



# *An Evaluation Of* SOA Frameworks

**Author:** Ramakrishna Raju

## OVERVIEW

Globalization, rapid business changes, technology advances that are flattening the world, the need to stimulate business innovation, ever growing challenges posed by Integration, external compliance drivers, pressures to increase efficiency by optimizing process and increasing interest in leveraging capabilities across organizational domains is causing many organizations to consider Services Oriented Architectures as an architectural style for their IT systems.

**The industry is buzzing with developments in this field**

Service Oriented Architecture (or SOA) perhaps needs no introduction. Much has been said and written about it in recent months. As SOA projects move from initial pilots to the organizational mainstream, a large number of SOA Frameworks, Methodologies and Models, are emerging to help enterprises navigate the complex path to a successful SOA implementation. SOA frameworks attempt to offer SOA practitioners structured approaches, so that they can leverage the growing body of real-world experience to charter a successful route to SOA.

The industry is buzzing with developments in this field. Existing standards bodies like OASIS, Open Group and OMG are actively deliberating new frameworks or modifications to existing frameworks to accommodate SOA. Analyst firms, IT consortiums, Vendors and even Governments are pitching in with their own SOA Reference Architectures, Models and Methodologies.

This Paper surveys the current state of the SOA Frameworks “marketplace” and examines a few of them to understand their potential and evaluate how such frameworks can help in expediting successful SOA adoption.

## SOA DEFINED

This Paper assumes an understanding of the basic concepts of SOA. However, before proceeding to the main task of exploring SOA Frameworks, it is appropriate to set down a formal working definition for SOA since that is central to this entire discussion.

There are many prevalent definitions of SOA. Some describe it more abstractly as a paradigm; a style or even a journey while others put forward more detailed and formal definitions. For this discussion, the definition by OASIS provides a good foundation. According to OASIS, *"SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations"*.

**If an architecture does not offer services effectively, (i.e. if its effects are not consistent with measurable pre-defined expectations), it is not an SOA**

This definition emphasizes some important characteristics for an SOA that are relevant to subsequent discussions:

- SOA is an architectural *paradigm* (or style) for organizing and utilizing Capabilities (or Services).
- Services are *distributed* under the control of different ownership domains - A fundamental goal of SOA is to ensure that capabilities in different ownership domains (and even organizational boundaries) can be shared and re-used effectively. If architecture cannot support such cross-domain sharing, it is not an SOA.
- "Capabilities" (or Services) must be provided *effectively*. If an architecture does not offer services effectively, (i.e. if its effects are not consistent with measurable pre-defined expectations), it is not an SOA.

## FRAMEWORKS DEFINED

**A Framework is a support structure in which a software project can be organized and developed.**

The term “Framework” has varying connotations in the IT Industry. While some frameworks operate at lower layers of the IT stack, helping in the effective development of software components (for example, the Java Struts Framework) others such as Zachman's Enterprise Architecture Framework operate at a much higher level of abstraction defining views of an Enterprise's data, function, networks, people, time, and organizational motivation. There are also frameworks that have specific focus – trying to address issues such as Governance, Compliance or Infrastructure Management.

In the context of Software Development, Wikipedia defines a Framework as a *“support structure in which a software project can be organized and developed”*. Throughout this Paper, the term “Framework” has this broad connotation – *Any* “support structure” that can help in the successful implementation of an IT System, and specifically Service Oriented Architecture is of interest, not just conventional Enterprise Architectural Frameworks. This could include SOA Reference Architectures, Models, and Methodologies, Techniques or other such approaches that could contribute to the development of an effective SOA adoption.

## SOA FRAMEWORKS

This paper explores developments in the following Frameworks, Methods, Architectures and Models, drawn from a wide variety of approaches proposed by Open Organizations, Analysts, Academia and Vendors.

**The goal of this Paper is to “sample” some of the many existing and emerging SOA frameworks**

- ***Service Component Architecture (SCA)*** - an open Reference Architecture proposed by OSOA, a consortium of leading industry SOA vendors.
- ***Service-Oriented Migration and Reuse Technique (SMART)*** – a technique developed by the Software Engineering Institute.
- ***The OASIS (draft) Methodology for SOA*** - which has been accepted by the OASIS SOA Adoptions Blueprints Technical Committee
- ***Compliance Oriented Architecture (COA)*** - an SOA based architecture for Compliance proposed by Redmonk, an analyst firm.
- ***The Open Group Architectural Framework (TOGAF)*** – An architectural framework proposed by the Open Group.
- ***Model Driven Architecture (MDA)*** - an approach developed by the Object Management Group, which is being enhanced to accommodate SOA.
- ***Oracle’s SOA Maturity Model*** – A model defined by Oracle, to map organizational “SOA maturity states”.
- ***IBM’s SOA Foundation*** – a lifecycle and Services modeling approach defined by IBM.

It should be noted that the above list of SOA frameworks, architectures, methodologies and models are not, in any way, a reflection of CSC’s specific interest in (or intent to adopt or otherwise promote) any one of them. The goal of this Paper is to “sample” some of the many existing and emerging SOA frameworks to get an insight about the activity and trends in this space.

## SERVICE COMPONENT ARCHITECTURE

Service Component Architecture (or SCA) is a set of specifications that describe a framework for building applications in an SOA. It is being developed by OSOA, a collaboration comprising many leading industry vendors. Presently OSOA offers version 0.9 of the Specification (the pre-release version) with the first full-fledged version of SCA scheduled for release in late March/early April 2007.

**Service Component Architecture (or SCA) is a set of specifications that describe a framework for building applications in an SOA**

### MOTIVATIONS

While some low-level standards like WSDL and SOAP have now become established, there is still a lot of flux in higher-level SOA standards. For example, there are still no pervasive standards that govern how services can be assembled, composed and deployed and how their QOS can be defined, etc. Artifacts produced during assembly or deployment is generated by vendor specific technologies and not portable across vendor toolsets or platforms.

SCA aims to fill some of those gaps. Specifically SCA targets two areas:

- Development of a common “language” via a graphical syntax that supports the assembly of coarse-grained components from finer-grained components.
- Development of standard bindings that separates the implementation of an SOA from the definition.

Vendors can use these specifications to build SOA Infrastructure and tools that are “standard”. Organizations using SCA frameworks will avoid integration issues and vendor locking when composing, assembling, deploying or defining QOS for services in a multi-language and multi-platform environment.

### GAINING MOMENTUM

Given its young age (OSOA created in late 2005), SCA already enjoys broad Industry support from 17 vendors including IBM, BEA, TIBCO, Oracle and many others (Microsoft is the missing notable). There is an active community feedback program into the development of the specification and major analysts seem positive about its future. Open Source has already started two major SCA projects - Apache Tuscany to support the SOA and Programming model in Java and C++, and under Eclipse, which is building a tool set for SCA.

### MODELING

SCA defines a programming model for service-based systems that includes three broad areas:

- Construction of Services
- Assembly of heterogeneous components into service networks
- Deployment of components into heterogeneous environments

Each of these is described in greater detail in the following sections.

## SERVICE CONSTRUCTION

SCA specifies Service Constructions with multiple programming languages including Java, C++, BPEL, PHP and COBOL and multiple containers like SPRING and EJB. (Microsoft C# and .NET are possibilities in the future). SCA not only supports creation of services using these languages but also introduces modern programming paradigms like Injection-based APIs, where the container itself can set attributes for the Class (unlike client based API's like JMS, JDBC and JAX-WS). Annotations support to declare SCA parameters is included.

## SERVICE ASSEMBLY

Assembly in SCA provides a language-independent way to connect services into "service networks". The hierarchical composition technique in SCA is used to expose service configuration end-points. It is similar to the concept of "Deployment descriptor" in J2EE allowing external configuration of Java, C++ classes, BPEL processes. Service dependencies can be modeled by "Wiring" - which is a modeling technique to connect Services to one another. Assembly also facilitates dynamic service configuration (including properties, protocols and qualities of service). During Assembly, Bindings can be specified for dependencies on externally deployed services (for example, one may bind a service to use JMS or SOAP). Such bindings are optional. If left unspecified the Container would decide a suitable binding. Also, if a binding is specified it may later be over-ridden at deployment time. During Assembly, Properties, Policies (example: Security) and Required Qualities of Service may also be specified. A note on Bindings - SCA bindings provide transport mechanisms primarily over Web Services using WSDL and SOAP but the specifications also support Java RMI, IIOP and REST.

