastructure does not need to change. In other words, SOA allows the business to drive changes in Information Technology (IT) rather than having IT limitations constrain the business.

SOA is not really a new concept. It can be thought of as an extension of technologies introduced to automate the Enterprise Architecture (EA). Moving from object orientation to service orientation seems to have moved the focus of EA technology one step further from relying on specific platforms, tools or implementation technologies.

The implementation of SOA within a large organization offers great hope of improved efficiencies, reduced redundancy, tactical agility and federation. For any organization, these benefits do not come without costs. This paper reports on research in progress intended to help organizations answer the question of whether SOA is right for them in light of what it costs to implement and what value the implementation might bring to the business. The first section of the paper describes what an SOA is and why organizations are considering SOA migrations. After this a detailed discussion is presented focused on the cost issues associated with deploying and maintaining SOA infrastructure. Cost estimating methodologies for various scenarios are presented along with a discussion of potential cost drivers within those scenarios.

SOA Definitions

The most important aspect of service oriented architecture is that it separates the service's implementation from its interface [2]. Through loose coupling and tight standards for interface, consumers need only know how to interact with a service. There is no need for them to know or understand what is under that service's covers. The building blocks of SOA include the service consumer, service provider, service registry and the service contract.

A **service** is a software implemented capability that is well-defined, self contained and does not depend on the context or state of other services [3].

A **service consumer** is a service, application or other software component that requires a specific service. The consumer locates the service through the service registry, accepts the terms of the contract and initiates the service using the mandated interface.

A **service provider** is the software entity that represents the service being delivered. The service provider makes the service contract available through the service registry and it accepts and executes requests for service that satisfy contractual criteria

A **service registry** is the network space where service providers publish service contracts and service consumers locate desired services

A **service contract** is the vehicle through which the service consumer and the service provider seal the deal. It specifies the rules of engagement as far as what the provider will supply, how the consumer will interface with the service and whether (or how) a particular consumer can be granted access to the service.

Cost Perspective for SOA

There are two dimensions from which to examine cost implications of an organization considering SOA. The first of these dimensions has to do with the fact that the introduction of SOA is an incremental process requiring an organization to reach maturity with their SOA practices. As organizations pursue this maturity, costs will shift from being applied at the project level to being applied at the enterprise level. The other dimension considers a categorization of recurring and non-recurring costs associated with various activities that may or may not be necessary for a particular SOA project.

Enterprise and Per Project Costs

From a cost perspective, one would expect costs to eventually shift from the project level to the enterprise level as an organization matures with SOA. This shift will not happen overnight and SOA projects in the early stages of adoption may end up bearing the brunt of costs which will eventually transition to an organizational level. In order to accurately estimate cost, risk and schedule of SOA projects, it is necessary to have an understanding of the cost structure for the projects being estimated. The details of the cost structure vary depending on two main factors – governance structure and the enterprises level of SOA maturity.

Governance Structure

A successful SOA program requires changes in the way that IT is organized and operates. This will differ depending on the type of enterprise adopting SOA. Factors such as size of the organization, geographical dispersion and organizational structure will dictate what governance structure will be most effective. Figure 1 depicts the four possible governance structures:

