DataAccessTechnologies

Where Business Meets Technology

The Architecture of Services ™

Achieving Business Value with SOA

Agility, effectiveness and efficiency are the modern cornerstones of business, government and defense. These key goals are the basis for much of the transformation in the way we architect our enterprise and information systems and SOA has emerged as a key enabler.

There has been some confusion in the industry as to the positioning of SOA as a technology or business architecture. SOA, as presented here, is *both* a business and a technical approach. The SOA approach to organizing an enterprise focuses on agility by treating customers, suppliers and separable parts of the enterprise as a network of services. The SOA approach to technology helps realize these enterprise services with an agile, interoperable and modular technology base. The *combination* provides a path for transforming the enterprise.

To achieve pervasive business value the architecture of services focuses on how business and technical services work together to achieve business, government or military goals. The focus of architecture is not on "a service" but the network of services that serve a community or organization. SOA is a path to architecting this network of services, incrementally transforming both the enterprise and enterprise systems.

This paper provides a business and technical framework for achieving agility, effectiveness and efficiency with Service Oriented Architecture, or SOA. We will start with a foundation for what SOA is and why it is important. We will then explore what constitutes a quality SOA and the business value it provides.

Table of Contents

Driving Requirements for SOA

As the modern world has progressed the key markers for effectiveness have also progressed. The key metrics of today must focus on agility in the face of change, the ability to collaborate inside and outside the organization, exploitation of technology and on effectively and inexpensively achieving our mission.

The structurally and technically monolithic, static, waterfall and closed approaches that were successful in the past are now barriers to transforming into the organizations we must become. Overcoming these barriers requires a change in our way of thinking about our organizations and a change in approach to solutions. It also requires a blending of business and technical architecture into mutually supportive views. The organizations that can achieve this transformation have an opportunity to prosper and will ultimately replace those who can't.

For these reasons the following are the driving requirements for our approach, the approach we will identify as SOA;

- Agility in every business[i], organizational and technical dimension
- Ability to collaborate quickly and effectively both inside and outside the organization
- Ability to respond to new mission, business or external needs and goals
- Ability to respond to and exploit technology and technical change
- Ability to operate with reduced resources and resources out of our control
- Ability to operate within communities, to achieve joint goals
- Modularity of business and technical capabilities
- Ability for incremental success and early results

We can and must architect our organizations and I.T. projects with an approach that can satisfy these requirements. This is different than what we are doing today but builds on what we have.

The cornerstone of SOA transformation

How can we hope to achieve such widespread and dramatic transformation? How can we deal with the complexity of our organizations, relationships and technologies in such a way that we are agile? Has the very success of our vertically and horizontally integrated organizations made it impossible to respond?

There are general and proven approaches that have succeeded over and over to manage complexity with effectiveness and agility. There have been groups, organizations, systems and projects that have overcome hurtles to achieve. What makes this work?

These same problems have been dealt with in manufacturing, government, defense and complex information systems. The unifying principle for success was identified by Edsger W. Dijkstra, a famous thinker in the area of software and software architecture. That unifying concept is;

- ***Separation of concerns***

Separation of concerns is what allows us, limited humans, to deal with complexity and interdependencies of diverse needs and resources. It is what allows us to achieve a joint purpose without a centralized control. It is what allows us to design, build and run huge organizations and information systems. While Dyjkstra focused on software, the principles of separation of concern can be applied to any "complex system", be it government, business, military, natural or technical. The modern "network centric" approach to warfare is another expression of the same concept, as are the proven management principles of delegation and a focus on core competency.

Separation of concerns is simply the art of focusing ones attention on a particular aspect of a problem, organization, mission or system. This focusing of attention provides both a way to **comprehend that aspect**, to **understand how it relates to other aspects** and to accept that **other aspects may be under the control of others**. Our challenge is to understand how to separate our concerns while making sure that these concerns come together to achieve our mission – this is the art of architecture and design.

Delegation – business and technical

Great managers know that they can't do everything, they are great at delegation – at assigning responsibility, letting good people achieve and making sure all the parts work together. Great managers have learned how to separate concerns along the dimension of responsibility and action so that their team works effectively. Thus delegation is a powerful tool for separation of concerns.