**Assembly in SCA provides a language-independent way to connect services into "service networks"**

## SERVICE DEPLOYMENT

After assembly, components are deployed into an "SCA Domain". Each domain represents a region of configuration and administrative control. SCA run-time domains will typically be distributed and heterogeneous. Final overrides of configurations can be made at deployment time. Bindings and End point representations of external services is simplified when deployment is within one SCA Domain. In such cases, no binding is necessary (Container will default the binding).

## SERVICE INTERFACES

SCA supports standard Service Interfaces in WSDL 1.1 and WSDL 2.0. However, it is not necessary that each SCA Service will need to take the "WSDL translation hit". If two services are local (i.e. in the same service domain and in the same language), the specifications allow them to be defined as "Local" so that direct communication is possible. This allows fine-grained services that do not need to be loosely coupled to directly wire to one another (SOA Best Practices advocate Services that are Coarse Grained be loosely coupled but do not necessarily require this at a fine grained level)

## SCA POLICY

In SCA, each binding has “Policy Sets” for how that policy intent is achieved. For example, a binding may have WS-Policy assertions that configure “confidentiality” with encryption. Once this is defined, Service may require the capability by just naming it.

## SCA RUNTIME

SCA runtimes will typically have the following characteristics

- Light weight.
- Multiple implementation/ language support - Including Java, C++, BPEL, Query, XPDL, Proprietary ESB Proxies
- Strong abstraction from communication technology

**SCA is a fast rising star  
on the SOA horizon**

## THE BOTTOMLINE

Service Component Architecture provides a technology-neutral model that supports the creation, assembly and deployment of Services in an SOA. It fills in many gaps that exist in existing SOA modeling capabilities including the ability to compose and reuse services and the ability to create vendor tool neutral deployment artifacts. The full power of the Java majors is behind SCA – IBM, BEA and SUN to name a few. The Open Source Movement is excited about SCA with major efforts in incubation to conform to the SCA specifications. All this is indicative of the fact that SCA is a fast rising star on the SOA horizon and is here to stay.

However, there is one damper to SCA - Microsoft is conspicuous by its absence in the development of SCA. While SCA provides support for multiple languages and technologies, (including, at a future date, specifications possible around C# and .NET), most analysts predict that SCA will find wider adoption amongst either non-Microsoft oriented organizations or heterogeneous ones while Microsoft loyals will look to use Windows Communications Foundation (or WCF) to fulfill their need. Convergence between SCA and WCF is a possibility but at the present time, not a strong one.

## SERVICE-ORIENTED MIGRATION AND REUSE TECHNIQUE

Legacy System Modernization and Integration is a topic of keen interest to most organizations. Over the years, layer after layer of software additions in a disjointed manner has resulted in complex spaghetti of legacy systems in most organizations. SOA offers an attractive pathway to modernization by allowing legacy systems to expose complex business functionality through shared, re-usable and standards based services. The incremental approach that SOA offers is an added incentive to organizations compared to the expensive and often risky "Rip-And-Replace" approaches. However, SOA migrations of legacy systems are complex. Many factors need to be considered and the Software Engineering Institute has developed a technique to help address this complexity. The sections below will outline SEI's general views on SOA and drill down into more details about their proposed migration techniques.

### SOA AND SEI

SEI has developed a technique that focuses on the space where SOA intersects with Legacy Systems -defining an approach called Service Oriented Migration and Reuse Technique (or SMART) that develops migration strategies to modernize legacy systems to SOA.

**SEI has developed a technique that focuses on the space where SOA intersects with Legacy Systems**

SEI's SMART methodology has evolved from their work with the US Dept. of Defense. SMART considers the business objectives of the organization, its current systems portfolio and attempts to identify the costs, benefits and risks of migrating to SOA. It takes into account three different views on the migration challenge: The Application Developers' view, the Infrastructure Developers' view, and the Service Providers' view.

It looks closely at the "Wrapping" concepts that makes SOA such an appealing mechanism for organizations - the ability to modernize legacy systems "in-place" by exposing their core functionality as standard services, instead of the "Rip and Replace" approach which is usually risky and expensive.

### FOUR PILLARS OF SOA

SEI contends there are "Four Pillars" for SOA:

***Pillar 1: Strategic Alignment of the SOA and business*** - Business Goals should determine the type and nature of services, applications and infrastructure. Service Identification should be done through both a top down and a bottom-up (from existing legacy system inventory). Prioritization should be performed based on business goals and SOA migration works best when done incrementally starting with small pilots and expanding out to cover larger areas of business.

***Pillar 2: SOA Governance*** - A well-defined governance model is a critical success factor and needs to manage all aspects of an SOA including services discovery and design, change management, ownership of common service and data, versioning, monitoring policies and Service Level Agreements.

***Pillar 3: Technology Evaluation*** - A formal evaluation process that allows organizations to perform "hands-on" experiments with technologies before they are inserted into organizations is essential. SEI recommends a process called "TechChecks" to verify claims about different technologies and approaches. The approach consists of formulating hypotheses about the technology, and then examining these hypotheses against actual formal

experimentation. As the evaluation process unfolds, the hypotheses are either sustained or refuted.

**Pillar 4: Awareness of a Different Mindset** - SOA Development varies from conventional development in many ways, including the need to use approaches like business process modeling, the need to anticipate potential requirements from unknown users, the need to keep up with standards and technology stacks constantly, and the need to do additional testing coordination and impact analysis because services cross administrative boundaries.

## SERVICES

SEI's view of "Services" is that they are coarse-grained discoverable and self-contained software entities of re-usable functionality, citing examples such as "Customer Lookup" as a suitable granularity of service as opposed to finer grained functionality such as "Customer Address Lookup". Such services interact with other applications and services via a loosely coupled, synchronous or asynchronous, message-based SOA Infrastructure.

## MIGRATION GOALS

SEI contends that organizations should keep three major goals in view during an SOA Migration from a Legacy environment.

**Assembling new functionality from existing logic, thereby reducing costs is accomplished by exposing suitable parts of a legacy system as standard coarse-grained reusable Services**

**1. Leveraging legacy investments** - assembling new functionality from existing logic, thereby reducing costs. This is accomplished by exposing suitable parts of a legacy system as standard coarse-grained reusable Services. Doing this, transfers the burden of modernization into the SOA infrastructure rather than on the developers - allowing developers to easily expose complex parts of the functionality instead of grappling with re-writing it and also hiding the underlying diversity of the applications through the creation of standard interoperable sets of services

**2. Eliminating redundancies** - by improving efficiency. This is done by identifying, consolidating and re-using common services across applications.

**3. Adding agility** to the legacy systems enabling easy adaptation of systems to changing needs, because of standard and loosely coupled interfaces that communicate through the SOA infrastructure. Therefore, as long as the interface does not change, the underlying application logic supporting the services can change as much and as quickly as needed without the ripple effect that would be typically felt in traditional tightly coupled systems.

## COMMON ISSUES IN MIGRATIONS

Some of the commonly faced migration issues include:

**Lookup and Search** - While there is much talk of reuse, at a practical level, SOA Service Registries and Repositories are not always easy to search in order to locate the kind of services you need.

**Service composition** - can be more complex than it sounds, requiring transformations and additional glue code to pass information from one service to another. Management (aspects like Security, Qualities of Service, and Audit) also becomes more complex with a composed service.

**Dynamic or Runtime Binding** - is not easy and hasn't arrived technologically. Finding Services, Composing/Binding them at design time is not difficult but doing so dynamically at run time is difficult- for example - what happens to a dynamic binding if a particular service

within a composition of services, with a particular desired QOS is not available at run time - the consumer will need to provide a ranked order and exception handling and this needs to be supported by the registry for dynamic invocation. SEI believes that current technologies do not yet fully and effectively support runtime dynamic binding.