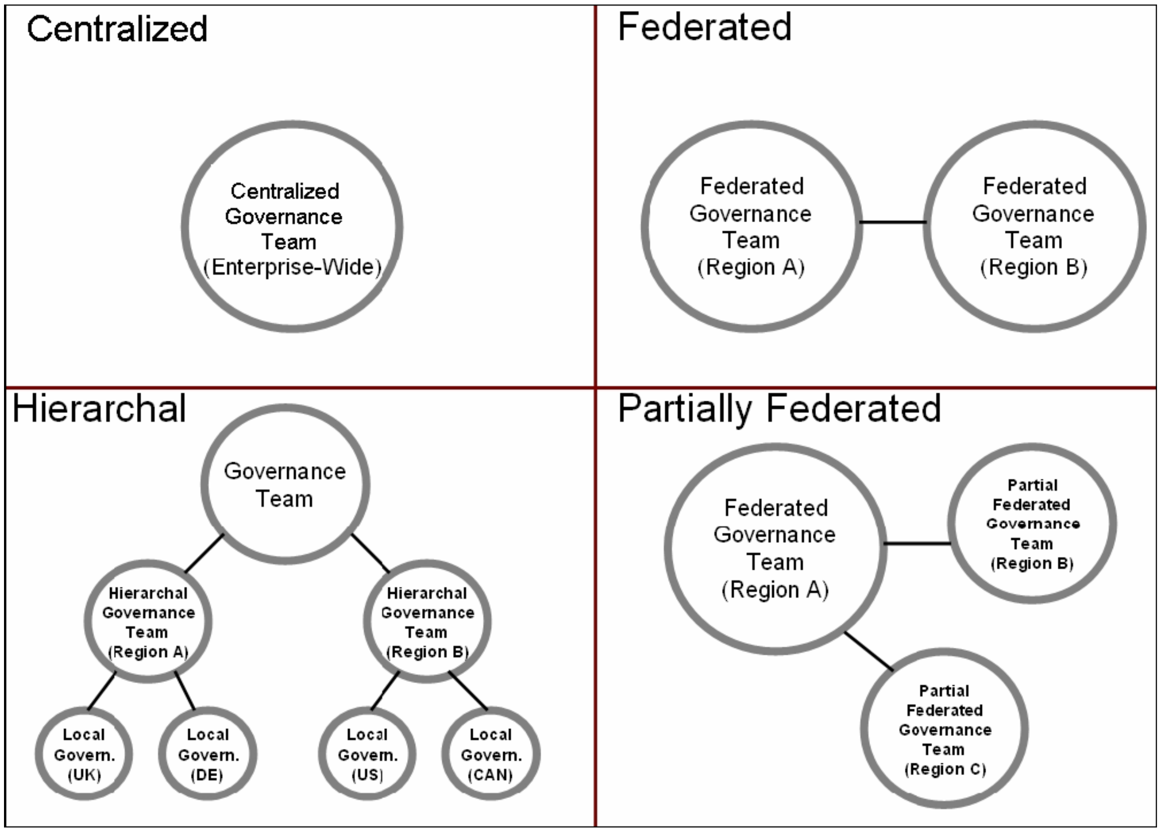
Centralized
Centralized Governance Team (Enterprise-Wide)
Federated
Federated Governance Team (Region A)
Federated Governance Team (Region B)
Hierarchal
Governance Team
Hierarchal Governance Team (Region A)
Hierarchal Governance Team (Region B)
Local Govern. (UK)
Local Govern. (DE)
Local Govern. (US)
Local Govern. (CAN)
Partially Federated
Federated Governance Team (Region A)
Partial Federated Governance Team (Region B)
Partial Federated Governance Team (Region C)

Figure 1: Potential Governance Structures [4]

Organizations that are smaller and have fewer locations will generally use a centralized governance structure. Organizations that have many separate but equal divisions or many geographic locations will need a federated governance structure. Extremely large organizations with wide geographic dispersion and a hierarchy of sub-organizations will generally need a hierarchical governance model.

While a governance structure is already in place in most organizations, time will be required to adopt SOA. This is an incremental process which can take many years from the initial stages until the infrastructure and governance processes reach a steady state, the organization has a sizable set of available services and the full value of SOA is being realized. The stage of SOA maturity an organization has attained can be gauged with the use of a SOA maturity model. At each stage the cost structure may change significantly, especially with respect to how cost element consumption is spread between the enterprise and the individual projects. In order to accurately estimate SOA project costs and benefits, it is necessary to understand the cost structure and how it varies at each stage of maturity.

SOA Maturity

Research into SOA maturity models [5] indicates different levels of maturity that have different implications on cost.

Early Stages (Level 1)

When SOA is first introduced, there is no SOA organization. SOA adoption and associated costs are interspersed at the discrete project level, driven by few isolated SOA-literate individuals.

Middle Stages (Level 2)

After some SOA successes and an organizational commitment to SOA, a “SOA Stakeholders Board” is put in place to define the SOA vision and strategy for adoption. At this stage, some of the planning costs move from the project level to the enterprise. Also, work is done toward creating an initial common infrastructure. As projects shift to this common infrastructure, integration activities move from the project to the enterprise as well.