Delegation allows creativity and diversity in how responsibilities are met, while putting constraints on that creativity and diversity so that the goals of the whole organization are achieved.

Encapsulation – business and technical

Encapsulation is another term that is popular is systems architecture, but that applies just as well to organizations. Encapsulation is separating the concern of **what** Vs. **how**. In computer terms this is sometimes called the "interface" Vs. the "implementation". One thing the great manager is doing when they are delegating is encapsulating a responsibility – this is where the creativity and diversity come in.

A business example

If a manager is telling someone exactly what to do, this is assignment, not delegation. Assignment is a powerful tool, but it has problems in the large. Eventually the responsibility, the "what" has to be separated from the details of how something will be achieved.

Consider two scenarios. In one case a detailed process for accounts receivable has been "wired" into an organization. The details of how A/R is done are specified in great detail and it is integrated into all parts of the organization. This kind of tight integration is called for in some situations and is the basis for some management theories. In the other scenario A/R is delegated to a specific group of experts, experts who have a clear responsibility and interact with customers and other parts of the organization to gather

information and assist where such information is required. What A/R does for the organization is clear – but exactly how they do it is not so wired into the organization.

What if management decided to outsource accounts receivable? In the first scenario the organization would be dramatically effected, they would have to change their processes and the way they work together. In the second scenario the change could be almost invisible, because in the second scenario A/R was already "acting" like an outsourced organization – they were providing a service to the organization without being wired into it. This starts to provide a hint as to why a services approach is a good way to think about your organization.

Now think outside your organization, isn't your customer (or whoever your organization serves) delegating some responsibility to you? Aren't you delegating responsibility to your suppliers? Externally and internally there is a need to encapsulate "the other guy" so that we can play in a larger community while not being overly concerned about how they work "inside". It should be starting to be clear that "encapsulation" is needed to structure or organizations and missions and that this is the business side of SOA; SOA as "business services" that interact and, together, achieve the goals of the organization and the communities in which it operates.

A technical example

In software the concepts of encapsulation are well established, in fact they are the basis for most quality software architecture. The "object oriented programming" revolution of the 80's and 90's is largely based on encapsulation – that "objects" have interfaces and associated required behavior, but how the objects are "implemented" is hidden. This hiding of implementation was controversial at first, because it seems that it could impact efficiency (the same comment is made when businesses encapsulate and delegate). But the complexity and fragility of systems that do not employ encapsulation has proven to be a much more substantial problem than the loss of efficiency – the agility and stability are worth the cost.

With the onset of the internet and the ability for widely distributed systems to interact, the need for encapsulation has multiplied. Our systems now routinely work with other systems inside and outside our sphere of control, it is simply unreasonable to think we can impose on their implementation or their process, we can only deal with what they expect from us and what we expect from them – as we will see, this is an essential element of SOA as a technical architecture.

If the "pattern" of SOA doesn't seem much different for the business and technical example, you have caught on – that's the key.

Community and Collaboration

A central driving theme in business, government and defense is collaboration – bringing together diverse groups, individuals and resources to achieve a shared goal. Collaboration exists within a community – some groups that wants to or needs to work together. The capability these communities need is collaboration, the ability to work together.

This drive towards communities and collaboration is so deep that it impacts the way we think about our organization, from supply chains to network centricity. The focus is now on "joint action" rather than individual actions. The environment in which we operate becomes a driving force for defining our purpose and how we fit in to the community, how we collaborate and the value we provide.

Communities and collaboration fit hand-in-hand with delegation and encapsulation – only with separation of concerns are we able to be sufficiently agile to work within a dynamic community, only by "encapsulating" our details from "the outside" can we remain efficient and agile and only by delegating responsibility to others can we help achieve the goals of the community. Helping the community achieve its goals correspondingly provides value to each participant.

This suggests a shift in thinking from one that is focused on "our process" to "how we collaborate in a community". Our process is not the center, it is a means to providing value to communities. That value is realized by the services we provide to the community and accomplished using the services of others. These services are realized by a combination of organization, resources and technology, working together.

