Succeeding with Component-Based Architecture in e-Government

Concept Level WHITE PAPER

Developed for the Federal Enterprise Architecture Program Management Office (FEA-PMO)

Industry Advisory Council (IAC)
Enterprise Architecture SIG

March 2003

Disclaimer

While the Federation of Government Information Processing Councils/Industry Advisory Council (FGIPC/IAC) has made every effort to present accurate and reliable information in this report, FGIPC/IAC does not endorse, approve or certify such information, nor does it guarantee the accuracy, completeness, efficacy, and timeliness or correct sequencing of such information. Use of such information is voluntary, and reliance on it should only be undertaken after an independent review of its accuracy, completeness, efficacy and timeliness. Reference herein to any specific commercial product, process or service by trade name, trademark, service mark, manufacturer or otherwise does not constitute or imply endorsement, recommendation or favoring by FGIPC/IAC.

FGIPC/IAC (including its employees and agents) assumes no responsibility for consequences resulting from the use of the information herein, or from use of the information obtained from any source referenced herein, or in any respect for the content of such information, including (but not limited to) errors or omissions, the accuracy or reasonableness of factual or scientific assumptions, studies or conclusions, the defamatory nature of statements, ownership of copyright or other intellectual property rights, and the violation of property, privacy or personal rights of others. FGIPC/IAC is not responsible for, and expressly disclaims all liability for, damages of any kind arising out of use, reference to or reliance on such information. No guarantees or warranties, including (but not limited to) any express or implied warranties of merchantability or fitness for a particular use or purpose, are made by FGIPC/IAC with respect to such information.

Credits

This paper is the result of work coordinated under the Enterprise Architecture (EA) Shared Interest Group (SIG) of the Industry Advisory Council (IAC).

Primary Authors

John C. Butler Unisys Corporation/IAC, John.Butler@unisys.com
David R. Mayo, Everware/IAC, david@everware.com
John Weiler, ICHnet.org /IAC, john@ICHnet.org .

Other Contributors

Susie Adams. Microsoft Corporation/IAC
John Dodd, CSC/IAC
Praveen Kosgi, Unisys Corporation/IAC
David Isaacs, Business Performance Systems/IAC
David Webber, XML Global/IAC

Table of Contents

Table of Figures

1. Introduction

1.1 Purpose

Industry's shift to Component-Based Architectures (CBA), a new enterprise architecture (EA) process for delivering applications, has fueled a tremendous amount of interest in the media over the past few years. With the search for the silver bullet that solves the continuing problems of integrating enterprise solutions as fervent as ever, IT organizations everywhere have jumped on the CBA bandwagon in hopes that it might finally ease the IT planning burden. As one might guess, it is not that simple. The purpose of this white paper is to provide a context for the rise of CBA, sort through the major issues, and provide guidance to the government business and technical managers so that sound business decisions can be made with respect to this key technology approach.

1.2 Scope of Paper

This paper briefly provides background on component-based architectures, discusses the business drivers that led to the rise of CBA, outlines the challenges and enablers of CBA, and provides some guidance on implementing CBA in government organizations. These issues will be touched at a high level. This paper is not meant as a comprehensive discussion of the various component technologies and the details of the processes that should be in place in order to build these systems with component-based architectures. The authors recommend the reader see the references provided below for further information on these topics.

1.3 Audience

This paper is intended as a vehicle to provide guidance on the implementation of component-based architectures to upper- and mid-level government executives with overall responsibility for the management of IT services with their organization. It is not intended to provide detailed insights to technologists or architects looking for a specific methodology for engaging this new architecture paradigm.

1.4 Referenced Documents

See Appendix B.

2. Succeeding with Component-based Architectures

2.1 Background

Component-Based Architectures represents a major shift in the IT industry from the traditional software development paradigm. Evolved from object management approaches, the component model enables a "plug and play", solution integration alternative to the custom-development oriented, "design, code, and test" development methodology. As the IT industry has transformed around this new computing model, it is incumbent on government IT organizations to transform their solution development life cycle processes to gain the promised benefits: shorter time to market, lower risk, modular and adaptive systems.

A software component in today's world is any piece of pre-written code that defines interfaces and can be called to provide the functionality that the component encapsulates. Components are typically packaged in "industry standard" ways so that they can be called from multiple languages, or from multiple environments.

Growth in component development and integration has been fueled over the past decade by a number of factors including a desire by organizations to achieve greater levels of reuse, the availability and popularity of component-based technologies, and increasing pressure to reduce time-to-market. These factors have combined to create a vibrant market in third party components that can be reused by organizations building applications for themselves and others.

For the most part, however, these components have generally been fairly fine grained in nature and/or highly specialized, such as GUI widgets or scientific libraries, and have been developed by outside organizations. Developers incorporate such components based on their knowledge of the component market or the components that come bundled with whatever integrated development environment (IDE) they happen to use. Comparatively little reuse of internally developed code has been seen within development organizations other than by developers using snippets of code they themselves had previously written for another project. Reuse of this type often introduces issues arising from version management, or lack thereof, of these snippets. Logic changes in the original code base are rarely propagated to the descendents, thereby allowing bugs to live on for some period of time.

At the other end of the spectrum, reuse has been sought in the form of full COTS applications. SAP, PeopleSoft, Seibel, and Oracle, among others, provide high-end business applications designed to support large portions of an organization's information needs. The approach to implementation of these packages differs significantly from that of fine-grained reuse discussed above. Generally, system integrators or integration teams have been responsible for these installations and have typically had a difficult time customizing them and creating seamless communication between them and other existing business applications.

2.2 CBA in the Context of the EA Lifecycle

Building an application architecture from components requires that the overall architecture be divided into "modules" of software with very well defined interfaces and that it be implemented using a compatible set of technologies to ensure interoperability. Identifying the appropriate decomposition is difficult at best and is confused further by the fact that problems can often be solved in a variety of ways. The goal is a solution that minimizes the impact of likely changes in the architecture and aligns with the direction of the organization while providing the best ROI.

INS Component-based EA

The U.S. Immigration and Naturalization Service (INS) has begun the process to develop and implement a component-based architecture. The INS incorporated CBA principles into their recently completed Enterprise Architecture Planning effort and has defined the set of services ("logical" components) needed to implement their business activities. The INS is currently in the process of identifying the necessary physical components along with the technologies and infrastructure modifications needed to implement these components. The INS EA Transition Plan ultimately results in a full-scale integration of components across the enterprise to fulfill their immigration mission. This EA Planning effort will provide the Department of Homeland Security a "jump start" in developing a component-based enterprise architecture.

The EA lifecycle is concerned with precisely these issues across the enterprise. Typically focused at the coarse grain level, EA looks at the overall business processes and the data, applications, and technology infrastructures that support them. For example, the context in which a COTS package will fit is defined at this level. But even further, this same business modeling activity can also be used to identify opportunities for finer-grained component reuse as the information needs of various organizations are analyzed. Often similar information needs surface across organizations and can be delivered in very similar ways.

In fact, history has shown that information technology evolves at a much greater rate than other factors that affect enterprise architecture. Business processes change, by comparison, more slowly and the information requirements even more slowly. From this we infer that investment in understanding the business and its information needs will pay dividends for a longer period than focusing on the technologies, which are likely to change in a relatively short period. For this reason, detailed analysis and documentation of the business architecture of an enterprise is built into the majority of EA frameworks including the FEAF [1], the TEAF [2], and the C4ISR Framework [3], and should be included in the process of defining and building a CBA.

[1] [FEAF99], [FEAFGuide01]
[2] [TEAF00]
[3] [C4ISR97]

Despite these issues, there is increasing experience and success with large-scale reuse of both in-house and third party components and applications. Organizations achieving this success have reaped significant benefits as described in the following section.

2.3 Business Drivers and Benefits

In recent years the necessary ingredients have become available to enable the vision of CBA. With the market acceptance of two of the leading enablers, the Java 2 Platform, Enterprise Edition (J2EE) and the ".NET" platform, standards-based technical platforms that support reusable components are becoming the norm. This is giving organizations the confidence to build the architectures needed to consume components from internal and external sources. These standards are also opening the door for independent software suppliers to provision components for public consumption. To broker the supply and demand for components, marketplaces, such as ComponentSource and Flashline, have emerged to meet the demand for application components.

The result is that organizations are now architecting their systems to consume components. Based on the architecture, they can determine how to acquire the components they need – purchase them from the marketplace, build them internally, harvest them from legacy applications, or commission their development by outside providers. Purchasing commercial components is much less expensive than building the functionality internally. Assembling components into applications takes less time than building systems from scratch. And, because publicly available components have been tested under real world conditions, the quality of the systems developed using the CBA approach is improved as well. This section discusses some of the main drivers for adopting CBA and the benefits from this approach.

2.3.1 Adaptable & Flexible Infrastructure

Substantial benefits of CBA are derived from the establishment of a common and interoperable information infrastructure upon which application components and services can be built. This results in an adaptive and resilient architecture that enables faster time to market of new capabilities together with a lower cost of operation. These lower costs are realized by eliminating the need for each program to develop its own set of infrastructure services, thereby reducing stove pipes while enhancing information sharing and interoperability.

Key Business Drivers and Benefits

- *Adaptable, flexible infrastructure,*
- *Consistent application of business rules,*
- *Time to market/mission fulfillment,*
- *Lifecycle cost and risk mitigation,*
- *Interoperability and information sharing, and*
- *Stakeholder alignment.*

By architecting an application as a set of layered components that expose services to higher layers and uses services of lower layers , the CBA approach creates applications that are inherently easier to understand and maintain. Further, by encapsulating

functionality, the "plug-and-play" nature of components makes it easier to replace functionality as business requirements change: unplug the old capability and plug in a new one.