**Standards flux** - is still an issue - For example, SOAP 1.2 and 1.1 are different and there are scenarios where valid SOAP 1.1 messages may be rejected by a SOAP 1.2 service because it is missing required elements. More advanced standards (like Services composition) are still in flux. There is however gradual convergence.

## SMART

Service Oriented Migration and Reuse Technique (or SMART) is a systematic process to develop a migration strategy from a Legacy state to an SOA state. SMART gathers information about the migration context, potential services, existing legacy components, and the target SOA environment, to produce a service migration strategy as its end result.

**Service Oriented Migration and Reuse Technique (or SMART) is a systematic process to develop a migration strategy from a Legacy state to an SOA state**

It consists of five activities as described below, each divided into sub tasks. Information-gathering activities for the first three activities are directed by the Service Migration Interview Guide (or SMIG). The SMIG contains questions, analyzes the existing systems, targets architecture both from a high (architectural and design) level, and low (code) level and raises awareness concerning potential common issues that must be addressed in service migration efforts. It ensures coverage of the factors that influence the cost, effort, and risk involved in migration to services.

### STEPS OF SMART

**Step 1: Establish Migration Context** The goal of this activity is to understand the business and technical context for migration - goals, expectations, rationale, schedule and any budget, constraints. This stage also involves the Identification of stakeholders within the organization structure. Finally a set of initial candidate services for migration is identified. Following conventional business process modeling approaches can identify this list of candidate services and identifying common tasks in these processes as candidate services can follow this. At the end of this step, a decision must be made about the migration. Inability to identify a suitable set of candidate services would imply that the system is not suitable for migration.

**Step 2 Describe Existing Capability** This involves obtaining descriptive data about the components of the legacy system including a list of all the components, names, functionality descriptions, size in lines of code, language, operating platform, and age. Information is also gathered about the architecture, design paradigms, code complexity, and level of documentation, module coupling, interfaces and dependencies. Cost data for development and maintenance is also gathered. It is usually an iterative process and if needed, this step will be re-visited after the future steps to dig deeper into the analysis.

**Step 3: Describe Target SOA State** The goal of this step is to understand the requirements and constraints of the destination SOA and how they will potentially impact the Candidate services collected in prior Step. Service interactions and Technologies are studied. Current state of the infrastructure and expectations about the execution environment are understood. As an example, if the target SOA is very stringent on a particular standard (like security) or Quality of Service, and if you determine that your legacy component cannot meet those requirements - there is a potential gap.

**Step 4 Analyze the Gap** Estimate the gap between where you are and where you want to be and estimate the effort and cost needed to convert the legacy components into services. During this phase it may be needed to use code mining and architecture reconstruction tools to understand existing source code - this is especially useful in cases where there is inadequate documentation or it is not clear what the external dependencies are. It may also help to give the decision makers an idea about the complexity and state of the legacy system.

**Step 5: Develop Migration Strategy** In this step, a detailed migration strategy is developed that includes recommendations on wrapping or re-development of components, restructuring of the application into layers, and addition of supporting services.

The plan will typically: identify specific components to migrate, recommend a sequence based on priority, specify migration paths (wrapping vs. rewriting of code), suggest coordination with related efforts, such as SOA infrastructure or data architecture and will be rolled into two important documents: (a) a high level Migration Strategy Presentation for management and (b) Migration Strategy Report with detailed findings

**One of the interesting aspects of SMART is its use of a comprehensive questionnaire, in the form of an interview guide called the SMIG**

## THE BOTTOMLINE

In summary, SEI SMART analyzes the viability of migrating legacy components to SOA by systematically analyzing the existing systems, the target SOA environment and then proposing a migration strategy based on the analysis of gap between the two. It is not remarkably different from conventional architectural approaches to modernizing or integrating legacy systems, which advocate similar lines of activity (i.e. identify the “as-is” state, the “to-be” state and then analyze the gap). However one of the interesting aspects of SMART is its use of a comprehensive questionnaire, in the form of an interview guide. This questionnaire is based on a collective body of past migration experiences and offers a simple tool that migration teams can use to evaluate the potential costs, benefits, risks and impacts of migration to SOA. SEI seems to have put a good deal of thought in putting together this comprehensive set of interview questions based on its own SOA migration experiences with clients like the US DoD. The questionnaire however is not published openly and an organization would need to engage SEI (in a consultancy or advisory capacity) to access this information.

## OASIS (DRAFT) METHODOLOGY FOR SOA

In late 2005, the OASIS SOA Adoptions & Blueprints Technical Committee approved for consideration, a draft methodology for SOA. This is presently the only formal "methodology" for SOA under consideration by OASIS.

### OVERVIEW

This SOA methodology is fundamentally driven by the business domain, rather than the business function and presents a simple business driven lightweight "methodology" for SOA to derive the initial Business Services Architecture. It does not attempt to delve into details of Solutions or Implementations. It focuses on the most important aspect of the SOA - the *Services*. Its goal is the creation of a "Business-Driven SOA".

Traditionally, most IT driven methodologies concentrate very early in the development process on the "How" of implementation rather than the "What" or the "Why". This methodology focuses on the high level domain groupings of an organization to understand the core services that the business provides –closely examining its "What's" and "Why's". It applies the same concepts to either an Enterprise Level or to a Project level SOA effort.

**This SOA methodology is fundamentally driven by the business domain**

Driving service discovery on Process flows is explicitly avoided because

- Service discovery based on Processes tends to "drill-down" too quickly to a lower level of abstraction because it will drive the discovery into too much detail too quickly.
- Taking a process oriented approach will likely preserve "silo" based views of the organization and hinder the identification of common reusable services that could be leveraged across different domains of the organization.

Central to this methodology is the collaborative process that is involved in creating the SOA Level 0 and 1 maps at the Enterprise or Project level. Success is predicated on creating a broad sweeping consensus that breaks silos to achieve a common service vision for the organization or project domain that is being analyzed. Consensus is achieved by an initial Level 0 workshop that brings all "key" persons together and defining a high level (50,000 ft) view of the architecture. More detailed Level 1 flows are then derived in a top-down drill down by subgroups that are specialized in a particular domain in the organization.

### THE WHAT'S AND THE WHY'S

Like other similar approaches, the methodology focuses on the high level "What's" to derive services but one of the interesting aspects of the methodology is its focus on the "Why's" - to derive motivations of service interactions. While the "What's" map to the Services of the SOA the "Why's" are a way of integrating business motivation and drivers into the design of the Service Architecture model from the beginning, thereby imposing motivational constraints into the model and helping towards the alignment of the SOA with Business goals.

### EXAMPLES

An example cited in the methodology will make this distinction between the "What" and "Why" clearer and also help separate from the "How". Consider the classic organizational function of a finance department of a manufacturing business - Receiving and processing Purchase Requisitions.

From a business standpoint - The organizational function is "Finance". This is the "What". One of the business motivations, a "Why" is "Match requisition to delivery". The corresponding "How" is the actual physical handling of the transaction (i.e. complete a form with these fields, submit in duplicate, etc).

From the IT standpoint, when the above is mapped to Service Architecture, "Finance" (the What) becomes a Service. The Why's (for example, Submit Requisition(), LookupDeliveryStatus(), etc) will simply become the invocation points for the service and not services unto themselves.

Another example is Customer (the What), which is mapped to a Service called Customer. Its "Why's" such as "GetCustomerCreditRating()" or "UpdateCustomerStatus()" are the invocation points for the service and not in themselves separate services.