Communities can be large or small and have almost any purpose. Anytime there is "working together", "customers", "collaborators" or "suppliers" there is some community, here are a few examples;

- Retail buyers and sellers
- The multinational force in Iraq
- The accounts payable department
- The U.S. Central Contractor Registration "CCR" (The vendors, manager of the information and consumers of the information)
- Internet routers
- The federal supply service
- Insurance providers
- Visa International
- The 9th Infantry Division
- Facilities Management
- The providers and consumers of internet time service (NTP)
- General Motors

Service Viewpoints

We should already be getting a sense of one kind of separation of concerns, the encapsulation of services within a community. But, this is only one dimension of separation of concerns – we have to think about technologies and security and finances and regulations. There are entire families of concerns that ultimately must be knit together, such as;

Roles within a community

Architecting services based on organizational structure, existing systems or current partners tends to be fragile as boundaries frequently change, both within and outside the organization. In addition, organizations and responsibilities move across organizational boundaries, being outsourced and in-sourced. Systems, of course, change. For these reasons we want to keep our primary view of services relative to the roles people,

organizations or systems play within a community instead of the physical structure.

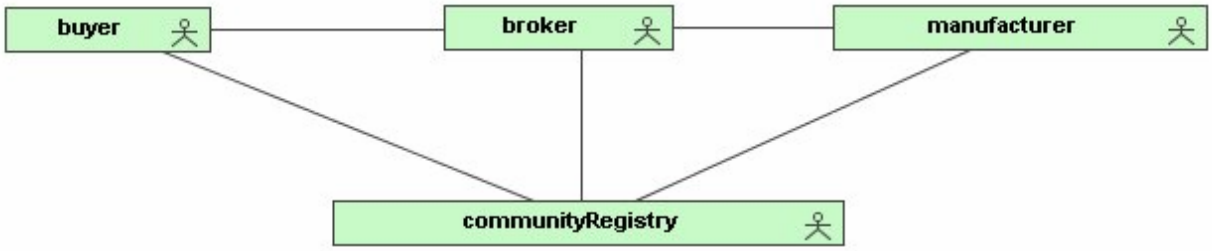


Figure 1- Examples of Roles within a Community

A community is then a set of roles collaborating for some purpose. A community may be global, such as international defense logistics, or local – such as the accounts payable department in a corporation.

Regardless of the size, a community has an over all purpose or goal and there various actors who "play a role" in this community, with specific capabilities and responsibilities.

The roles within a community define what services are provided and used by actors[ii] in that community and how they interact to collaborate within the community.

Roles may be "business" in nature or may be technical (such as a registry service). In either case the roles have specific responsibilities within the community and provide services to other roles as part of that community.

Atomic Communities

The smallest community, but a very typical one, has only two roles – frequently a provider and consumer of a single service. This type of atomic community (having a single service) is frequently the building block for more complex architectures. Many atomic services already exist in the enterprises and in existing systems and form the basis for "bottom up" analysis and exploiting legacy capabilities.

Services – Business and technical

A "service" in a business, government or defense sense is some capability offered to another party. Using the role concept, above, a actors playing a role provide a set of services based on their responsibilities and capabilities and also uses a set of services based on their need.