This encapsulation also helps prevent application architecture degradation. Over time as systems are modified to reflect changing requirements, the architecture typically deteriorates. What may have started out as a straightforward and clean design becomes a tangled web. CBA minimizes this effect by providing a structure in which components can be replaced to upgrade the functionality without significantly affecting the rest of the system.

Further, the CBA approach creates an adaptable infrastructure (e.g., using Web Services and enterprise application integration – EAI) that facilitates the introduction of new channels for access and delivery. For example, wireless access, voice response/synthesis, system to system access (using XML), portals, and a multitude of other access methods are enabled. Architecting the user interface of an application (the presentation) as a separate layer from the control logic and the business functionality, allows new channels of access to be introduced without impacting the core functionality of the system. An ancillary benefit is that the potential life of the application is extended, further increasing the ROI of the system.

2.3.2 Consistent Application of Policy and Guidance

Government agencies are also finding that they operate based on business rules that are similar if not identical to those used by other agencies. Unfortunately, these similarities are not identified until too late in the process. Again, due to the lack of mature enterprise architecture processes, these rules are often implemented multiple times in slightly different forms, frustrating efforts to reuse the capabilities, integrate the systems in which they live, or more importantly apply the rules consistently across the enterprise. These almost parallel efforts drain critical resources in many ways including duplication of initial cost of development, ongoing system maintenance of multiple versions, and system integration.

Recognizing these issues, organizations are striving to establish enterprise architecture processes that will identify potential areas where capabilities can be shared and are establishing solution development life cycle (SDLC) processes to support component development and reuse.

2.3.3 Time to Market/Mission Fulfillment

Building a major application from scratch takes considerable funding and time, and also presents significant risks. Capabilities can be delivered more quickly through assembly of components than by building all of the functionality from scratch. Even in cases where components do not (yet) exist to meet all the functional requirements of an organization, development of the new components can be commissioned and proceed in parallel – again reducing the time to market.

In addition to the initial application development, CBA significantly aids the modification and enhancement of systems. In many cases, a component can simply be unplugged and replaced with a new component that meets the new requirements. Where coding changes are required, the modularity of the system makes it much easier to understand the logic and make changes to the appropriate components, again reducing time to market.

The improved time to market translates into being able to provide better system support for the agency's mission and to respond to changes in legislation, regulations, and executive orders. The result is better service to the citizens.

2.3.4 Lifecycle Cost and Risk Mitigation

In today's budget-constrained environment, CBA offers a practical way to reduce system lifecycle costs and implementation risks. Cost savings result not only from the ability to reuse components within a CBA, but also from reduced integration effort. The savings associated with reuse can be significant: even components that need to be modified slightly for a given application typically cost only 20% of what it would cost to develop the same component from scratch.[4] Most of these savings arise not from the reuse of the software *per se*, but from leveraging all of the requirements analysis, design, and testing effort expended to create it. Similarly, by working within the established framework of a CBA, the integration costs and risks are lower than with an *ad hoc* set of interface conventions created by each project.

Finally, we note that using proven components within a proven framework not only reduces costs, but also eliminates unknowns that drive technical, cost, and schedule risk. In this manner, risk and cost is more tightly confined to the new components being developed.

2.3.5 Interoperability and Information Sharing

One of the greatest challenges facing large, multi-discipline organizations is the failure of target systems to interoperate and effectively share critical information. Leading organizations around the world (governments, standards groups, technology suppliers and integrators) have struggled to achieve interoperability and information sharing goals. This struggle is evidenced by the number of government interoperability initiatives and the plethora of standards organizations created to address shortcomings. Recent evidence from well funded research efforts indicates that these interoperability and information sharing problems are not technology or standards problems, but problems associated with poor IT planning and EA processes. As it turns out, the interoperability and information sharing deficiencies that enterprise users are experiencing can, in fact, be avoided if the IT value chain and business stake holders can develop better "blueprints" that document key interfaces and information sources. Today's technologies, or building materials, are quite capable and can enable a high degree of integration and information

[4] [Poulin97], pp. 25-31.

sharing. The key is not in picking "the best technology" but having viable "blueprints" that translate business needs into technical solutions.

Thanks to a comprehensive study commissioned by the Department of Defense in 2000 (DepSecDef: Electronic Commerce Coalition Working Group), and an analysis of related industry studies, we find the answers to interoperability challenges to be within reach but not easy to attain. Root cause analysis of the problem indicates that the problems are in EA documentation methods and governance models[5].

Business process owners and CxOs must identify interoperability and information sharing needs up front. Analysis of these needs must be performed at the enterprise level – above the individual project level. Implementation approaches, systems interfaces, and integration points must be defined. Establishing a common and interoperable information infrastructure is a critical success factor that is hard to achieve with the current government funding models. The output of IT planning and EA processes must be modified to support component-based architecture and leverage a common library of components and specifications/interfaces.

2.3.6 IT Value Chain and Business Stakeholder Alignment

Getting stakeholders to come to agreement is often half the battle when it comes to building large systems. Agreement provides the foundation for marshalling resources and making the hard decisions that will inevitably come over the course of the project. But reaching consensus is not always easy.

The starting point for reaching agreement among stakeholders begins with a common understanding of the system to be developed and the process by which it is developed. The first provides the *what*, the second provides the *how*. CBA helps in both these dimensions.

From an IT value chain perspective, CBA forces clear specification of interfaces which enables suppliers and customers to agree on exactly what a component must do. Service Level Agreements (SLAs) and other agreements can be written around this agreed upon functionality.

From a Business stakeholder perspective, interface specification means clear agreement on responsibilities of systems that support different business units. Further, reuse of components across the enterprise means common application of business rules. Business units that push component reuse are generally better aligned operationally as well.

2.4 CBA Implementation Challenges

In light of the significant business drivers and benefits to CBA, one might wonder why the industry is not further down the road. The answer is that there remain significant cultural and process impediments to the adoption of CBA and these impediments

[5] These reports have been posted at www.ICHnet.org.

combine to create a significant barrier to organizations trying to implement a CBA program. The following sections describe each of these impediments in more detail.

2.4.1 Cultural and Bureaucratic Impediments

Software development has historically been an industry of creation rather than one of manufacturing. The flexibility of the medium allows developers to realize virtually anything they can imagine. Higher education institutions value experimentation and research over "engineering" when it comes to computer science and so students making the transition to employees bring with them the predisposition to create rather than reuse. Further, creating a new and innovative solution is more appealing than "brushing off" a dusty, old component and plugging it into a brand new application.

CBA Implementation Challenges

- *Cultural and bureaucratic impediments,*
- *EA, SDLC, and asset management process challenges,*
- *Ineffective budget and funding structure,*
- *Legacy technologies and infrastructure*
- *Inadequate skills mix, and*
- *Complex licensing and intellectual property issues.*

Over time the habits of developing from scratch have become engrained in the culture and bureaucracy of organizations. Only with a top-down and bottom up assault will the change occur. Executives must commit both the time and money to the change process. Training, tools, and incentives must be committed in order to make headway. Similarly, higher education institutions must provide a more reasonable balance between research and engineering so that the workforce is replenished with skilled engineers. Unfortunately, all this takes time. Organizations must make the commitment and be in the game for the long haul.

2.4.2 EA, SDLC and Asset Management Process Challenges

Agencies face significant process-oriented challenges to implementing CBA. These can be grouped into three major areas: Enterprise architecture process challenges, SDLC challenges, and asset management challenges.

Early Federal enterprise architecture processes do not include the steps necessary to create a component architecture. Although the FEAF and C4ISR Framework embrace component reuse as one of the eight principles, the process for implementing this guideline and for leveraging reusable components is lacking. As a result, agencies are on their own to create such a process, if they intend to create a component architecture.

Likewise, the primary systems development methodology in use by the Federal government is a waterfall process, although to some extent object-oriented approaches are in use and some functional requirements are met by purchasing commercial packages.

These SDLC processes, in general, do not incorporate the necessary steps to create a component architecture or to assemble and implement systems from components.

The processes and infrastructure are also missing in the Federal government to manage software assets produced by a component-based approach. The lack of a Federal component repository that can be easily searched for Federal off-the-shelf components is a major inhibitor to the adoption of CBA. If developers cannot easily locate relevant components, they typically build the (redundant) functionality themselves.

2.4.3 Legacy Technologies and Architecture

Unfortunately, the vast majority of legacy applications are stove-piped systems that do not lend themselves to interoperability with newer component technologies nor facilitate the establishment of a common "plug and play" infrastructure. Even though current implementation initiatives are leveraging COTS and component technologies, monolithic and isolated legacy applications present a significant impediment to the successful adoption and incorporation of CBA into the IT landscape. Evidence of these challenges has been noted in three separate study efforts: ICH sponsored ECCWG Software Quality and Interoperability WG (in partnership with IAC, AFCEA and NDIA), SEI's COTS initiative, and AF Scientific Advisory Board's April 2000 report on Challenges of Integrating Commercial Items into AF mission systems. The evidence points to failures in the architecture process and not in the viability of the technologies. These patterns of failure result from the inability of the EA process to adequately model the business and to adopt a set of interoperable solutions.

> *"Nothing is more difficult than to introduce a new order. Because the innovator has for enemies all those who have done well under the old conditions and lukewarm defenders in those who may do well under the new"*
>
> ***- The Prince, Niccolo Machiavelli, 1513 A.D.***

Stakeholders require the legacy systems in order to run their businesses and are unable to give them up until replacements are fielded. In the transition to a component-based architecture, stable legacy assets can often be wrapped with component facades to obtain a suite of services at a cost lower than for new development. During the transition, the new and legacy technologies will coexist and must be managed.