## TERMINOLOGY

Central to this methodology is a specific set of terminologies used for deriving an SOA. The terminology provides the common language for defining the SOA and includes the following:

**Central to this methodology is a specific set of terminologies used for deriving an SOA**

- 1. Level 0 view** - The "50,000 ft" view of the domain under investigation - For example, a view of a whole company. Rule of Thumb for Level 0 is that it will contain only between 1 and 5 core business services that collectively represent the core "What" of the organization and where each one of the services can be separated (for example out-sourced) without having significant impacts on the others. The Level 0 picture defines the overall context of the business and its primary services. This is the "Chairman and Board" view of the Enterprise. Examples of Enterprise Level 0 Services could include 5 services - Marketing & Sales, Finance, Manufacturing, Logistics & Warehouse, and Purchase.
- 2. Level 1 view** - Decomposed view of Level 0 into the next level of granularity. Elements at Level 1 show the "What's" (Services), "Who's" (external organizations/actors) and "Why's" (reasons and motivations that cause actors/services to interact). This is the CIO/CTO/CEO view of the enterprise. If (and only if) needed, the methodology suggests drilling down to a further level. This is only recommended for complex Enterprises or Projects and will not be the typical case. Most enterprises or projects will stop at Level 0 and 1 for the Enterprise. Then similar Level 0 and 1 maps are derived for each Project domain that correlates to the Enterprise Level 1 map.
- 3. Support Services** – Non-core (but still needed) business services. Examples of this would be an Auditing Service that tracks Compliance data and logs it.
- 4. Technical Services** - Non Business support services (Example: infrastructure and hosting support services)
- 5. Contract** - Formal definition that equates to a Service Level Agreement (SLA) for each Service.
- 6. Actors**- person, system or service - similar to actors in RUP / UML.

## STAGES

The methodology proposes a sequence that can help organize and sequence the Service Discovery and Design process in a few stages.

### Stage 0: Pre-work

The goals are:

1. Develop a broad understanding of the area for which the task is being under-taken. If undertaking the task at an enterprise level this should include the external drivers on the company as well as an understanding of the sector it is in. If it is a specific project, then it is key to understand the primary drivers for the project. The result is either case, is a clear understanding of the external drivers.
2. Identify the key stakeholders who are required to create the Level 0 and Level 1 service architecture
3. Develop a plan of events to which stakeholders will be brought in order to create the architecture.
4. Develop initial set of collaborative tools and artifacts for sharing the outputs of the work

Deliverables of this Stage are:

1. Statement of Sponsorship
2. List of Stakeholders
3. Event Planning - includes information gathering and defining Event Context Statement
4. Tools and Infrastructure Support to set up a collaborative space
5. This stage has a small initial team comprising the Project Manager, Lead Architect, Lead Business Analyst as the prime movers with support from administrative and infrastructure staff as needed.

**The important outcome is one single diagram that will convey to all stakeholders in an explicit manner what the Service Architecture is all about**

### Stage 1: "The Event"

The Event is the first set of meetings of all of the stakeholders to build consensus and common approach. An event facilitator who has a broad understanding of the business domain and can help to build the consensus coordinates the event. The event results in a Level 0 picture of the organization or program or project. The question "What" is used to identify services, "Who" is used to define actors (more significant at the corporate level), "Why" is used to understand the motivations of interactions between services,

The important outcome is one single diagram that will convey to all stakeholders in an explicit manner what the service architecture is about. Once this is done the services need to be refined more. To do this, identify the separate domains and split into break out groups to brainstorm and drill down deeper into each domain. The drill down is executed iteratively until all services are clearly defined. The draft OASIS methodology [x] outlines many suggestions about how this event can be organized and facilitated for the most successful outcome.

Deliverables of this Stage are:

1. Develop a Service Diagram - ideally on a single physical sheet of paper easy to read and share.
2. Define who the "Actors" are

3. Define Service Agent Interactions
4. Define Services and their interactions with one another
5. Develop a final report that summarizes this phase and outlines next steps. (For example in an Enterprise level effort, next step could be to define process maps between services).

### **Stage 2: Developing the complete architecture**

What has been accomplished thus far is a good starting point - the identification of the Level 0 and 1 Services. The service model has defined is the context, concept and goals of the architecture. It also describes the various different domains and the different business drivers that they have.

The authors of the methodology feel that beyond this point the SOA can be expanded at the Enterprise level or Project level as needed. Some guidelines on the next steps are provided but no attempt is made to go into these details. So this stage is only defined at a very high "guideline" level. The use of a more detailed and proven solutions oriented framework is urged as a complement to this methodology to help to carry the SOA to a successful implementation

**The methodology advocates a strong Change Management process to keep the views up-to-date**

The following suggestions are made:

- Focus within a project, is on understanding where these services "live" in the infrastructure of the business, and on "how" they are to be actually delivered to users.
- Evaluating vendor strategies and determining the best technical and architectural standards and practices for the organization to implement a full SOA that map to IT and business needs.
- Establish Governance including security constraints, performance requirements, redundancy and reliability.

Generally, the intent here is to leave the rest of the steps flow out as per the organization's best practices but the important point is to use the basic service architecture as the foundation at both the enterprise level and project level to deliver a true SOA.

This methodology is therefore building the Service Architecture focused on constructing a well founded "skeleton" onto which the complete architecture, whether Enterprise Level or Solution level, "adds its own layers of flesh, sinew and muscle".

It is expected that the definition of all the technical and some of the support, services is left until the detailed low level project architecture/design phases. It is also expected that during this stage, service understandings, in particular around shared services will cause impact and change to the Level 0 and 1 maps produced in earlier stages.

### **CHANGE MANAGEMENT**

The methodology advocates a strong Change Management process to keep the views up-to-date by instituting review processes that will ensure that significant changes are always reflected in the Level 0 and 1 maps so that they maintain a consistent view for all stakeholders at all times.

During the Service Design process, updates happen as decisions are taken about Services. Subsequently they would happen on a regular interval or whenever there is a significant business event (example: a Re-organization or Merger/Acquisition).

Change is generally easier to manage because the services map to the high-level business domains. Changes tend to be limited within those higher-level business boundaries.

**With all the IT and vendor hype about SOA today, this methodology provides a refreshing reminder that a clear understanding of Business Services is critical to an SOA initiative.**

#### THE BOTTOMLINE

This methodology focuses only on the beginning stages of an SOA - laying the foundation but this is also perhaps the most important phase. All the effort is focused on getting the Services right and making sure that everyone has a constant and consistent view of the Services. A high level Service Architecture that clearly maps the Services at all levels is the end product of a collaborative effort of all stakeholders. No attempt is made to go into the actual implementation strategy. With all the IT and vendor hype about SOA today, this methodology provides a refreshing reminder that a clear understanding of Business Services is critical to an SOA initiative. However, it is rather simplistic in its overall approach and assumptions focuses only on the very early parts of a typical SOA effort. It must be used in conjunction with a robust realization and governance strategy to make it more complete as a typical “methodology”.

# COMPLIANCE ORIENTED ARCHITECTURE

In recent times increasing pressures exerted by Compliance Regulations are causing businesses to re-think IT strategies. Compliance pressures may originate from multiple (and ever increasing) external regulations and mandates such as Sarbanes-Oxley (SOX), Health Insurance Portability and Accountability Act (HIPAA) or SEC 17a. However, there are various compliance needs arising internally as well externally such as Six Sigma initiatives that need to be managed.

## COMPLIANCE => AGILITY

Most regulations require, by their very nature, constant ongoing efforts to ensure compliance at multiple levels of the organization. Government regulations require organizations to remain vigilant and up-to-date, constantly adjusting and paying attention to the way business is done. Fundamental to meeting the requirement of efficient compliance is the need for IT agility - IT Systems that are able to respond quickly to changes in compliance requirements.

**Fundamental to meeting the business driver of efficient compliance is the need for IT systems to be agile.**

## COA

Redmonk, a US based analyst firm proposed a SOA Framework to help organizations focus on delivering agile systems that can meet compliance challenges. This framework, called "Compliance Oriented Architecture" (or COA) has been adopted in some organizations (including the likes of EMC and Geisinger Health).