Later Stage (Level 3)

Achieving this level allows SOA to scale and become tightly aligned with the business. There is a focus on realizing reuse goals and collecting metrics through monitoring capabilities so that SOA can yield higher levels of IT efficiency

While Level 2 focuses on achieving the technology benefits of SOA at an enterprise scale such as improved integrations and low-level service reuse, Level 3 promotes more business benefits by leveraging higher order architectural layers such as Business Process Management (BPM). Additionally, at this level monitoring includes not only technology metrics, but business metrics as well, providing insight into business Key Process Indicators (KPIs). All of these costs, which were once accrued at a project level, are now accrued by enterprise governance teams such as an “SOA Center of Excellence” and “SOA Program Management Group”

With such a focus on business alignment, strong business architecture skills are required to be able to model process, manage business service portfolios and understand performance metrics that are meaningful to the business. Service domain teams will generally assume ownership of defining business services and charting the service roadmap. These are SOA trained business and system analysts who manage service portfolios, service architecture and roadmap to produce optimal business-IT alignment. At this stage, costs to handle service versioning and change management issues are moved from project levels to the enterprise level.

Final Stage
After the infrastructure becomes steady in the later stages and the governance processes are established and institutionalized, some aspects of the governance activity at the enterprise level will experience a learning curve like effect. After they encounter and successfully handle many situations, they will become more and more efficient in technical decision making and program management.

SOA Project Components

Figure 2 portrays SOA from a more technical perspective. While full discourse on the functions of each of these components is outside the scope of this paper, a brief discussion of the SOA specific layers will help facilitate an understanding of how cost issues can be stratified (see [1] for more details on the technical aspects of SOA).

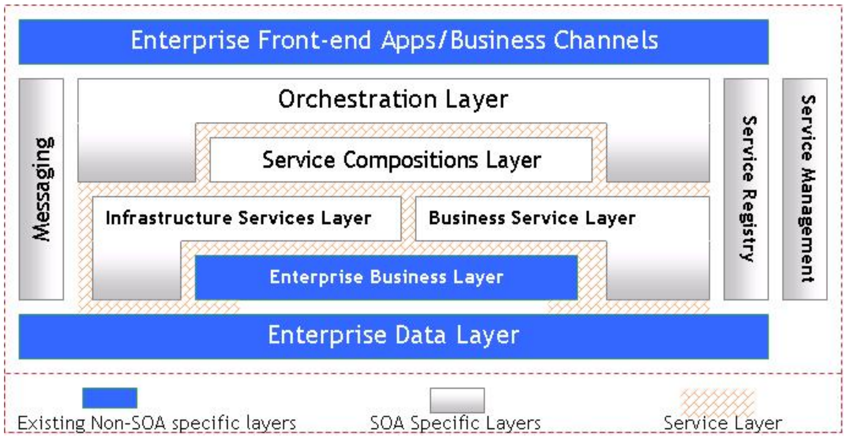
Enterprise Front-end Apps/Business Channels
Messaging
Orchestration Layer
Service Compositions Layer
Infrastructure Services Layer
Business Service Layer
Enterprise Business Layer
Service Registry
Service Management
Enterprise Data Layer
Existing Non-SOA specific layers
SOA Specific Layers
Service Layer

Figure 2: SOA Structure

The Business Service Layer consists of services that represent steps in the business process or mission thread. The Infrastructure Services carry out the functionality necessary to support successful use of the business' services, such as messaging, service discovery, and identity management. Business services implement the business rules of the organization. At the Orchestration layer business processes are implanted through the composition of various business services into applications.

For the purposes of this research into costs of SOA, the organizational migration to SOA has been divided into five specific types of SOA sub-projects. This division is a result of a study of various SOA projects in the Army and externally. Most SOA projects will contain components that are of one or more of these sub-project types. Activities necessary to accomplish each of these types of projects differ along with their associated costs. The five sub-project types are:

Prototype of SOA Infrastructure: This project type is the research and development phase during which an organization learns about SOA and determines how best to deploy SOA to meet their needs. During such a project the organization will determine what existing tools and infrastructure will support their SOA needs and what tools are available to meet SOA specific needs such as messaging, service management, and registry functions. Hardware and software will be procured to build a stack to implement the proposed SOA infrastructure and infrastructure services will be developed. At the end of this project the organization should have a working stack that will support some SOA functionality. They will probably have built some sort of small scale pilot applications to demonstrate capability to help sell the migration of SOA to the organization.