2.4.4 Skills Mix

Further complicating the issue, a new set of skills is required to realize reuse in general and CBA in particular. As discussed later in the document, the emphasis of the development process shifts to business modeling, process and application factoring, architecture, COTS evaluation, integration and testing. Architecture skills, the skills needed to produce the "blueprints" and "building codes" for systems, are lacking in most agencies. Organizations are currently focused on where, in their view, the "rubber meets the road" – coding. Particular attention is paid to implementation technologies to the detriment of other skills such as requirements, design and integration. Even more

"mature" organizations, while focused more closely on the entire SDLC process, typically do not place the appropriate amount of emphasis on the skills needed for reuse and CBA.

In the new paradigm, coding skills are giving way to component integration skills, although user interface coding is still required, along with development of the "glue code" that ties all of the pieces together. The change in the skills mix does not mean that the overall staffing level should be decreased, rather that the staff is able to produce more and higher quality products.

2.4.5 Budget and Funding Structure

CBA by design emphasizes shared capabilities that can be used by many applications. As stated above, however, this requires analysis and design across the organizations and applications that might reuse the components. Federal budgets and funding, on the other hand, are structured along the concept of a single application or mission. This disconnect makes it difficult to fund construction of the common infrastructure or common components that will eventually benefit multiple applications. The problem is evident within individual agencies, but is even more severe in a cross-agency context. Cross-cutting efforts such as the eGov initiatives, however, are beginning to pave the way for the kinds of cooperative efforts that will be essential to realizing the benefit of CBA.

2.4.6 Licensing and Intellectual Property

One of the key prerequisites for the successful implementation of CBA is that the architects selecting components for inclusion into an application architecture be confident of the performance of the components. This level of confidence goes well beyond technical performance to include such factors as quality (is the component "certified" and will the performance of the component degrade over time?), maintainability (who will fix it if there is a problem?), support (is there help desk support available to help sort things out?), warranty (will someone fix or replace it if something is wrong, and for how long?), and prospects for the continued existence of the "owner" of a component (what if the company goes out of business?).

From the component architect or developer's perspective, issues such as property rights and licensing are important, as well. For example, if a Federal contractor develops a component using their own technology, who owns it? If it is public domain, how can they be compensated for reusing it elsewhere? If component development is federally funded but contains proprietary components, how is it distributed for reuse? Web Services adds another dimension: Service Level Agreements (SLAs) to define availability, response, performance, etc. characteristics. Who do you negotiate the SLA with? Assessing all these factors brings us to the concept of intellectual property (IP) and licensing issues.

Software component licenses address many of these factors. However, there are still significant issues, particularly for large-grained, enterprise components. Commercially available components come with license agreements that spell out the terms discussed above and define who and under what circumstances the component can be used (e.g., single CPU, single site, department-wide, agency-wide, Federal government-wide, etc.).

Government produced components (GOTS) have other issues as well. First of all, the issue of ownership has to be resolved. Public domain components are "owned" by all, and, therefore, by no one. How is the confidence level of the architect assessing the component to be established? Sharing non-commercial components across organizational boundaries (including states, local governments, educational institutions, etc.) may require exercising intellectual property release processes that currently may be lengthy or cumbersome. Agencies may need to refine their contract language and intellectual property release processes before the full benefit of CBA can be realized. Agencies will have to assume a role of responsibility for components they have created and released.

A common approach to describing, managing and licensing components is needed across government. Architects searching for components need to know where the component comes from, who maintains it and how they can use it. If there is no license agreement that spells these things out, another mechanism must be developed.

2.5 Key Enablers

The road to effective CBA is long and as described above, riddled with potholes. Recent advances in the market have filled in some of these holes. Taking advantage of these enablers, described in more detail below, will smooth the road to adoption tremendously.

2.5.1 Component Technologies

With the market acceptance of component technologies such as Common Object Request Broker Architecture (CORBA), Common Object Model (COM), the Java 2 Platform, Enterprise Edition (J2EE), and the ".NET" platform, as well as EAI technologies such as Message Oriented Middleware (MOM), XML, and Web Services, standards-based technical platforms that support reusable, interoperable components are becoming the norm. This is giving organizations the confidence to build the architectures needed to consume components from internal and external sources. These standards are also opening the door for independent software suppliers to provision components for public consumption.

Key CBA Enablers

- *Component technologies,*
- *Commercially available components,*
- *Standards and best practices, and*
- *BRM as the starting point for Service Oriented Architecture (SOA).*

2.5.2 Commercially Available Components

Along with the new component technologies has come a wealth of commercially available components. Go online and search for software components and you will find a broad array of components of all sizes available from a variety of sources. Each year, we find a greater percent of the business model being supported by both large and fine grain COTS (and GOTS) components. The question is no longer if or when, but HOW one will apply this new paradigm.

The richness of the market includes all kinds of components: GUI components, mathematical components, supply chain components, workflow components,

infrastructure components and many more are available for purchase. Some component solutions are offered as bundled sets either as part of an application server such as WebSphere™ and WebLogic™, or as add-ons. Finding the right component is a different matter but the fact remains that there is a rich market waiting to be tapped.

2.5.3 Standards and Best Practices

Since the mid-90s, commercial best practices, along with a rich, standards driven market, have supported and evolved the CBA process and the underlying technologies. Leading IT organizations in finance, telecom, manufacturing and distribution have adopted this approach and have demonstrated patterns of success. Best practices captured from Delta, Air Products, GM, and Bank of America have identified a set of factors required to transform the SDLC process.

It is the lack of a common technical infrastructure that inhibited widespread adoption of CBA during the 90s, even though the component paradigm was in use by some organizations. One of the main reasons organizations have embraced CBA and supporting component technologies such as J2EE, .NET, and Web Services is the wealth of deployment environments available in the market and the wealth of services they offer. Due to industry-accepted interoperability standards, it matters less what component technology is chosen than that the standards are complied with. Modern application servers offer sophisticated transaction management, integration with TP Monitors, relational database management systems, systems management consoles, directory services, and email servers, as well as both horizontal and vertical scaling. Further, the relatively close adherence of these products to industry or de facto standards provides a level of risk reduction that was not available in the past.

Along with the standards come best practices. These practices are derived from years of experience in solving the problems currently facing the Federal agencies and are often presented as *patterns*. Business patterns, interoperability patterns, lower-level design patterns, and even coding idioms have been available for many years in published forms such as Martin Fowler's *Analysis Patterns*[6] or in the *Rational Unified Process*™ (RUP). They can be exploited to provide a jump-start for adopting CBA.

2.5.4 BRM as the Starting Point for Service Oriented Architecture

Initially published by the Federal Enterprise Architecture – Program Management Office (FEA-PMO) in 2002, the Business Reference Model defines business functions and sub-functions and relates these to the agencies that perform them. This high level business model is meant to provide the foundation for common understanding of business processes across the Federal government in a service oriented manner. It provides input to the EA lifecycle that enables each agency to define an enterprise architecture as mandated by Clinger-Cohen[7] based on a common business model. As such, it represents a significant step forward in reaching a common understanding as to how services are provided to citizens across the Federal government. Agencies that perform the same

[6] [Fowler96]
[7] [ClingerCohen96]

function can either share system support or can be reorganized to factor out the common function and thereby operate more efficiently. When coupled with a Service Oriented Architecture (SOA), agencies can directly trace component development to agency mission fulfillment.

2.6 CBA Critical Success Factors

The move to component-based architecture has many benefits and rewards for those agencies that successfully transform their organizations and processes to enable this paradigm shift. Successful transition can only be achieved, however, by paying close attention to the factors described in the following sections.

CBA Critical Success Factors

- *A service oriented logical architecture,*
- *A business driven approach,*
- *Organizational transformation,*
- *Revised solution development life cycle that supports CBA, and*
- *Effective software asset sharing and management*

2.6.1 Service Oriented Architecture

The shift to a services-oriented mindset is one of the most important, yet most difficult, factors in succeeding with CBA. It affects virtually all aspects of IT from business analysis to funding and procurement, as well as network architectures. Breaking the silo, isolated functionality way of thinking about applications is critical in adopting CBA. In fact, a services-based approach is most likely the only way to achieve the systems flexibility and adaptability required by government agencies. The manner in which the services approach is implemented is subject to some variation; however, the basic services approach is a fundamental requirement.

Industry experience and best practices lead us to three ingredients of the service-based approach: components (CBA), layered architectures, and architecting for interoperability. Implementing a services approach through a component-based architecture represents the most practical means of achieving the goals of IT. Components offer consistent ways of providing functionality that can be reused by implementing it in multiple, different applications or by invoking the functionality from a single location (as in web services).

For understandability and ease of use, the component architecture should be structured in layers[8]. This layered approach provides a separation of purpose and allows developers to specialize their skill sets, so that each developer does not have to be an expert at everything. At a minimum, the layers should include: presentation, workflow, business, common, infrastructure, and operating system. The presentation layer manages the user's interface with the system; the workflow layer controls the interaction of the clients and the services; the business layer provides services specific to the business application, the common services layer provides business functionality that is common to multiple business areas (such as currency conversion in financial applications); infrastructure level

[8] Described in both OSI's RM-ODP (Reference Model for Open Distributed Processing) and IEEE's 1471.

services include those necessary for the application to function properly (such as error/exception handling, audit trails, etc); and, operating system services include low level activities such as secure access to system resources and monitoring system performance.