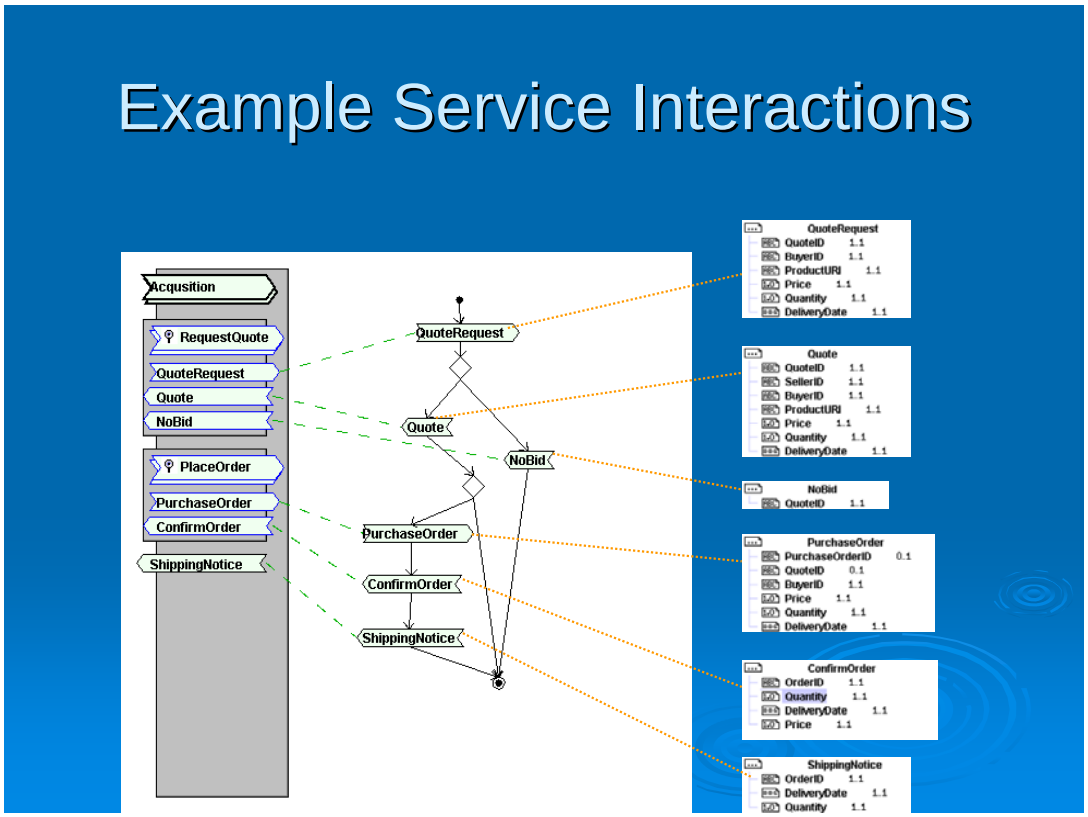
A service in a technical sense often means an interface that you can call to exchange data or establish or satisfy some obligation. This is generally organized as a set of "requests" from the service requester with a "response" from the service provider. The services in the technical sense is a means to providing the service in the business sense, for this reason we distinguish the technical notion of service as a "service interface" which implements an interaction for providing a service.

Interaction

Once we have more than one role there is a need to interact between those roles. The interactions are how services are requested are delivered between actors playing roles.

Some interactions are formalized, essentially the “must haves” to participate in the community, others are informal but assist the community in meeting its goals. Typically we architect the required and more formal interactions but provide for the informal ones as ad-hoc interactions. The formal interactions become part of the “contract” for participating in the community (or providing or using a service).

The contract of interaction includes all of the information, goods, services and obligations that flow between participants in the community, these interactions are part of the community contract. The other elements of the contract are the obligations that are created and satisfied as a result of these interactions, such as performing a service, delivering a product, taking an action or assuming risk.



Interactions are the visible part of a service oriented architecture – it is the interactions that are the basis for designing technical service interfaces, such as web services or distributed objects. At the architectural level we consider interactions to be potentially long-lived and bi-direction, as is the common case in business, military and human interactions. These interactions are then realized by technology interfaces that are frequently short lived and one unidirectional – these are the service interfaces.

The business interactions between participants are frequently mediated by our technology, thus the business interactions define some of the technical interactions between our systems. To the extent possible we would like our business services to drive our technology service interfaces.

Minimizing interdependence

The popular SOA technologies, such as Web-Services represent one dimension of concern – how our systems will interact. The concepts and technologies to achieve "distributed systems" have evolved over the last 20 or so years and blossomed as the internet has become pervasive. The internet has provided the technical infrastructure for collaboration and integration at a scale that is having a fundamental impact on how society works. We have been able to create new communities in record time, and also impacted some established institutions and ways of working.

The essential feature of working "on the internet" is that each party is typically working with an *entirely separate entities*. You don't share the same resources, processes, systems or even culture. What you do share is the desire to work together in some way.

Since there is so much interaction between separate entities you only want to agree on what you absolutely need to get the job done, otherwise everybody would have to be changing everything to collaborate with anyone new (which is the problem with some of our legacy systems). So the idea is; they control "their stuff" and you control "your stuff", what is in the middle is your "contract" for how to interact. The technical side of that contract is what is defined by technologies such as web services, RSS and email.