## BACKGROUND

Traditional approaches to compliance have been more Project-oriented. The main issue with this approach is that even though compliance based initiatives are typically integration challenges requiring modifications to many existing legacy systems, each compliance effort if tackled like a project, will operate in separate project silos and may fail to effectively leverage already existing IT compliance capabilities or IT assets. It is all too common to find the same organization with two separate efforts concurrently underway, for example with SOX compliance and Basel II compliance efforts going on concurrently, with little or no leverage happening between the two. This not only leads to more effort and expense but this approach will also add more layers of complexity to an already brittle IT System making future compliance even more challenging.

## A DIFFERENT APPROACH

COA proposes a shift in this traditional project based approach, instead encouraging organizations to use an IT architectural paradigm across the organization (specifically SOA) to address compliance needs more effectively. COA is an SOA framework that makes the linkages between IT and Business controls more explicit. Many compliance requirements can be implemented as a set of core services. According to Redmonk - "COAs apply the virtues of Service Oriented Architectures (SOAs) to the specific business challenge of compliance, and the result is a flexible architecture that can meet compliance challenges now and in the future".

## COMMON COMPLIANCE SERVICES

COA is centered on the identification of “common business compliance services” – services that are common to multiple compliance regulations and then implementing them in a SOA so that multiple consumers across the organizations can leverage the same set of common compliance services.

An example of a common business compliance service is a “Record Retention Service.” Almost every Regulation requires record retention in varying degrees. Other examples could include:

**COA is centered on the identification of “common business compliance services”**

- Access Control Service
- Analytics Service
- Archive & Backup Service
- Audit and Logging Service
- Record Recovery Service

*(For a larger list refer to the Redmonk’s White Paper on Compliance Oriented Architecture listed in the Reference Section)*

## BOTTOMLINE

The COA approach could help organization deal with Compliance related needs by leveraging existing systems in an SOA, instead of traditional "Rip and Replace" project based approaches to compliance. This makes the organization overall compliance efforts more coordinated and efficient. Even though it hasn't entered the "mainstream" there are some valuable insights that can be gleaned from this framework to help any organization concerned with managing IT systems effectively to deal with compliance

## THE OPEN GROUP, TOGAF AND SOA

TOGAF (The Open Group Architecture Framework) is an approach developed by the Open Group, a leading vendor-neutral standards organization. It can be used to develop a broad range of different IT architectures including Service Oriented Architecture. It is not in itself architecture but provides the framework to develop one. TOGAF has been in use for many years before SOA arrived on the center stage. The sections below first present a high level view of TOGAF and then go on to discuss current activities in the Open Group around SOA and TOGAF.

### MOTIVATIONS

The Open Group believes that using an architectural framework like TOGAF will speed up and simplify architecture development, ensure more complete coverage of the designed solution, and make certain that the architecture selected allows for future growth in response to the needs of the business. The primary benefit of using TOGAF is that because it is vendor, technology and tool neutral, it enables IT users to build genuinely open and flexible systems-based solutions to their business needs

### INSIDE TOGAF

**The Open Group believes that using an architectural framework like TOGAF will speed up and simplify architecture development**

TOGAF has two broad constituents:

1. The **TOGAF Foundation Architecture** - This is a collection of architectural building blocks, services and functions using which a specific architecture may be built.
2. The **TOGAF Architecture Development Method (ADM)** that can be used to define a methodology. It has the following nine basic phases:
  - Pre-work and preliminary activities: Gather stakeholders and develop consensus for high-level plan. Get everyone on board with the plan.
  - Development of a definition of broad scope and architecture vision and map the high level strategy.
  - Description of current as-is and target to-be business architectures and analysis of the gap in the two states.
  - Development of information Systems architectures for applications and data.
  - Definition of the target Technology architecture that will be implemented in future phases.
  - Development of Solutions strategy - what will be bought, built or reused, and how these solutions will implement the architecture described in the previous phase.
  - Development of migration plan based on priorities.
  - Implementation of Governance.
  - Implementation of Change management

Essentially, TOGAF is a top-down methodology, Like other top-down methods, it starts with the high level view and breaks it down into progressively finer levels of detail.

## SOA IN TOGAF

In late 2006, The Open Group announced the creation of The SOA/TOGAF(TM of Open Group) Practical Guide Project. This will be a joint effort of the recently formed SOA Working Group and the Open Group's Architecture Forum. The practical guide is scheduled for release in early 2007. At the present time, there is a lot of activity in the working group to lay down the details of this document. When completed this effort will produce the following deliverables:

**In late 2006, the Open Group announced the creation of the SOA/TOGAF Practical Guide Project**

- A Practical Guide to using TOGAF to 'do SOA'
- SOA Definition
- Adjustments to TOGAF ADM Recommend Views & Attributes for SOA
- Recommend Artefacts for SOA
- Provide example SOA Building Blocks
- Provide Recommended Reference materials and Reference Architecture (from other SOA WG project), SOA Ontology (including those other SOA WG Project) and Process Models.

## BOTTOMLINE

The Open Group is an influential organization with a large membership of experienced Enterprise Architects. There is still debate in the Architect community about how to approach SOA. Is it just another type of Enterprise Architecture? Or does it deserve special treatment. At the present time, this debate is active within the Open Group itself and it will probably take a few months for the dust to settle and the Practical Guide book project will go a long way in bridging this gap between traditional EA and SOA.

## OMG, MDA AND SOA

The Object Management Group (or OMG) is a leading vendor agnostic industry standards consortium. It is well known for its Model Driven Architecture (or MDA) approach and is now leveraging this approach to become a prime player in the SOA space.

### MDA

The term "Model-Driven Architecture" (or MDA) is a registered trademark of the Object Management Group (OMG). It separates business-level functionality from the technical details of its implementation. It enables business-level functionality to be modeled by Unified Modeling Language (UML). This approach allows the models to exist independently of platform-induced constraints and requirements. Such models can then be instantiated into specific runtime implementations, based on the target platform of choice.

**There is a need for SOA modeling techniques that can help architects design the SOA independently of the implementations.**

### MOTIVATION

OMG believes that while there are several existing standards and approaches for SOA, there is a gap that needs to be filled in this space. On one hand standards like WS-\* operate at low-level of abstraction focused on data (XML), communications stacks (such as SOAP) and inter-operability. Because of this low-level view they are focused on only the technology aspect of SOA. On the other hand, existing architectural modeling tools like UML2 are also not fully service centered in their modeling approaches. There is a need for SOA modeling techniques that can help architects design the SOA independently of the implementations. Such modeling approaches need to address gaps in existing UML modeling for SOA including location and binding information support, Service Composition and many other SOA specific attributes so that modeling of service interactions is more effective. OMG is working to leverage MDA to address these gaps and become a mainstream SOA technique.

### GOALS

OMG's primary goals include the development of a common vocabulary and meta-model for SOA and to revisit UML to include SOA capabilities Specifically, OMG is focusing on strengthening the following areas:

- ***Service contracts*** that define interactions between service consumers and providers, including roles, responsibilities and behaviors.
- ***Service Specifications*** that will define Service operations, provided and required service interfaces
- ***Service Data*** including the definition of Structural information exchanged between service consumers and service providers
- ***Service Invocation and Event Handling*** including support for both synchronous and asynchronous service invocation. (this includes the ability to receive and process an events)
- ***Service Parameters*** passed between Consumers and Providers

OMG would also like modeling capabilities to:

- Specify how services are composed from other services
- Define "Service Channels" for connecting between usages of service consumers and service providers
- Enable customization and extending Services
- Support Service Model Interchange through XMI (XML Metadata Interchange)

**OMG is making strong attempts to align and make its MDA-SOA efforts complementary to ongoing work at other major standards bodies and organizations.**

#### ALIGNMENT

OMG is making strong attempts to align and make its MDA-SOA efforts complementary to ongoing work at other major standards bodies and organizations. This includes efforts underway at the Open Group, W3C, Integration Consortium and others. The goal is to avoid duplication and instead develop complementary approaches. Due to this alignment, OMG is focused primarily on the development of modeling techniques and not currently pursuing other areas such as Methodologies for Service Design, SOA Governance and low-level standards like the WS-\* stacks.