The component architecture must also be designed for interoperability. Functionality should be consistent and non-redundant ("normalized") through a series of architecture templates. It is not necessary that components reside on the same technical platforms; they must simply adhere to the common standards and be able to operate as "good citizens" within the eco-system[9].

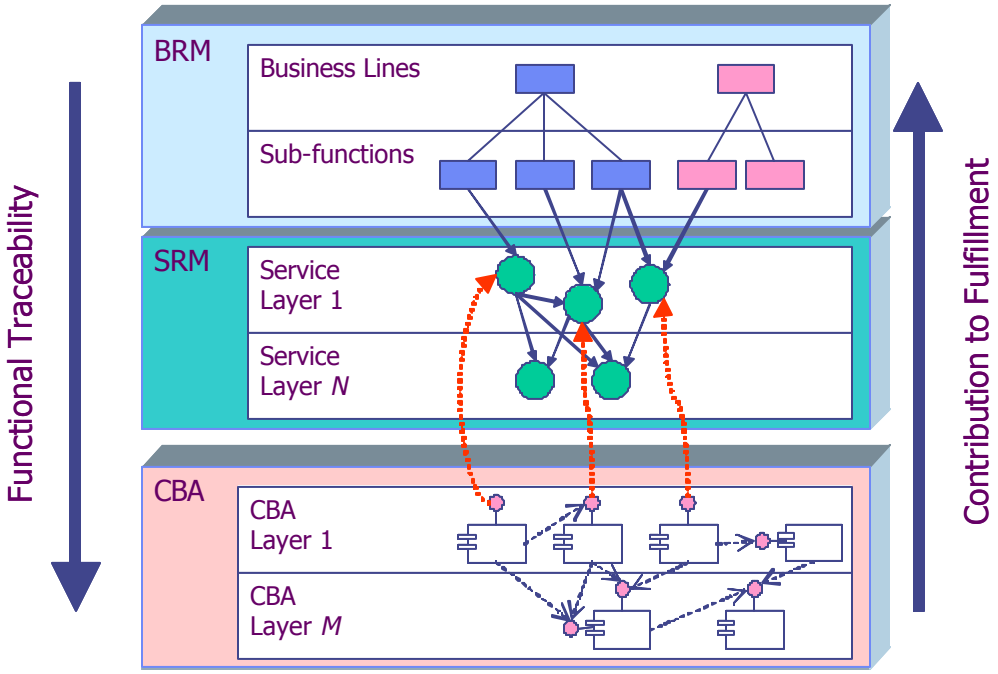

Figure 1 - CBA Realizes the SRM

2.6.2 Business Driven Approach

The business must drive the definition of the services in the SOA and subsequently be implemented in the CBA. This is the only way to ensure that components fulfill the agency mission and thereby add value. Once a business driven approach is established, justification of IT related initiatives becomes much clearer due to the direct traceability from the business need to the components that support that need as shown in Figure 1.

The starting point for this approach is the capture and definition of a business modeling. As discussed above, Agencies should use the BRM as input to their efforts so as to facilitate commonality across the Federal government. The process must go further and

[9] This approach is supported by a number of recent process innovations such as the OMG MDA and the ICH Business Alignment Process.

define the details of the strategies of the organization, the business drivers and the details of the processes that fulfill those needs. In doing so, activities and artifacts are identified and defined as part of a glossary of terms. These business entities or artifacts are used throughout an organization during the execution of one or more processes. At each step in a business process a worker changes an entity slightly or creates new entities from existing entities. By creating a layered architecture, components that manage the life cycle of business entities can be reused in many applications throughout the organization. Further, since these "business components" are implemented once and shared by all the applications, the organization reduces the risk that different business rules will be enforced in different parts of the organization. The same holds true for other aspects of the architecture, such as presentation logic and data management logic.

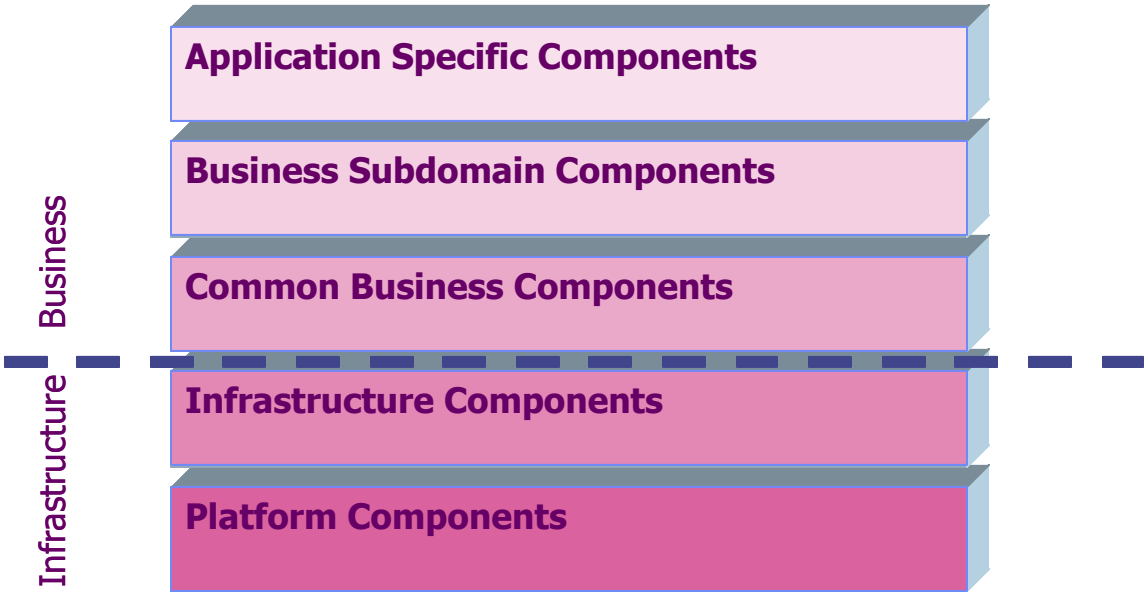

Figure 1 - Component Layering

2.6.3 Organizational Transformation

Another critical success factor in the adoption of CBA is a fundamental transformation of the way in which organizations carry out the IT process. Industry has sought to institute new processes that better align stakeholder views of the IT value chain. This desire to transform and mature the process to a more engineered and component assembly process such as that found in the manufacturing and construction industries, has been driven by the demand for more adaptable and flexible systems. The era of monolithic applications controlled by a centralized IT organization has given way to a decentralized framework in which the IT and business organizations partner to accomplish the overall mission.

The organizational transformation includes several aspects: process, structure, and approach. The approach to building applications that support the business mission must evolve to a services-oriented way of thinking. This involves both the business and IT

sides because of the close collaboration required between the two. Applications must be viewed as collections of services grouped together for development or usage convenience. The organizational processes must be altered to reflect the services approach and to encourage the reuse of existing services or components. The organization must be properly structured to facilitate this collaboration and the modified processes required.

New Roles and Responsibilities

Within the modified organizational structure, new roles, responsibilities and skill sets are required. These changes are most pronounced in four areas: leadership, business analysis, application development (SDLC), and asset management.

In the leadership area, the critical success factor is that executive management must emphatically support the new direction and require that all individuals and organizations get on board. As with any organizational change program, frequent and heartfelt communication of this support is mandatory.

The process for architecting applications with components relies on the foundation of a business model that reflects the current situation and future direction of the business. As a result, the role of business analyst becomes critical. The business analysts must work closely with the application and component architects to ensure that the component framework is kept up to date and adaptable for future needs.
The roles within the IT organization must adapt to the modified software development life cycle (SDLC). In particular, CBA puts additional emphasis on architecture, assessment and evaluation, as well as integration and testing skills. Architecture skills are paramount because a good application and component structure will deliver the benefits of flexible systems and improved time to market. The CBA process places less emphasis on coding functionality and more emphasis on evaluation of existing components to meet requirements and on the ability to integrate components to assemble applications.

2.6.4 Revised *Solution* Development Lifecycle Process

A critical success factor in establishing adaptive systems that can evolve with both changing requirements and emerging technologies is the adoption of a new software development life cycle that integrates CBA. The IT process must change from a software ***design, code and test*** approach to one of ***architect, acquire and assemble***. The key areas that are different from the current standard processes are iterative development, focus on architecture, and the services-based approach. The Clinger-Cohen Act has laid the ground work for changing these processes to better enable the adoption of commercial best practices by calling for changes in the technology process.

As discussed above, the fundamental change in the process is to treat applications as collections of services. The process must be built around this concept and provide the means of defining the services and exposing the services in legacy systems, as well as locating the services in commercially available components. The SDLC process developed by the FEA-PMO SAWG represents a constructive step in this direction.

Current EA efforts like the C4ISR Framework (DOD), and FEAF need to be expanded to incorporate a component mind set including a business reference model, validated component frameworks, interfaces/information sharing opportunities, and component "building-codes." Government and industry should collaborate in establishing these building code specifications based not on standards alone but taking into account emerging technology innovations and implementation/testing advances.

The second area for major process redesign is in the area of architecture. In CBA, architectures are developed at several levels. At the highest level, the Enterprise Architecture is created to align all of the business and IT drivers. At the next level, the Application Architecture is created to distribute the normalized services (i.e., non-redundant functionality) across applications. The applications consume components (or the services from them) and a Component Architecture is created that defines the structure of components and how they relate to each other.