This is different from other forms of system integration in that nothing but these contracts of interaction are necessarily shared – no DBMS, no "common process", no "common component". This doesn't mean you can't use the same information or components, it does mean you don't have to. Sharing "common components" becomes a choice, not a necessity.

So the technical standards for SOA have to be able to be used across these very separate domains, separate organizationally and technically. The architecture for how this is done becomes the integration architecture for the community. For this reason we say that an SOA should minimize technical and business interdependence.

Information

Organizations run on data – agreements, accounts, plans, assets – information runs the show. The DBMS has been the centerpiece of information architecture as the basis for "what you know". What a collaborative environment and SOA does is add to the information picture. We are concerned with the information we *exchange* just as much as the information we *keep*. For this reason information actually constitutes two viewpoints – our information assts (the DBMS) and the messages (What is exchanged in SOA interactions).

The challenge, thus far, has been in keeping these views of our information consistent. The messages are frequently designed and implemented with minimal connection to the information stored. And while we don't want to couple our messaging information to our DBMS, we do want to share the concepts between them.

So in an SOA architecture we want our message information to be defined entirely in terms of the contracts of interaction, without notions of "storage". However, modern design practices make it possible to share the semantics of these messages with the semantics of what is stored in the DBMS.

Integration of our messaging and information storage semantics without coupling the DBMS or message design provides the basis for a solid SOA.

Security & Authority

The security and authority set of concerns dovetails with roles and actors. Roles give us a way to name a kind of participant in our community and the actors are then assigned rights based on those roles. The need for privacy and security in interactions comes from business requirements and is realized by the growing set of SOA security specifications and technologies.

Various security needs, such as secured identity, privacy of interaction and non-repudiation are mapped to the underlying technologies such as PKI and logging, in the technical infrastructure. Both the business needs for security and the technical architecture for providing that security must be matched.

Separating business and technical concerns

If there has been one constant in technology, it is change. In particular software and systems technical capabilities have progressed and changed with amazing speed over decades. A typical mistake in architectures is to couple the business architecture and needs with specific technologies. What then happens is that when the technology inevitably changes, it is hard to change the business. Or, when entities merge or try and work together, their technologies don't match-up.

To enable business and technical agility these concerns should be separated. To the extent possible a "business architecture" (remembering this includes government and defense) should *not* be tied to a specific technology. Business architecture can then be "mapped" to one or more technical architectures and this mapping can change over time. This allows *both* the business architecture and technical architecture to evolve somewhat independently. Recent advances in "Model Driven Architecture" (Or MDA as defined by the Object Management Group[iii]) provides standards for automating this connection between the business design and the technical architecture. For example, web services are frequently generated from the business centric model of a community of interacting roles.

To separate these concerns we would actually like three views of an SOA solution; a model of the business domain, a model of the application components and then the technology specific realization of the business and application components. Where we are focused on the SOA the technical specification will consist entirely on responsibilities and the contract of interactions, not on how any particular service is implemented.

EDOC SOA Architecture Standards

SOA focuses on a particular view of communities and organization, one that separates roles, services and interaction into the viewpoints suggested above. While there are many technical standards for SOA there are actually few that bring together this collaborative view at both the business and technical level. The "Enterprise Distributed Object Computing" (EDOC) standard of the OMG is one of the most comprehensive and therefore the standard we recommend for a service oriented architecture. This standard

describes roles, collaboration & components using the MDA paradigm such that the architectures can be realized in a variety of technologies – it is not dependent on "web services" yet it supports web services well. EDOC has also been mapped to BPEL and many of the other technical interface formats. The following are the requirements for an architectural approach;

- Ability to model roles and collaborations
- Ability to model bi-directional, nested and long-lived interactions between roles
- Asynchronous document oriented messaging model
- Ability to nest roles, having collaborations within collaborations
- Ability to relate logical roles with components
- Ability to recursively compose and decompose components and protocols.
- Mapping of components to implementation technologies
- Mapping of protocols to SOA middleware such as web services
- Connection between business processes and collaborations
- Connection between the data model and the messaging model
- Reuse of roles, protocols and messages
- Ability to map to the FEA
- Ability to define security requirements
- Ability to support simulation

SOA in the Enterprise

An SOA is part of the overall design of an organization, one focusing on services, roles and interactions within some community (noting that each organization is also a community).