Production Instances of SOA Infrastructure: In larger organizations, one stack will be not enough to support all of their SOA needs. Additional instances of the stack will need to be deployed to support requirements for scalability, redundancy, performance, etc. Although the creation of each instance of the stack will be an integration effort in its own right, much of the research, experimentation, and integration expertise can be reused from the development of the prototype infrastructure. While this is still a ‘software development’ exercise, there is potentially enough reuse of knowledge that this can be thought of as a production rather than a development activity.

Development of Services: Services need to be developed to implement business rules. At first glance this sounds like simple software development but there is more to it than that. Most services are being developed for more than one stakeholder. And each of these stakeholders will have a slightly different view on the best way to deliver that service. Negotiating the solution that best meets all stakeholder needs will take additional time and effort when compared to a more traditional software development exercise. Services need to be reusable. They also need to be designed to not just meet the immediate need for capability or data;

they should be designed with some thought toward anticipating non intended uses of the capability or service being delivered.

Migration of Legacy Capability to Services: One of the most valuable aspects of SOA is that it allows organizations to leverage legacy applications by providing a new interface to existing capabilities. Because SOA stresses loose coupling, published interfaces and standard communication models, existing legacy systems can expose functionality as services, often without making significant changes to those systems. However, there are common migration issues encountered when migrating legacy capability to services. Analysis of each system is required to identify these issues at an early stage in the project. The Software Engineering Institute (SEI) at Carnegie Mellon University has done much research into service migration projects and has developed a technique to perform this analysis. This research leverages the SEI findings where possible. (See [6],[7] for more detail on SEI's work in this area)

Application Composition: Once there is an infrastructure in place and services have been developed or exposed, applications need to be composed to create capabilities that satisfy business process needs of the organization. Unlike traditional software development, application composition is more of an integration activity than a traditional software development activity. It also involves identification of available services as well as potential modifications to the infrastructure to meet new or emergent needs identified for the application being composed. Application composition requires different skill sets than more traditional software development projects. It can't be accomplished successfully without a clear understanding of the business rules that need to be implemented with the application.

SOA Cost Research Approach and Findings

The approach taken with this research has been to line up activities that occur in a traditional software development lifecycle (SDLC) with similar activities that would need to go on for the various flavors of SOA development projects. Using what history and data tells us about the costs of various activities, a gap analysis was performed to determine where and how activity costs for a SOA project might differ from the costs for similar activities with a more traditional software project. Following the gap analysis; research, data and experiential knowledge were applied to determine what factors were responsible for these gaps. Data collection continues both at the activity and cost driver level.

SOA Maturity Levels

As noted earlier, the level of SOA maturity within an organization not only drives costs for a SOA project but also determines where and when activities move from the project level to the enterprise level. For this research and as a result of data availability, SOA maturity has been stratified as follows:

- Emerging SOA – Organization is in it's infancy with SOA. At this level of maturity SOA is being piloted with one or more isolated project and there is much research and experimentation.
- Managed SOA – Organization has reached a level of maturity with SOA such that there is some enterprise support. Projects are still assuming some of the burden for activities that ultimately should be enterprise wide but there are is some support and institutionalization driven by the enterprise.
- Optimized SOA – SOA is entrenched in the organization, governance is in place and is being optimized

Emerging SOA

As SOA emerges in an organization each new project is a new adventure and required the IT staff to learn something new about SOA, the SOA technologies employed or processes specific to the business. Figure 3 shows an analysis of cost differences per activity as compared to more traditional software development lifecycle projects.