The third area of process change is from a sequential (aka, waterfall) process to an iterative process. This concept applies at virtually all levels of the SDLC. For example, at the architecture level, the Application and Component Architectures are developed in conjunction with the search for and assessment of off-the-shelf (COTS and GOTS) components: existing components are factored into the architecture so that they can be taken advantage of. At the development level, the process of producing the application is separated from the process of producing the basic functionality (the "twin track" approach). The application is developed by consuming available components and is incrementally improved as enhanced components become available. The overall development process evolves into an acquire, assemble and test process.

2.6.5 Effective Software Asset Management

One of the key success factors in implementing CBA is the ease of locating, understanding, and using components. What is needed is a mechanism for sharing, selecting and managing components and associated COTS products. This is typically accomplished by establishing an industrial strength component repository.

A component repository is an application, with an associated database, designed to manage the inventory of components (and other software artifacts) available to developers. If developers have a difficult time finding a relevant component and testing it against their requirements, they simply build the functionality themselves.

Some progress has been made in providing shared information on COTS solution templates and component specification data. Upon completion of a DoD Wide study on how to implement Clinger-Cohen relative to COTS and E-Business (ECCWG), a joint government/industry initiative was established to better enable the mapping of common e-business needs to COTS solutions. This collaborative effort resulted in the formation of the Interoperability Clearinghouse (ICHnet.org), which established a number of architecture enablers including; Solution Architecture knowledge repository, series of architecture templates, and a formal COTS evaluation and selection methodology.

In recent years, a marketplace for components has emerged to broker the demand and supply of components. Companies such as ComponentSource and Flashline provide websites where developers and organizations can go to search for application components that meet their needs and purchase components on the spot. These companies and others offer their software as a repository or catalog that can be licensed for managing an organization's internal inventory of components. In addition, these applications are able to assist in managing the purchase and license arrangements for acquired components.

State governments have taken this approach in establishing the National Software Component Exchange (NSCE) under the auspices of the National Association of State CIOs (NASCIO). The NSCE consists of three layers: a commercial catalog of components for purchase, a national state government catalog of components available to state governments, and a state specific catalog where the components owned (acquired or developed) by a state reside. A developer can search the entire catalog for desired components and the results will be presented in order of: (1) the state already owns it, (2) there is a state-oriented component available, and (3) commercial components that can be purchased.

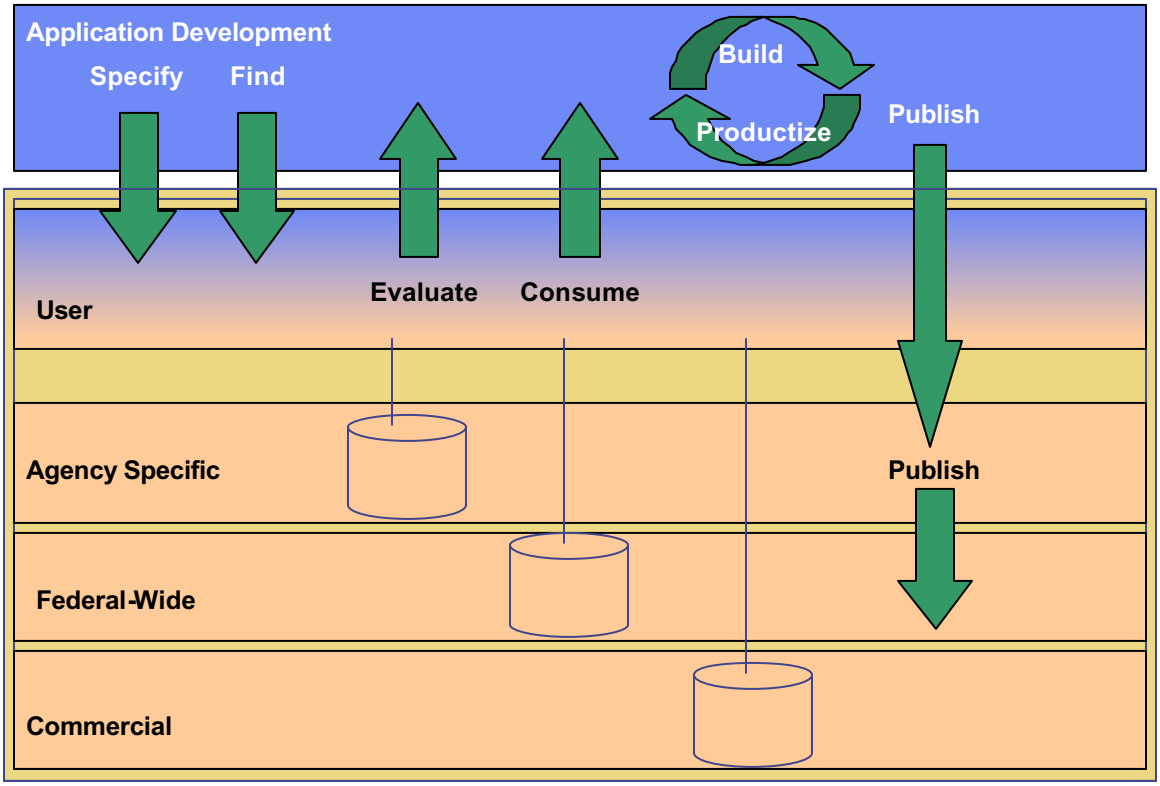

Figure 2 - Government Component Repository Needs

3. Summary and Recommendations for CBA

Commercial best practices relating to the successful adoption of CBA indicate a need for a major change in how solutions are architected and deployed. This new approach represents a major paradigm shift in how systems are viewed. As most of the technology providers have already re-engineered their products to support this new paradigm, it is incumbent upon the government leadership to embark upon a transformation journey. This section provides a set of recommendations that agencies must implement simultaneously to successfully implement a CBA approach. As OMB A-119 discouraged government from competing with industry, and encouraged agencies to leverage existing standards development efforts, we have attempted to identify existing investments and standards efforts that could be leveraged to meet OMB FEAPMO objectives.

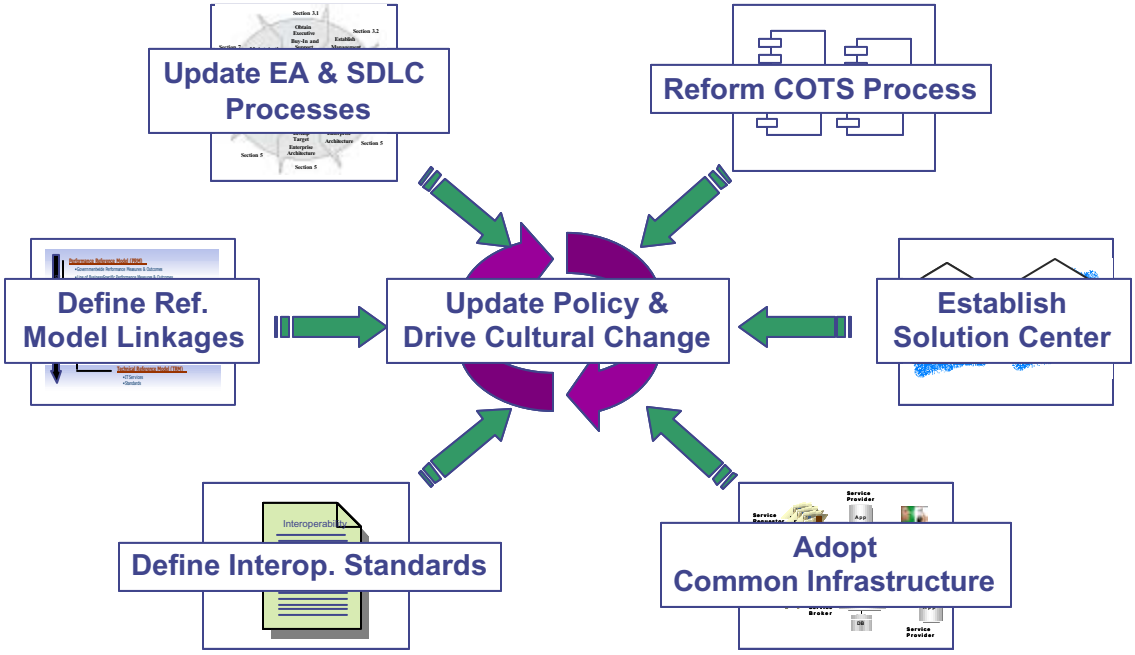

Figure 3 - CBA Implementation Recommendations

3.1 Revise EA and SDLC Processes

3.1.1 Enterprise Architecture Process

The primary enterprise architecture frameworks in use throughout the governmental agencies, such as FEAF, TEAF, and C4ISR Framework, focus primarily on the various models and artifacts that should be produced in order to fully define the enterprise. In fact, enterprise architecture is more a journey than a destination: the advantage is using

the process to align all of the factors that support the mission. The value is not so much in the products of EA, but in the usage of the products. An effective EA process also supports business process improvement of an agency's IT function and the overall agency mission

The topic of component architecture and reuse is rarely addressed in Federal EAs and, in those instances where it is, the references typically indicate that the technical architecture should support the reuse of software components or that the architectural artifacts themselves should be reused. Furthermore, IDEF, often used as the modeling notation for Federal EAs, does not adequately support component modeling and should be replaced by a more appropriate notation. In order to achieve the benefits of CBA described above, the current EA processes must be modified to integrate CBA concepts throughout. In particular, emerging EA processes must incorporate service-oriented architecture as a foundation concept in order to be able to align business drivers with technical solutions. The process must be driven by business objectives and derive the set of services necessary to achieve these business objectives. It is through this process that the overlap and redundancy of functionality will be identified and the component reuse opportunities will be exposed early in the EA lifecycle.