#### BOTTOMLINE

Independent observers and Analysts are optimistic about the future of this marriage between MDA and SOA, which OMG is brokering. By combining the business-level modeling focus with core SOA concepts, organizations can have a practical method to design and implement SOA's. OMG expects to have responses to its RFP by mid-2007 and finalize the specifications by early 2008. It is likely to elicit much interest across the IT architecture community.

## ORACLE'S SOA MATURITY MODELS

"SOA Maturity Models" are tools that can help in the assessment of an organizations progress with regard to SOA. They can also help organizations set goals and define roadmaps for their SOA initiatives. Numerous such models are prevalent in the market - some proposed by vendors like Oracle, IBM, BEA, Sonic, Systinet and AmberPoint and others proposed by independent IT Analysts like ZapThink and CBDI (*At the time of writing this document there were over a dozen!*) Most of them contain similar concepts with minor variations and typically categorize "SOA Maturity" in levels - such as Levels 1 through 5 with each level having particular characteristics. The sections below describe one such Maturity model defined by Oracle to illustrate a typical case in point.

### ORACLE'S SOA MATURITY MODEL

Through an SOA Maturity Model, Oracle helps to define the technical and organizational characteristics of businesses with SOA implementation, at several discrete levels of maturity. They define a SOA maturity model that helps to address common questions and provide guidance in planning and execution. The 5-level SOA Maturity Model gives decision makers a frame of reference to evaluate the SOA maturity level of their organizations. This Model gives decision makers a framework they can use to evaluate the current state of SOA capabilities across their architecture, tools, technologies, development processes, information management, governance, people, processes, and internal organization structure.

**Through an SOA Maturity Model, Oracle helps to define the technical and organizational characteristics of businesses with SOA implementation, at several discrete levels of maturity**

#### LEVEL ONE: OPPORTUNISTIC

This is the Initial exploration stage of SOA - Most organizations are currently at the first level of adoption. Here, businesses identify simple projects that can be completed quickly, often in the form of a prototype and usually building a service on top of an existing application and exposing it to a user through a Web portal or a Web Service in an Application Server environment. The focus is to show quick value and demonstrate ROI to business end-users and management

#### LEVEL TWO: TACTICAL

A big leap from Level 1. Organizations begin to use the ESB. They begin to orchestrate web services with BPEL. Users commonly interact with composite applications assembled from individual services. Service reuse benefits begin to be felt. Use of Web services security and management solutions to enforce different security policies begin to become a priority. User and identity management infrastructures begin to become more sophisticated.

#### LEVEL THREE: STRATEGIC

In this stage the focus shifts from technological change to organizational changes. The SOA infrastructure is in place and well understood. Level three is about putting in place the strategic business changes to effectively use SOA and Business Process Management. Automation of manual business processes is a key focus. At this stage, an organization's business users regularly uses BPEL to freely define, automate and change underlying processes according to business needs instead of being constrained to existing software solutions. SOA is now ingrained in the development strategy and all new projects must factor this in.

#### LEVEL FOUR: ENTERPRISE WIDE

Adoption in this fourth stage of the SOA maturity businesses have pervasive measurement and improvement mechanisms built into their SOA applications that track key performance indicators (KPIs) and service level agreements (SLAs) in real-time. This information allows the business to more efficiently and effectively manage its operations and to optimize business processes in response to real-time and historical data. Event-Driven technology and infrastructure starts being used extensively in order to build sense and respond applications.

**Maturity Level 5 is the “Nirvana” scenario of SOA – the organization is in a continuous, real-time feedback loop that automatically adjusts business processes and systems to changing conditions**

#### LEVEL FIVE: INDUSTRIALIZED SOA

Maturity Level 5 is the “Nirvana” scenario of SOA – the organization is in a continuous, real-time feedback loop that automatically adjusts business processes and systems to changing conditions. Complete automation of the system with the business users managing the entire process from a "Dashboard"

#### THE BOTTOMLINE

Organizations can use SOA Maturity Models to assess their current state of SOA and to understand the milestones they need to cross to reach the final destination. However, other than providing a high level road map on the path to SOA these types of models do little else. They need to be used in conjunction with other tools like a life-cycle methodology and architectural approach.

With regard to Oracle itself, Analysts are upbeat about its comprehensive SOA approach (part of its Fusion middleware offering). There is talk on Oracle’s web site about a “methodology” to help customers implement SOA but at this time, no details are available about a generic SOA methodology or framework other than the maturity model described above.

## IBM SOA FOUNDATION

IBM's SOA Foundation defines a comprehensive "life cycle" for SOA that includes the use of model-driven techniques to design the SOA. It includes four broad phases: Model, Assemble, Deploy and Manage.

**IBM's SOA Foundation Life-Cycle includes four broad phases: Model, Assemble, Deploy and Manage**

### MODEL

Modeling analyzes the business and captures the business design from an understanding of business requirements and objectives. The business model is a specification of business processes, goals and assumptions for the IT systems. IBM recommends the use of advanced modeling tools that allow processes to be iteratively refined by simulation to align business goals with the IT systems. The model is also used to capture key performance indicators (KPIs) for the business that can be input to the assembly phase of the lifecycle. (More on the specifics of IBM's SOA modeling approaches RUP and SOMA will be elaborated further into this discussion)

### ASSEMBLE

During this phase, one takes the business design is used as input to assemble the information system artifacts that implement the design. The enterprise architect and business analyst work closely with the software architect to develop the design of the services. Discovery and Reuse of existing services is an important activity. Typically, a few existing (legacy) services will be a perfect fit, while others have to be augmented as needed. Legacy assets that could be re-used are exposed as services. New services have to be created in cases where there is no reuse possibility. Software developers should use the standard SOA programming techniques and best-practices (defined separately by IBM) to create these new services. Policies must be pre-defined for services to control how they operate in the production runtime environment. For example, these policies and conditions could include business and government related compliance regulations. In addition, the assemble phase includes critical operational considerations that include creation of deployment configurations and policies to that ensure smooth operation of the application in the SOA Infrastructure.

### DEPLOY

This phase includes the creation of the SOA runtime environment (if it does not already exist) and deployment into the runtime. This phase also includes resolving the application's resource dependencies, operational conditions, and capacity planning and integrity/security/access constraints.

### MANAGE

This phase includes the specific tasks around the operational technology and tools used to manage and monitor the services that are deployed to the SOA. Monitoring of service for availability, performance, failure detection and recovery initiation is critical. Managing the routine maintenance, administering and securing applications, managing resources and users, analyzing and predicting future capacity growth are also part of this phase. It also covers all operational aspects of security - authentication, authorization, single sign-on, identity

management, and user provisioning (often this is done by a set of common security policies). The manage phase also includes feeding back and tuning the original business model, and also tuning the operational environment to meet the business needs. This creates the constructive closed feedback loop that is essential for good SOA governance.

## GOVERNANCE

Effective SOA Governance is central to IBM's strategy. Governance includes the following:

- Establish decision rights
- Define high value business services
- Manage the life cycle of assets
- Measure effectiveness

## SOA FOUNDATION REFERENCE ARCHITECTURE

IBM defines a logical architecture for SOA with the following "core" classes of Services.

- *Interaction Services* to enable collaboration between people, processes & information
- *Process Services* to orchestrate and automate flow of business processes
- *Information Services* to manage diverse data and content in a unified manner including data transformation or replication from different data sources.
- *Business Applications Services* that are called by end-users/consumers
- *Access Services* to facilitate communication ESB between services
- *Partner Services* to connect with trading partners

Supporting Services include

- Business innovation and optimization services
- Development services
- IT Management Services
- Infrastructure services and the Enterprise Service Bus

## MODELING FOR SOA

One of the main issues that SOA tries to address is the issue of Business - IT alignment. Historically, there has been a major disconnect between Business Goals and IT deliveries. SOA creates alignment between business and IT by focusing first on the identification of important "Business Services". Business Services act as a bridge between business tasks on one hand and IT software services on the other. IBM's Service Oriented Modeling and Architecture (SOMA) address these gaps in its model-driven approach for SOA.