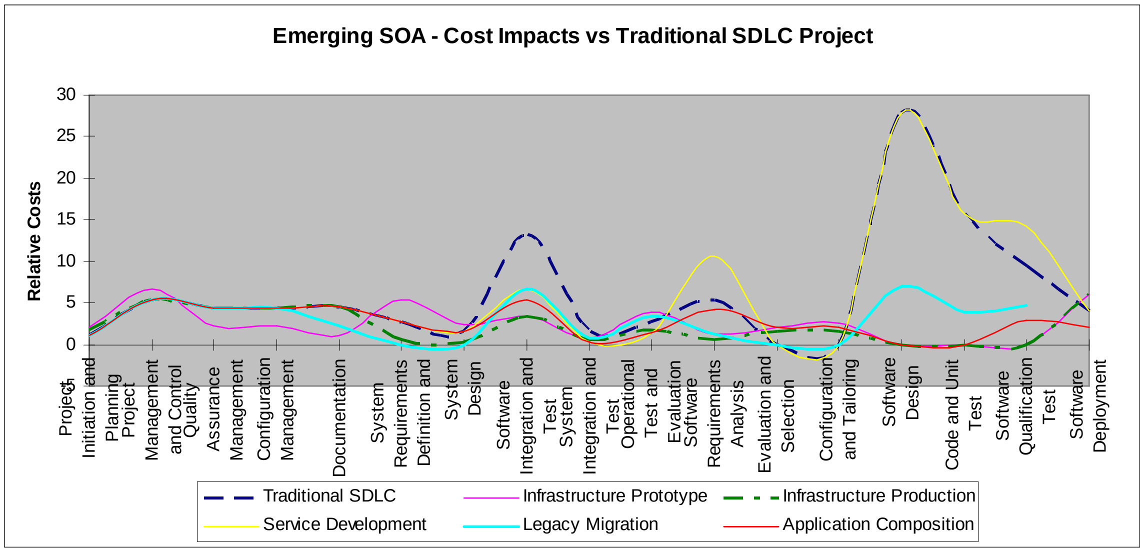

Figure 3: Emerging SOA Cost Impacts

Infrastructure Prototype

For prototypical development of infrastructure we find that there is a great deal of planning, requirements analysis, design exercises, evaluation of available technology, testing and tailoring. Activities associated with Configuration Management, Quality Assurance and Documentation have less import since the results of the project are not required to be of production quality. While the effort consists largely of integration and test type activities, the very nature of middleware leads to integrations that tend to be less time and effort consuming than as compared to more traditional software projects.

Infrastructure Production
At the emergent stage, production instances of the infrastructure continue to require significant oversight. Project Management, Configuration Management and Quality Assurance trend similar to any software development project, while the amount of effort required for planning the project stays high due to the still unfamiliar nature of SOA projects. Because most of the top level requirements and design can be learned from prototype exercises – the focus on requirement, design and integration and test activities is reduced. Since capabilities are still emerging, it is likely that more evaluation and selection will be necessary as new needs evolve.

Service Development
At early stages, the development of services should be treated as any software development project with respect to the overhead activities. Although as SOA matures much of this function will move from the project level to the enterprise, early on this is unlikely. Integration and test activities are light because these will primarily be realized as the services are composed into applications. Otherwise this is like a traditional software development with extra emphasis on software requirements and qualification to account for the fact that service design should account for intended and unintended uses of the service.

Legacy Migration
Legacy migration involves deploying existing capability with new or changed interfaces. This can basically be modeled like a reuse project where system level requirements and design can be eliminated since presumably this analysis has already been accomplished. Requirements, design and code activities are reduced since most of the functionality should already be codified, with the requirement being to develop glue code around this functionality. As with any reuse project, one of the biggest estimation challenges is properly sizing the functionality that is touched by the effort.

Additional factors that may impact the legacy migration effort include degree of invention, amount of legacy technology being carried along, communications and continuity between the development team and the migration team,

Application Composition
Overhead issues, at this stage, are comparable to any software development exercise but there is significant potential effort associated with finding and tailoring of services, with little or no software design, code

or unit test. Integration and test activities occur during application composition but integrations should be simplified significantly because service interfaces must adhere to a contract. At this point in an organization's SOA maturity it is likely that each time a new application is composed, there will be activities associated with tailoring the infrastructure components to meet new requirements introduced by the new applications requirements.