By looking across applications and even across agencies, applications can be "factored" in such a way as to significantly increase the level of reuse of software components including COTS, GOTS, and custom components within applications. Although business processes and the applications that support them can be refactored within an agency, even greater levels of reuse will be obtained if analysis and refactoring can occur *across departments or agencies*. Many of the functions performed within these organizations are similar, particularly administrative functions. Joint up front analysis and factoring across agencies will potentially enable the reuse of entire processes and thereby improve reuse across the government by an order of magnitude.

Recommendations for OMB : OMB's role in EA is to provide leadership, set policy, and supply a framework and templates for implementing EA within the agencies.

1. OMB should provide a forum for and establish policies encouraging business factoring across agencies. In essence, OMB is the guardian of the Federal-wide EA – the sum of the agencies' EAs, factored to identify commonality and remove redundancy. The closest thing we have to this is the set of reference models from the FEA-PMO. These should be further defined and built-out. For example, the BRM should be decomposed a further level and the other reference models (DRM, PRM, TRM, and SCM) should be completed. This would provide the context for the agencies' EAs and enable them to perform cross-agency factoring.

2. OMB should update and extend the FEAF to explicitly integrate CBA early in the EA lifecycle so that the opportunity to identify common functionality and requirements is maximized. The FEAF should also be revised to emphasize the linkage to the business drivers, so that the resulting EAs can be incrementally updated as the business objectives change (the EA must be kept in sync with the

business). This revision would help maintain current agency investments in the FEAF, while integrating the necessary CBA concepts into the process.

3. On an on-going basis, OMB should provide assistance to agencies to implement and improve the EA process.

Recommendations for Agencies: The agencies should adopt the guidance from OMB and should tailor the framework and templates from OMB to meet their requirements.

1. Agencies should update their EAs to reflect the guidance from OMB and should refine their EA process to an evolutionary process that focuses on the dynamic alignment of changing business and technical environments. The agencies should define the linkages of the EA to other management processes, such as program management, quality assurance, information assurance and security.

2. Update the EA models to include a service-oriented architecture as a mechanism to identify component opportunities.

3.1.2 Solution Development Lifecycle Process

Organizations looking to comply with Clinger-Cohen and OMB FEA-PMO guidance (move to CBA), must modify their current solution development life cycle (SDLC) process to accommodate the new emphasis on services and component consumption. In essence, as mentioned above in Section 2.6.4, the SDLC must evolve from one of ***design, code and test*** to one of ***architect, acquire and assemble***. The process must shift from a waterfall sequence of stages to an iterative set of steps with feedback loops and model refinements. The focus of the method is on identifying modules (services) and reuse at all stages in the process. That is, reuse is emphasized at each level, from business modeling (e.g., adopt business process patterns from other organizations with similar functions/processes) to component specifications and the use of components themselves.

The process must support an iterative development paradigm. The old way of building systems (large development efforts, followed by "big-bang" implementations) is too risky and is not sufficiently agile to meet the requirement for flexible systems. Under the new process, applications will evolve and grow, and over time, morph into what is needed. Each stage in the process will be approached iteratively, not just the component assembly stage. Models of the business will be developed iteratively, as will architectures, specifications, components, etc.

The process must promote collaboration along multiple dimensions, including business/IT, government/industry, intra- and inter-agency. The shift from the emphasis on coding permits a (better-placed) focus on understanding and modeling business requirements, specifying services and interfaces, improving the users' interaction with systems, and testing applications. The result is that the ***quality*** of systems improves in many ways.

Recommendations for OMB : OMB should establish policy that requires agencies to develop or acquire applications in a manner that promotes componentization and reuse across agencies.

1. Create a *solutions development framework* (SDF) that provides guidance to agencies on how to implement CBA and produce applications that consume components.

2. Assist agencies in implementing the SDF to create an agency-specific CBA-enabled SDLC.

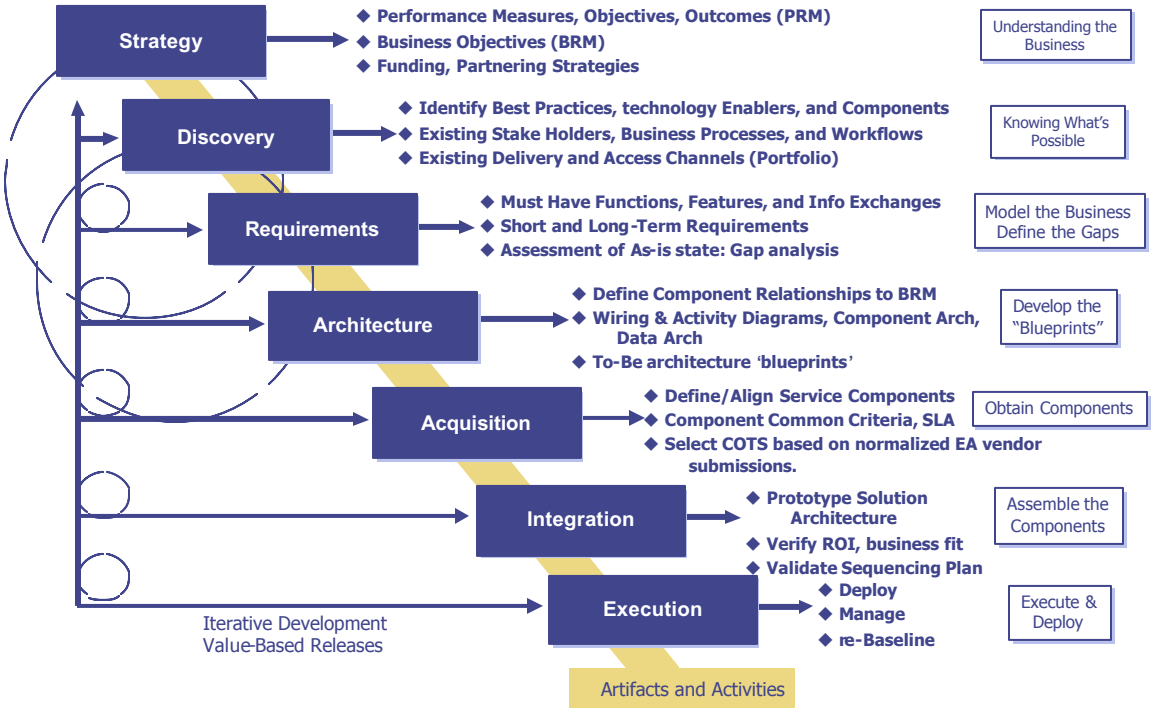

Figure 4 - Revised Solution Development Lifecycle Process

Recommendations for Agencies: Agencies should take the SDF and tailor it to their unique requirements within the guidelines that assure a level of process consistency across agencies.

1. Adopt and customize the SDF to produce a SDLC process that meets their requirements. The rollout of the SDLC process should be on an incremental basis, focusing on low cost, high impact areas first to establish a record of success for the new process, then build on that success.

2. Maintain and improve the SDLC process so that it evolves as the business and technology change.

3. Train staff on the revised SDLC and use it on new projects.

3.2 Define Reference Model Linkages

The FEA-PMO reference models (Performance, Business, Service/Component, Technical, and Data) represent the most significant vehicles for OMB to coordinate and communicate direction with the agencies. By publishing the reference models, the agencies will be able to align or map their efforts to the ***meta-framework*** of the Federal enterprise. However, in addition to completing (and evolving) the reference models themselves, OMB needs to spell out how the reference models relate to each other. For example, the SRM consists of services across the government. However, it is not clear how the services included in the SRM relate to the functions (sub-functions) of the BRM. In the same sense, the DRM should align with the BRM and the SRM. It is not clear whether the DRM will be developed in conjunction with the BRM (via parallel decomposition) and how the data elements correspond to the services in the SRM.

FEA Reference Models

The Federal Enterprise Architecture is augmented by a set of reference models that include:

- *Performance Reference Model,*
- *Business Reference Model,*
- *Service Component Reference Model,*
- *Data Reference Model, and*
- *Technical Reference Model.*

A description of these models can be found at:

http://www.feapmo.gov/fea.htm

Recommendations for OMB: OMB should define and publish the linkages between the reference models as a guide for the agencies to align their architectures to the models. This should spell out how OMB derives one model from another or at least how to trace from elements of one to the elements of the other. The Service Component Reference Model should be developed from an object perspective: the methods and the relevant data should be addressed. The data associated with each service should be mapped to the Data Reference Model. Defining the linkages will provide valuable guidance to the agencies in developing their architectures to conform to the reference models.

Recommendations for Agencies: As they develop their EAs, the agencies should create their business, service, technical, and data architectures using the reference models as a foundation. The agency architectures should map to the elements of the reference models and to the elements of the other agency architectures.

3.3 Establish Policies, Procedures, Technology Options for Interoperability and Information Sharing

Just as a building architect would not begin to develop a blueprint for a house without a set of "building codes," the architects of government information solutions require a set

of interoperability standards to produce applications that will work together and share information. One critical success factor for CBA is to establish, early on, a Technical Reference Model (TRM) including development and deployment technology standards as well as interoperable component standards. Compatible architectures and runtime environments must exist for CBA to succeed. Specifically, components are interoperable within the context of an application family (applications architected to share components – defined to use or consume services in a common manner). But this is not enough: the runtime, or deployment, environment must be based on interoperability standards, as well. Establishing these standards in the EA process allows organizations to build on them (as the building architect uses the building codes to create blueprints for custom houses) to develop component architected applications. Fortunately, industry has converged on a set of interoperability standards that the Federal government can leverage to accelerate this process.