**Effective SOA  
Governance is central to  
IBM's strategy**

## IBM SOMA AND RUP

Service-Oriented Modeling and Architecture (or SOMA) is IBM's method for SOA modeling. SOMA combines two IBM SOA methods. Originally it was derived from IBM's own internal architectural approaches (called GS Method) but later IBM integrated into it the SOA capabilities in UML 2.0 which it offers separately in its Rational tool set via the Rational Unified Process (or RUP), thereby providing a single modeling solution for SOA from IBM. SOMA is still internal to IBM but IBM has now rolled it out a direct derivative of SOMA, as a part of its Rational offerings. Presently, this is IBM's Rational Unified Process (RUP) for Service-Oriented Modeling and Architecture V2.x.

## PHASES IN SOMA

SOMA has three broad phases – Identification, Specification and Realization.

- **Service Identification phase** - Service identification can be done in either top-down or bottom-up approaches. Top-down-approach starts with a high level business view drilling down into models that represent business processes, and further broken down to identify business services Bottom-up begins at the solutions layer, looking at existing implementations and working up to the business layers. *(Note: Accepted industry best practices recommend using a combination of top down and bottom up iteratively for best results)*
- **Service Specification phase** -This phase involves describing the service: what it is, what it offers, what it requests, how it is exposed, etc. Interdependencies with other services, service composition, and service messages are also identified. Service Reuse, one of the most important aspects of a SOA is considered in this phase. The main model related to this phase is the service model.
- **Service Realization Phase**-This phase develops the actual solution services. The activity now focuses on the "how" the details of how the service is to be realized and results in the design model. (The design model will map directly to the realization model)

**SOMA has three broad phases – Identification, Specification and Realization**

## THE BOTTOMLINE

IBM has positioned itself strongly in the SOA space as a premier service provider, leading product vendor and an industry-accepted SOA thought leader. Its SOA Foundation methodology is one of the more robust SOA formal methods currently available. RUP has been a familiar and popular way to model traditional architectures and by adding SOA capabilities through its SOMA plug-ins to RUP, IBM is certainly positioning its SOA approaches to be a popular choice for SOA practitioners globally.

This said, IBM is still a “vendor” and at a time when customers are slowly but steadily moving themselves away from vendor locks, it is likely that many organizations will seek out vendor-neutral and open SOA frameworks and methods when considering SOA adoption. It is conceivable however that IBM’s SOA approaches (or some direct derivation) may be players in the Open space as well. Major standards organizations like the OMG and organizations like the US Federal Government have an interest in IBM’s SOA methods and models and it remains to be seen how these efforts will play out in the market.

## “OTHER” SOA FRAMEWORKS

As stated at the outset, this document only considered some of the numerous existing and emerging SOA Frameworks, Methods, Architectures and Models. There are a number of other SOA frameworks that were not considered in this paper but may merit a closer look by an SOA practitioner who is researching this subject more deeply. These include offerings by:

**There is a plethora of other SOA Frameworks, Methods, Reference Architectures and Models that were not considered in this paper but may merit a closer look if you are interested in this space**

- **Analysts** - IT Analyst firms like the Burton Group, Forrester, Gartner, ZapThink, CBDI Forum and the like are defining their own versions of vendor-neutral Reference Architectures, Models and Ontologies for SOA.
- **Open Standards bodies** – There are many noteworthy initiatives underway in various leading “Open” forums, which were not discussed in this document. These include efforts by ISACA (which defines the COBIT framework), The Integration Consortium and the OASIS Reference Model TC, which has laid down a comprehensive and open SOA Reference Definition and Model for SOA.
- **Government** –All levels and agencies of Government at the Federal, State and Local level are active on the SOA frameworks landscape FEA or Federal Enterprise Architecture is a US e-Government’s Enterprise Architectural framework that is actively looking at SOA. Another example is DODAF, an EA framework for the Department of Defense.
- **Vendors**– There is much consolidation of technology in the vendor space, with comprehensive SOA offerings and suites by IT vendors like BEA(under its 360 platform), TIBCO, Oracle (under its Fusion middleware offering), WebMethods, SUN, HP and the like. These vendors are not only enhancing their SOA product stacks but also beginning to offer comprehensive models, frameworks and methodologies comparable to IBM’s SOA Foundation or Oracles Maturity Model that were discussed earlier in this paper. Organizations that are loyal to a particular vendor may benefit from the adoption of such frameworks.
- **Open Source** – There is visible movements within the Open Source world to respond to the growing needs for SOA consolidation. Iona, Jboss and Apache have either already released or have under active incubation, several products like ESB’s and tooling for SOA. It can be expected that over the next year or two comprehensive SOA Open Source stacks and frameworks will begin to emerge to enable organizations to leverage Open Source for their SOA implementations.

## SO MANY FRAMEWORKS, SO LITTLE TIME...

The average SOA practitioner may understandably feel overwhelmed with the plethora of SOA frameworks, models, methods and reference architectures that are proliferating the IT industry. Some concluding thoughts and take-away points on this subject may help in providing guidance in this regard.

### WHICH ONE IS BETTER?

In doing a typical evaluation (of a set of technology products, for example), it is common to develop some formal evaluation criteria and measure the products being evaluated against the criteria to arrive at a “rank” or “order”. Such simple benchmarking is appealing because it sets things apart clearly in black and white and identifies the “winners and losers”. Unfortunately, things don’t work that simply in the world of SOA.

For starters, SOA, in itself is complicated. It is not a Technology, Product or even architecture. It is, as OASIS defined, a “paradigm”, an architectural style. It is not something that can be bought off the shelf and put to work. Rather it is a result of the crystallization and convergence of decades of IT experiences and best practices that can help IT Systems align with Business goals, an approach for constructing more standard, agile and reusable systems. It requires fundamental shifts in not only IT strategies, but also in the underlying Culture, Behavioral and Operational aspects of a business. The benefit of SOA is that Organizations, which go down the complex route, can do so incrementally and gradually work up the SOA maturity ladder but this is a non-trivial undertaking.

**SOA is not something  
that can be bought off  
the shelf and put to work**

SOA Frameworks, Models, Methods and Architectures are support structures that can help an organization with different aspects of this complex undertaking. However, as may be evident from the earlier discussions in this Paper, there are many sizes, shapes and flavors for these approaches. Some operate at a low and some at a high level of abstraction. Some focus on certain business aspects of an organization such as Compliance or Legacy Migration, while others may focus on IT architecture or Technology. Comparing one to the other would be like comparing apples and oranges – at best an elusive effort.

It would instead be prudent to view each framework as what they are - “support structures” or “tools”, each designed for a certain purpose and understand how they should be applied - individually or in combination to assist with the various tasks required in an SOA undertaking. Which framework or combinations you use will depend on what you are trying to accomplish. While a “one size fits all” framework may seem like utopia, the reality is that you need to invest the time to understand these different frameworks and then develop a strategy and the skills to use them effectively.

For example, an organization may use an SOA Maturity model to assess the current SOA maturity of an organization and to charter a high level roadmap with milestones. A method like the OASIS (draft) Service Architecture methodology or TOGAF could be used to develop an Enterprise level Service Architecture View of the System. An approach like MDA could be used to develop the models and generate the UML and code artifacts. Many other alternative combinations of such approaches are also possible.

### THERE’S NO SUBSTITUTE FOR SKILL AND EXPERIENCE

Continuing the analogy of support structures and tools - just as buying a set of good tools does not guarantee the success of a construction project, using even the best frameworks and methods will not guarantee the success of an SOA initiative. There is absolutely no substitute

for skill, experience and perseverance at all levels of an organization if SOA adoption has to really succeed. Frameworks are just what they are – supporting aids to help the skilled architect. Even the best tools are of little value to an unskilled or inexperienced practitioner. On the other hand, a skilled craftsman can still achieve a lot with simple tools (and will possibly achieve wonders with good ones). Given this, organizations venturing on an SOA should first focus on building a good team of skilled and experienced SOA practitioners before selecting a set of frameworks, models or methods. Experienced practitioners will then pick the tools that are right to get the job done for you. While this has probably been true for all IT initiatives, it is even more so with SOA where organizational, cultural and behavioral factors are so critical to success.