Additional factors that will impact cost for application composition include number of services available, data issues, granularity of services available and the ease with which the services can be located, skill level and experience with SOA, number of diverse stakeholders.

Managed SOA

When an organization reaches a point where SOA is fairly mature, there is a better understanding of both business processes and SOA technology. New projects should be more manageable, the proper skill sets should be developed and a good cache of services should be available and relatively easy to find. There should be more focus on planning at the enterprise level with many of the overhead costs being realized at the corporate level rather than on a per project basis. Figure 5 shows an analysis of cost difference per activity as compared to a more traditional software development project.

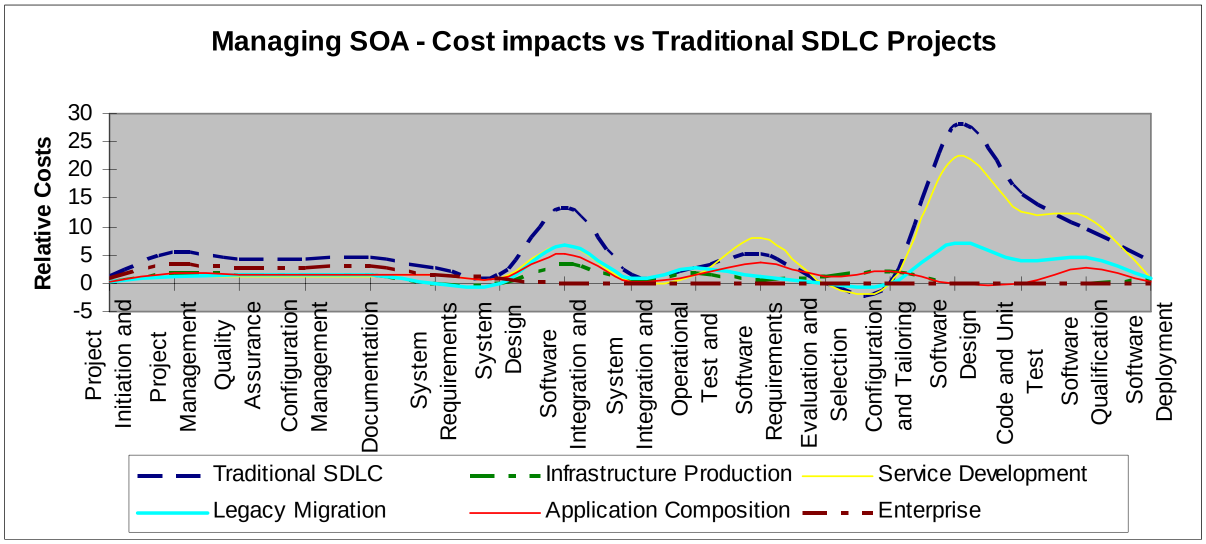

Figure 5: Managed SOA Cost Impacts

Infrastructure Prototype

An organization that has matured to the managed level would have enough experience with SOA and SOA technologies that prototypical efforts would be unlikely.

Infrastructure Production

At the managed stage, deployments of new instances of the stack, if necessary, should be fairly routine, requiring exploration commensurate with keeping any software up to date with respect to technology. Unlike the emerging stage, when projects tend to be stand-alone, the enterprise is now involved in managing many aspects of SOA so costs associated with overhead, management, planning and high level requirements shift from the project to the enterprise.

Service Development

Planning, oversight, management and high level requirements should be gradually shifting from the individual projects to the enterprise. Requirements analysis and testing activities for service development continue to be high to reflect the effort associated with designing for intended and unintended uses. As the organization's experience with SOA, on both a technical and business process level, grows there should be some reduction in effort on the software development and implementation activities for service development.

Legacy Migration

With increase in organizational maturity, costs are merely shifting from the project to the enterprise level for these types of projects. Otherwise, this continues to represent a typical software reuse effort.

Application Composition

This type of activity has become simplified as there is now a cadre of existing services to choose from when composing applications. It is less likely that composing new applications will result in additional tailoring of the infrastructure as the infrastructure should have attained a certain level of stability.