Recommendations for OMB: OMB must provide the leadership to establish policies and procedures for component interoperability. These standards include: common semantics for services, data elements, and data values; common metadata (e.g., XML schemas); common architectural framework for plug and play; common technical platform(s); and so on. The forums for communicating these standards are the SRM and the TRM. OMB should build out these reference models in a collaborative manner, seeking among a sufficient number of agencies to establish a common basis, and expanding the definitions and standards as additional agencies adopt them. The SRM and TRM should be updated on a regular basis to incorporate new requirements and developments in technology.

Recommendations for Agencies: The Federal agencies should collaborate with OMB in developing component interoperability standards and provide test beds and pilot projects to demonstrate the viability of the approach, as well as collect metrics for business cases. The agencies need to incorporate these standards in their IT strategic plans and enterprise architectures and establish the technical platforms to support interoperability according to the TRM.

3.4 Reform COTS Selection and Evaluation Process

The current EA and SDLC processes in use through the Federal government, combined with the current A-300 IT procurement process do not provide the necessary elements for formulating a workable COTS identification, evaluation, selection, acquisition and sharing process. For this reason millions of dollars are wasted every year in duplicate evaluations of products by multiple Agencies. By establishing a common COTS evaluation and acquisition process for use by all Agencies in conjunction with a widely available repository for storing both the evaluations and the components themselves, duplicate effort can be reduced significantly.

A certification process must also be put in place that ensures consistency in process execution, application of criteria, and documentation of the results. This level of

consistency provides the necessary confidence in potential users of a COTS/GOTS, or internally developed needed to ensure reuse.

Recommendations for OMB : OMB's role in refining the COTS process is to provide leadership, set policy, and supply a framework and templates for implementing the process within the agencies.

1. OMB should create component evaluation and acquisition templates for use by the Agencies. These templates form the backbone a common process that will result in consistent evaluations of candidate COTS. The process should include establishment of a component certification process so that developers have confidence when (re)using a COTS or internally developed component.
2. OMB should establish a common component repository in which not only the executable can reside but also the other metadata associated with the component. The repository should include robust search and management functionality as well.
3. OMB should coordinate an activity to begin factoring functionality of business across Agencies. This effort would be an extension of the work being done on the BRM but at a much more detailed level. The goal would be to identify and/or prioritize potential reuse targets.
4. Once these are in place the OMB should begin looking at areas of commonality across the Agencies and begin to acquire or contract to build common components. As components are built or acquired, the OMB should certify that these components meet the evaluation criteria and or needs put forth as part of the acquisition process.
5. Finally, a centralized group such as the OMB should manage and maintain the component repository. This would provide a central location for the Agencies to deal with on component issues such as use of a particular COTS, SLA, and licensing options.

Recommendations for Agencies: Each Agency should collaborate with the OMB and other Agencies with respect to process definitions, adopt the guidance and standards provided by OMB, and then tailor processes and templates for use within the Agency. In specific the each Agency should:

1. Initiate component assembly *pathfinder* projects. These projects would provide experience within the Agency for integrating the COTS components into solutions that will support the Agencies mission.
2. Tailor component evaluation and acquisition process from the template. Using the standard template will help to foster consistency across Agencies and lend to the overall reuse of the evaluations.
3. Agencies should initiate projects to identify potential component harvesting opportunities. Often both end users and developers have a good feel for common functions within systems and can identify areas where the same functionality is being provided in more than one place. This "intuition" can be used to begin to harvest components from legacy systems.

4. Agencies should actively use the repository. This use is bidirectional. First, the repository should be used right up front during the EA process when evaluating how to factor the architecture of a system. Significant gains can be made by looking for potential reuse opportunities as early in the process as possible. Second, agencies should contribute components to the repository whenever possible – whether internally developed or acquired from third parties. This will help to grow the "critical mass" of the repository over time. And finally, agencies should evangelize the use of the repository. As usage rises, the benefit to all users grows as well.
5. Agencies should participate with the OMB and other Agencies in business factoring. This effort will increase the common understanding of the business models across the Federal government and help to prioritize the efforts to create or acquire commonly used components based on real needs.

3.5 Establish CBA Solution Center

CBA is a relatively new approach in the solutions development lifecycle, particularly for Federal government agencies. Early realization of the benefits from CBA requires that it be adopted rapidly by the agencies – on a small scale at first and rolling it out to major programs as the successes from the initial efforts are demonstrated. As with any new technology or approach, initial efforts will provide experience that leads to best practices and patterns that will be of use to subsequent efforts. A concerted, centralized effort is required to collect these results and share them with other interested parties.

A central function should be established that is responsible for coordinating the acquisition, testing, and certifying of components for use by the agencies. One of the major IT redundancies in the Federal government is the evaluation of COTS/GOTS products – many agencies perform the same evaluation for the same products. Even though each agency's requirements may be somewhat different, there is often a core set of functionality that is common. Allowing the evaluation of products for the common functionality to be centrally performed would benefit the agencies and improve the consistency of COTS evaluations. Another aspect of this centralized evaluation function should be the testing and certification of components. Because certification benefits all agencies, its cost should not have to be borne by the first agency to acquire a particular component.

One of the inhibitors to adopting components and sharing information is the consensus hurdle: It is difficult to get agencies to agree on common definitions, requirements, products, etc. Therefore, a neutral party should facilitate reaching consensus on the many areas necessary for successful adoption of CBA. Rather than attempting to achieve a global consensus, it should be developed among a few motivated agencies and the results provided to other agencies to adopt as they are able. This would not be a static solution. The definitions, specifications, and other artifacts would evolve over time to incorporate a broader perspective.

Recommendations for OMB: OMB should establish a CBA Solution Center. The CBA Solution Center should consist of three groups:

1. Process Center of Excellence – This group would focus on improving the CBA/SDLC process. Its role would be to collect, organize, and communicate CBA best practices, lessons learned, business process patterns, and other artifacts that can aid agencies as they launch CBA initiatives. In particular, the Process Center of Excellence would maintain and evolve the reference models and the linkages between them. This "knowledge management" function would most likely require automated support.

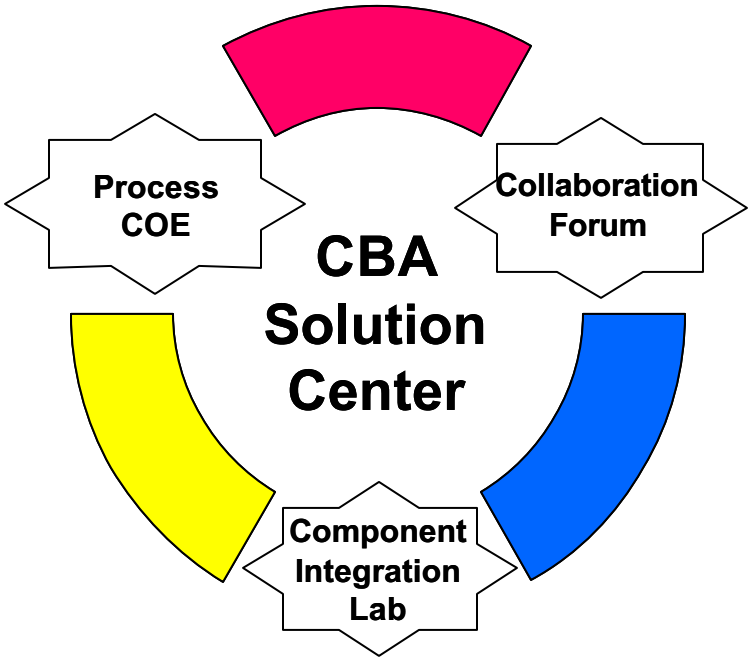

Figure 5 - Proposed CBA Solution Center

2. Component Integration Lab – The primary role of this organization would be to define COTS/GOTS evaluation process and perform evaluations for COTS/GOTS components with respect to common requirements. It would develop or commission the development of common components, particularly in the early phases, to jump start the adoption process and encourage agencies to consume components rather than (re)build the functionality they provide. An important role this group would play is in the certifying of components. A testing function is necessary and certification that the components perform as advertised is critical to instilling the confidence in the component for an application.

3. Collaboration Forum – This group facilitates establishment of consensus among agencies on business processes and data. The results of this collaboration will provide input into the SRM and the DRM: common processes and data definitions. As mentioned above, we recommend that OMB not attempt to bring all interested parties together and hope for the best, but rather bring together a few motivated

agencies with specific needs to derive an initial set of process, data and service definitions that other agencies can adopt as they evolve their EAs and architect applications. Again, start small and grow from success to success.

Recommendations for Agencies: The Federal agencies should contribute some of their most business-knowledgeable staff (on a detail basis) to participate in the definition of common services, processes and data. They should adopt and share best practices, business process patterns, common services and other products with the CBA Solution Center to facilitate their efforts.

3.6 Adopt Common CBA Infrastructure

As discussed in the section on enablers and critical success factors, the platforms to support CBA exist primarily in the form of J2EE, .Net, and Web Services technologies. The deployment platform is important, as is the interoperability incorporated into these technologies. However, in order to successfully implement CBA, agencies must also establish the technical infrastructure to support the new process. Like the technical platform, the supporting infrastructure must be interoperable to facilitate collaboration among all of the necessary participants. For example, it is not necessary that a single modeling tool be adopted by all agencies. However, the tools must incorporate the same modeling constructs (such as UML) and be able to exchange models with other tools.

Recommendations for OMB: OMB should define a set of compatible technologies that facilitate CBA and information sharing. These technology options should be documented in Release 1 of the TRM. The technologies that should be covered include runtime platforms, development environments, testing tools, registries, and system management consoles among others. The intent is not to select a single product in each category, but to provide a set of options that are known to work together to achieve the CBA objectives. OMB should establish a Federal component repository that facilitates locating and evaluating components that have been certified for Federal use. The repository should have areas to house agency specific components as well as Federal-wide components.