**There is absolutely no substitute for skill, experience and perseverance at all levels of an organization if SOA adoption has to really succeed**

## DON'T UNDER-ESTIMATE GOVERNANCE

Amidst all this discussion about Frameworks, Methods, Models and Reference Architectures, don't forget about Governance. SOA Governance is NOT over-rated. If there is one thing that can break an SOA, it is the absence of effective Governance. Good governance provides the solid foundation on which an SOA Framework can operate. So when selecting a Framework or creating your own, make sure that it provides for this. In fact, some organizations (like The Burton Group) are creating Reference architectures, frameworks and models aimed specifically at this one key aspect of SOA.

## CONSIDER A CONSOLIDATED IN-HOUSE APPROACH

Most organizations that are serious about SOA will study these different frameworks, architectures and models and either combine them, adapt them or integrate them into their own set to develop a comprehensive set of SOA approaches that suit their business needs and objectives. Following this best of breed selection and adapting it based on its own business experiences may be the optimal strategy

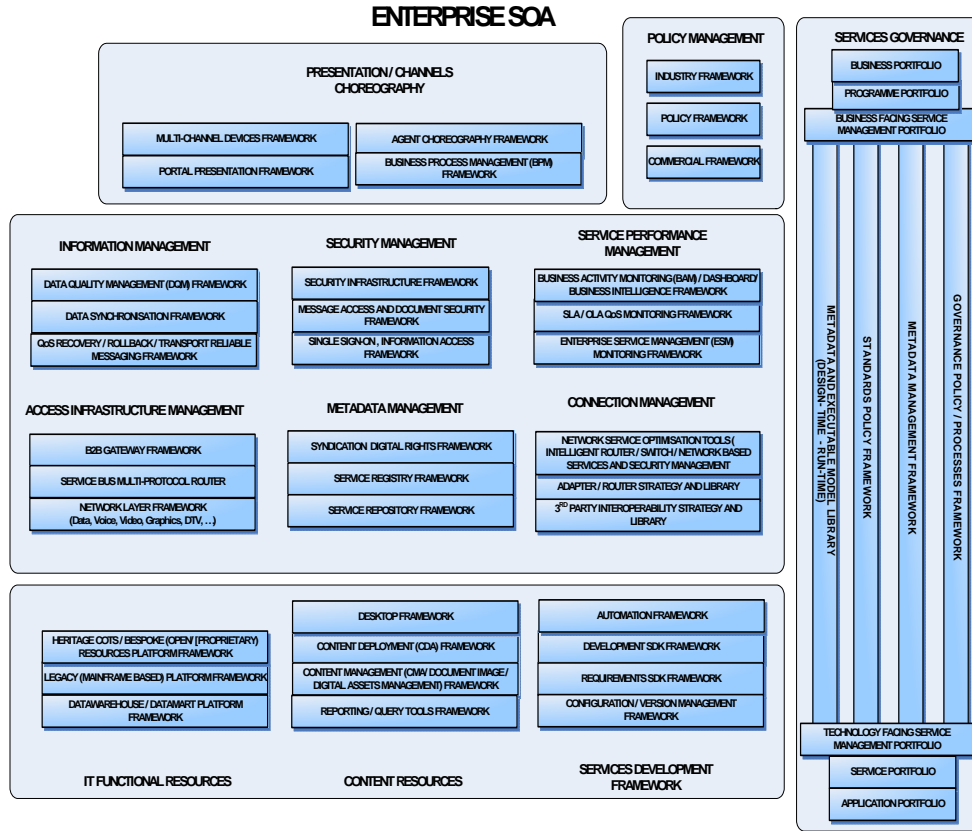
## CSC AND SOA

CSC is a good case in point for this comprehensive approach. With a thriving community of IT architects, a global pool of SOA thought leaders and a vast array of real world SOA experience, CSC has much to leverage. The CSC Catalyst SOA project, which is underway currently is tapping the pool of internal thought leadership and experience, along with industry best practices and rolling it into CSC's Catalyst methodology. The CSC SOA portfolio also includes an SOA Reference Architecture (shown on the next page) and a comprehensive SOA framework for Enterprise Architecture that provides a common "language" to bridge IT and Business.

## THE BOTTOMLINE

Frameworks, Methods, Reference Architectures and Models can be valuable support structures in your SOA initiatives. They consolidate and crystallize the best practices based on vast bodies of experience and can help you get past re-inventing the wheel and expedite your SOA implementation. Trying to do an SOA without these approaches is like trying to do construction without a good set of tools. Organizations will be well advised to study these frameworks well, become familiar with their characteristics, select the ones that are right for their business and then use them intelligently to build a successful SOA.

**CSC SOA Thought Leader TECHNOLOGY ARCHITECTURE FRAMEWORK – 2006 PATTERN  
LOGICAL COMPONENTS MODEL  
VERSION 2006.1**



CSC SOA Thought Leadership Technology Reference Architecture (Courtesy: Mark Skilton)

## ACKNOWLEDGEMENTS

I would like to gratefully acknowledge Mark Skilton and Victor Harrison. Mark and Victor are senior CSC Enterprise Architects and SOA Thought Leaders, and have also been personal friends and mentors on my SOA sojourns. I thank them for being ever-willing “Sounding Boards” at different points over the last few months despite their busy schedules. I am also grateful to the many individuals at the CSC Research network who readily funneled huge volumes of interesting information that became valuable inputs in the process of researching this topic. Any thoughts, views, opinions (and any flaws or defects thereof) in this documents are entirely my own and do not in anyway reflect the views of CSC or any of the above individuals.

## REFERENCES

1. Marks Eric, Bell Michael, Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology. Wiley.
2. Erl, Thomas Service-Oriented Architecture (SOA): Concepts, Technology, and Design Prentice Hall.
3. SMART: The Service-Oriented Migration and Reuse Technique, by Grace Lewis,Ed Morris,Liam O'Brien,Dennis Smith,Lutz Wrage,September 2005.
4. Developing Realistic Approaches for the Migration of Legacy Components to Service-Oriented Architecture Environments, Grace Lewis and Dennis B. Smith, Carnegie Mellon University, Software Engineering Institute
5. Building SOA Solutions Using the Rational SDP (IBM) Draft Document for Review March 5, 2007.
6. OASIS Reference Model for Service Oriented Architecture 1.0 Committee Specification 1, 2 August 2006.
7. A METHODOLOGY FOR SERVICE ARCHITECTURES )by Steve Jones (Approved for release 26th October 2005.
8. Service Component Architecture Specifications, SCA Assembly Model V1.00 , March 2007.
9. Service Component Architecture Specifications, SCA Java Common Annotations and APIs V1.00 , March 2007.
10. Service Component Architecture Specifications,SCA Policy Framework V1.00 , March 2007.
11. Service Component Architecture Specifications,SCA Web Services Binding V1.00 , March 2007.
12. SOA Meets Compliance - Compliance Oriented Architecture by Redmonk. August 2004.
13. Introducing The Open Group Architecture Framework (TOGAF), Part 1: Understand TOGAF and IT architecture in today's world.
14. Connecting business needs with the technology infrastructure. (IBM Developer Works) 14 Feb 2006.
15. MDA Guide Version 1.0.1, OMG, Date: 12th June 2003.
16. CSC Research Network Briefings on SOA.
17. CSC SOA Thought Leaders Technical Reference Architecture, 2006.
18. Oracle Defines an SOA Maturity Model , By Cameron Crotty. September 22, 2006.
19. Amsden, Jim , ,IBM "UML Profile and Metamodel for Services RFP UPMS Services Metamodel" 28-Sep-2006.
20. "BEA Discloses its SOA 360° Platform Architecture" - Industry's First Native SOA Platform BEAWORLD 2006–San Francisco–Sept. 19, 2006.
21. "Open Source ESBs for Application Integration", Redmonk, February 16, 2007.