Optimized SOA

When an organization has fully matured with respect to SOA they have fully institutionalized their SOA processes and practices. All of their legacy applications have been abandoned, migrated, replaced or determined best not implemented as part of the SOA. A full cadre of services is available and accessible to support existing business processes. New service development is focused on changing business processes and emerging requirements. Planning, requirements, management and other overhead functions are almost exclusively handled at an enterprise level reducing per project costs. Figure 6 shows an analysis of the cost difference per activity as compared to more traditional software development projects.

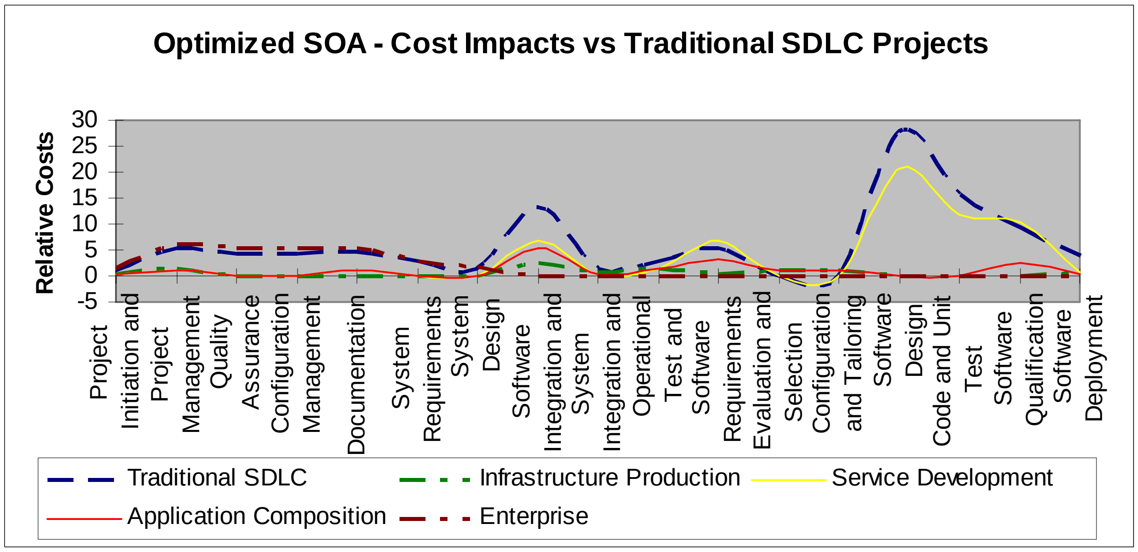

Figure 6: Optimized SOA Cost Impacts

Infrastructure Prototype
An organization that has matured to the optimized level would have enough experience with SOA and SOA technologies that prototypical efforts would be unlikely.

Infrastructure Production
With SOA institutionalized across the enterprise, organizational experience and defined processes should make the deployment of infrastructure instances a highly productive activity. While some project planning and management tasks still occur at the project level, most of the overhead functions are handled entirely at the enterprise level, causing enterprise costs to go up but per project costs to decrease.

Service Development
Overhead costs for service development will also shift almost entirely to the enterprise with the exception of some planning, project management and documentation activities. Increased understanding of SOA, highly experienced personnel and institutionalized processes will lead to some productivity improvement for the software development process associated with developing services.

Legacy Migration
At this stage it is safe to assume that all legacy capabilities that should be migrated have been migrated

Application Composition
Most of the overhead costs for this type of project have also shifted to the enterprise level. With a comprehensive service inventory and skilled SOA

developers, activities of evaluation, selection and tailoring are performed with optimal efficiency. Activities associated with integration, test and qualification are also performed with optimal efficiency.