Recommendations for Agencies: The Federal agencies should develop plans to evolve their technical architectures to adhere to the TRM. They should rapidly establish a CBA technical platform, at least in a R&D environment, to initiate prototype or pathfinder projects that provide the basis for the development of internal standards. The agency EAs should define the roadmap and timeframes for migrating the existing infrastructure to a TRM-compatible configuration.

3.7 Drive Organizational Transformation

As discussed above, the cultural impediments to implementing a component-based architecture are among the most significant barriers. Overcoming them is, therefore, critical to the success of this endeavor. Implementing the process, architectural, institutional, and other changes, even with an extensive training program, will not produce the necessary shift in direction for an agency. Changing beliefs and behaviors is required and much more difficult.

Fortunately, one of the products of the business process reengineering (BPR) discipline of the past decade is a focus on managing organizational change. The critical success factors for engineering cultural change are well known and will not be covered here. For adopting CBA, the key recommendations in this area are as follows:

Recommendations for OMB: OMB must play a leadership role in guiding the transformations of the agencies' EA and SDLC processes. OMB should develop a "CBA maturity" program to provide policy guidance to the agencies. OMB should set goals and targets for agencies and monitor their progress against them (such as a "get to green" program). To assist agencies, OMB should create organizational transformation project templates for agencies to adopt and customize.

Recommendations for Agencies: Federal agencies should create and implement an organizational transformation program that accomplishes the following:

1. Obtain high-level sponsorship. Leadership is critical and the support for this change must be effectively and continuously communicated.
2. Obtain buy-in from the key stakeholders. Hold facilitated sessions to elicit their objectives and objections and ensure that they are addressed.
3. Treat this cultural change as an implementation project, with appropriate project management, schedule of tasks, clearly defined objectives and metrics to assure that they are achieved.
4. Implement a training and awareness program that allows the IT staff to be confident about the process and to build on successes.
5. Establish a incentive program that rewards individual support and progress toward the goals.
6. Start small within an overall plan and produce early and frequent success to establish and sustain the momentum of the program.

4. Conclusion

Large-scale gains in productivity can only come by moving to a development model wherein reusable assets account for the majority of the components of an application system. Incremental improvements in process and tools will only provide incremental productivity gains.

Evolution to component-based architecture has been hampered by many factors, both organizational and technical. Organizational factors include lack of coordinated domain analysis, entrenched cultural barriers, and disincentives to intra- and inter-agency sharing. Technical barriers, until recently, included lack of component repositories that support the development cycle with rich query and configuration management capabilities, and difficult to use component assembly and deployment mechanisms.

From a lifecycle point of view, successful adoption of component requires modification to the traditional architecture process so as to exposed COTS capabilities and solution frameworks early in the SDLC. Without a view of existing best practices and lessons learned, the cost and risk of component integration greatly increases. Consistent with existing guidance in Clinger-Cohen, OMB should encourage the retooling of current EA processes to better enable the use of components, adoption of best practices, and the inspection/validation of solution architecture prior to deployment. A small investment in improving the front end of the traditional IT lifecycle process will greatly reduce the risk of failure in the integration and deployment phase of the SDLC.

As evidenced in commercial industry, a key enabler of component based architectures is the establishment of a secure and interoperable information infrastructure that will support this "plug and play" paradigm. Many standards have evolved to enable a web based, component services infrastructure that has standards based interfaces for supporting a myriad of applications. This approach solves many problems:

- Greater interoperability and information sharing. By having one set of common "plumbing and wiring" interfaces, you establish a standard in which applications can plug into.
- Reduced redundancy of infrastructure investment. As application infrastructure represents a significant portion of the cost/effort of applications, creating a common, web services infrastructure provides those common services to serve most of the corporate applications (both new and legacy).
- Legacy Integration. A common infrastructure using APIs has proven to be an effective approach for legacy application integration. Legacy functionality can be exposed as services and fit into the evolving technical environment.

Solutions to these challenges exist. Component technologies are maturing, enabling service oriented architectures to be deployed as web services. Component repositories and development and configuration management tools are available to support the full life cycle. As new initiatives, process transformation and changes in agency culture

begin to tear down organizational barriers these new technologies and techniques will be leveraged to provide much higher levels of reuse and productivity going forward.

Appendix A – Glossary and Related Terms

The following are frequently used terms related to components and architecture.

Architecture: Representation of the structure of a system that describes the constituents of the system and how they interact with each other.

Application Architecture: Representation of an application and its parts, their inter-relationships and functions.

Application Family Architecture: Representation of a related group of applications, their inter-relationships and functions. Uses the standards and policies defined in the Enterprise Architecture to ensure consistency and interoperability across all components and applications.

Business Component: Component that offers business related services – applying business rules and accessing business data.

Business Logic Component: These components are those that offer small grained business logic that have a large degree of reuse throughout the organization. They include interfaces for services such as currency exchange, remarks, code tables, customer, and address.

Component: Independently deployable unit of software that exposes its functionality through a set of services accessed via well-defined interfaces. A component is based on a component standard and is described by a specification and has an implementation. Components can be assembled to create applications or larger-grained components.

Component Architecture: Internal structure of a component described in terms of partitioning and relationships between individual internal units.

Component-Based Architecture: An architecture process that enables the design of enterprise solutions using pre-manufactured components. The focus of the architecture may be a specific project or the entire enterprise. This architecture provides a plan of what needs to be built and an overview of what has been built already.

Component Repository: Application designed to store component specifications and implementations. Provides facilities to efficiently search for and retrieve components for evaluation against desired component specifications.

Enterprise Architecture: The meta-architecture of an organization, or the sum of all architectures within an organization.

Enterprise Component: A large-grain business component. Typically consume smaller-grained components. Examples include Customer Management, Case Tracking, etc.

Infrastructure Component: A software component that provides application functionality not related to business functionality, such as error/message handling, audit trails, or security.

Interface: Mechanism by which a component describes what it does and provides access to its services. Important because it represents the "contract" between the supplier of services and the consumer of the services.

Notional Component: Set of services packaged into a component, derived from requirements definition. A "desired" component – prior to implementation.

Reuse: Any use of a preexisting software artifact (component, specification, etc.) in a context different from that in which it was created.

Service: Discrete unit of functionality that can be requested (provided a set of pre-conditions is met), performs one or more operations (typically applying business rules and accessing a database), and returns a set of results to the requester. Completion of a service always leaves business and data integrity intact.

Service Oriented Architecture: Representation of a system in which the functionality is provided as a set of services that are called by other parts of the system.

Web Service: Functionality provided by a service, which is exposed using the Internet (XML, TCP/IP) as the transport mechanism. Can be internally provided as part of a suite of services or can be offered by external organizations.

Wrapping: Creation of an interface around legacy functionality (code) that exposes the functionality as services via interfaces that conform to a component specification.

Appendix B - References

[Blechar02] M. Blechar, "COM-17-0018: Rules for Playing in the Web Services Sandbox," Research Note, 24 June 2002. Gartner Research, 2002.

[C4ISR97] "C4ISR Architecture Framework, Version 2.0" C4ISR Architectures Working Group, December 1997.

[ClingerCohen96] "Clinger-Cohen Act of 1996" ("Information Technology Management Reform Act"), Publication L. 104-106, Division E, codified at 40 U.S.C. Chapter 25.

[ECCWG02] "Outbrief to the DEPSECDEF on Software Quality and Interoperability" Chaired by John Weiler, ICH. http://www.ichnet.org/fecc.htm

[Everware02] "Architecting and Assembling J2EE Applications: Best Practices," D. Mayo, D. Wasson, M. Russell; Everware, Inc. White Paper, August 2002.

[FEAF99] "Federal Enterprise Architecture Framework, Version 1.1," Chief Information Officer Council, September, 1999.

[FEAFGuide01] "A Practical Guide to the Federal Enterprise Architecture Framework, Version 1.0," Chief Information Officer Council, 2001.

[FEAPMO02] "Component-based Architecture," FEA-PMO, SAWG, October 2002.

[Fowler96] M. Fowler, *Analysis Patterns: Reusable Object Models.* Reading, MA, Addison Wesley, 1996.

[GAO02389T] "OMB Leadership Critical to Making Needed Enterprise Architecture and E-government Progress" Testimony by Randolph C. Hite and David L. McClure. United States General Accounting Office, March 21, 2002.

[IACEA02a] "Interoperability White Paper," Industry Advisory Council, EA SIG Draft, November 2002.

[ICH AAM 02] "Achieving Business-Aligned and Performance-Based Enterprise Architectures" M. Nelson, J. Weiler, R. Smith. Presented at the Spring 02 Federal CIO Council conference on Secure E-Business; www.SecurE-Biz.net

[Jacobson97] I. Jacobson, M. Griss, P. Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success.* New York, NY. ACM Press, 1997.

[Lim98] W. C. Lim, *Managing Software Reuse: A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components.* Upper Saddle River, NJ. Prentice Hall, 1998.

[OMB-A11-S300] "A11 Section 300 – Planning, Budgeting, Acquisition, and Management of Capital Assets," Office of Management and Budget, 2002.

[Poulin97] Jeffrey Poulin, *Measuring Software Reuse: Principles, Practices, and Economic Models*, Addison-Wesley, 1997.

[TEAF00] "Treasury Enterprise Architecture Framework, Version 1" Department of the Treasury, Chief Information Officer Council, July 2000.