Cost Drivers

As noted above, the level of SOA maturity and the types of SOA projects being undertaken both determine how one would approach estimating costs for a SOA projects. Additionally, there is a need to determine the 'size' and 'complexity' of each project as one would for any software estimate. For service development, legacy migration and application composition this sizing exercise is fairly straightforward and should rely on measures that relate to functionality. Determining the right 'size' for infrastructure and middleware components requires additional research. There are other factors that may also impact costs for a SOA project for which data collection and research are still on-going:

- Number of stakeholders: this drives the amount of communication and coordination required to complete projects. This impacts per projects costs as SOA emerges but ultimately drives only costs at the enterprise level.
- Organizational agility: this is a quantification of how well the organization responds to change. This issue applies to all types of projects in the emerging phase of SOA
- Level of SOA experiences: this quantifies how experienced or inexperienced the team of designers, architects, developers, systems engineers, business analysts, etc. are with SOA technologies. Extra time and cost will be associated with re-training IT employees to use SOA as well as the time associated with on-the-job training necessary to apply the training. With most organizations this will cease to be an issue over time.
- Amount and granularity of data: This quantifies the amount of information that needs to be dealt with and the granularity that applications require of that data. Since must of service orientation revolves around easy access to usable data the amount of data being dealt with can be a significant driver regardless of organizational maturity.
- Amount of existing Enterprise Architecture (EA) – The transition from EA to SOA is less traumatic than a similar transition with no EA in place. Having an EA in place suggests that thought has been given to centralization and coordination activities. This is only an issue in emergent stages.
- Sophistication and integration of middleware suites: The compatibility and ease of use of various middleware components drives the amount of time it takes for implementation team to come up to speed. There may be significant overlap with this driver and the level of SOA experience.
- Security concerns: Clearly the extent and types of security necessary will drive the costs of any SOA implementation at least as the infrastructure is assembled and deployed (See [9],[10] for more on SOA Security

concerns). This should cease to be a cost driving issue once SOA is mature except in cases where new security requirements emerge.

Conclusions and Next Steps

Service Oriented Architecture is emerging as the next generation of object oriented thinking. The difference with SOA is that technology has advanced enough to more fully support notions of encapsulation and abstraction and also to support the level of visibility of reusable services required to facilitate actual reuse. As with all paradigm shifts, there are cost consequences associated with the transition. And while SOA is too immature in the Aerospace and Defense industry to support any specific conclusions, industry experience and common sense indicate that properly deployed SOA, once mature, will result in increased agility with reduced costs.

Before these benefits are realized, organizations will need to transcend through various levels of SOA maturity. As SOA first emerges within the organization, per project costs will be increased and most of the SOA 'projects' will be self contained and not very SOA-like. As the organization learns more about SOA and accepts that the changes that SOA will introduce to the business will add value to the business, costs associated with SOA projects will shift from the project to some sort of SOA governance body at the enterprise level. At this point the per project costs will reduce because (a) overhead costs are handled at the enterprise and (b) loose coupling, well defined standards, and reusable services will make application composition significantly less expensive than new application development.

This paper reports on on-going research into the costs associated with SOA solutions. Conclusions reached to date are based on a limited data collection effort and require validation against a larger data set. Data collection and analysis is on-going to validate conclusions and further develop cost estimating relationships associated with potential cost drivers identified. Readers experienced with SOA projects with data they would like to share should contact the author at arlene.minkiewicz@pricesystems.com.

References

[1] McGovern, J, et. al., *Java Based Web Applications,* Elsevier Science, 2003.
[2] Shy, Cohen, "Ontology and Taxonomy of Services in a Service-Oriented Architecture, MSDN Library, April 2007, found at http://msdn.microsoft.com/en-us/library/bb491121.aspx
[3] BEA, "Domain Model for SOA, Realizing the Business Benefit of Service Oriented Architecture, Version 1.0
[4] Service-oriented architecture (SOA) Definition available at http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html
[5] Srikanth I., Sriram, A., "SOA Maturity Model", BP Trends, April 2007
[6] Lewis, G., et. al., "SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture", SEI Technical Reference CMU/SEI-2008-TN-008, June 2008
[7]Lewis,G., et.al., "SMART : The Service-Oriented Migration and Reuse Technique", September 2005, available from the SEI at http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn029.pdf
[8] Greenbaum, J., "Return on Investment for Composite Applications and Service Oriented Architectures: A Model for Financial Success and Enterprise Efficiency, 2006
[9]Chopra,d., "Security for SOA and Web Services", Dec 2004, available at https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/512de490-0201-0010-ffb4-8bd1620b2386
[10] NCES, "NCES Security Server", Defense Online, July 2006, available at http://ges.dod.mil/ServiceSecurity.htm