Mission & Motivation

The mission of any organization should generally start with looking outward, what value it provides its citizens, customers or stakeholders and how it integrates with other organizations – suppliers and partners. This outward facing view of an organizations helps define its value proposition and role within the communities it participates in. For example, the U.S. General Service Agency frequently will play the roles of a broker or market maker – interfacing with both consumer organizations and suppliers.

The mission of an organization reflects this value proposition offered as the roles it plays within communities. Based on this mission the internal processes and resources of the organization can be focused and optimized. Connecting the outward facing value proposition with the internal processes and responsibilities is the basis of value chain engineering.

Business Process

You or your organization is prepared to act, to take on a role within a community to use and/or provide services, satisfy your responsibilities and comply with the communities contract. How do you do this?

It is important to distinguish your contract and role within one or more communities and how you will satisfy those responsibilities – this is the fundamental separation of concerns between what and how. The other members of the community probably don't care (or you don't care to tell them) how you satisfy your responsibilities. The community contract is the requirement for the design of your business processes. You business process is *how* you are going to do what is required, you have choice here. In fact, you can even change you process or have different processes for different situations or locations – provided the communities service contracts are satisfied. The ability to tune your process within your roles is fundamental to agility.

We emphasize this what/how split because it is imperative for an agile enterprise. All too often detailed business processes are specified that, if followed, would stifle the organization. It is frequently better to specify the interaction contracts and let the business processes be a bit more flexible. Specifying a business process is invaluable, over-specifying a business process is inflexible.

However, ultimately, someone or something **acts** and may do so with an established business process. This process specifies both the responsibilities for actions and the specific activities that must take place. This representation of business process can then include the well established "swim lane" diagram, where the swim lanes represent roles and within these are the activities of the business process. The business process specifies the way the organization will satisfy its requirements by performing activities and using and offering services. In addition, any subset of the process can be delegated to another party, satisfying responsibilities by delegation instead of direct action.

As services are used and offered within the process, the interactions of those services (based on the contract of interaction) are satisfied. This provides the connection between the external obligations and the internal processes.

Resources and Resource Allocations

The planning of resources, projects and funding can be integrated into an SOA architecture based on the roles various organizational units are playing (or are intended to play) and the resource requirements to support the activity. The separation of concerns provided by SOA can make the job of analyzing in source/outsource options easier as well as more accurately assign resources based on the services provided by organizational units to the organization and external community. Resource allocation is one of the governance views of SOA.

Organizational Structure

SOA provides mechanisms for organizations to be more effective in their structure and to tune that structure. By using separation of concerns at the business level the organization can be structured as semi-independent units that provide services to each other and to stakeholders. Once the roles each organizational unit plays is better understood it is more

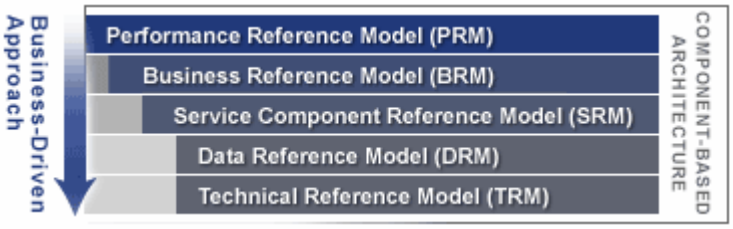
practical to spot and correct redundancy and inefficiencies. By isolating change to specific services it becomes easier to adapt the organization to new business needs and opportunities.

For this reason SOA should be looked at as part of business planning, as a way to distribute responsibility and resources to a network of interconnected business services. The SOA technology layer then provides the glue between these business services, facilitating and monitoring the interactions and providing system support.

In structuring the organizational model it should be clear what business units are playing what roles in various communities. While it may seem attractive to say that only one business unit may perform in a role, it is frequently more flexible to allow some redundancy where indicated, to provide for local control. The choice to centralize or decentralize becomes a business decision; the SOA can support approach and can support changes in the approach with minimal disruption.

The Federal Enterprise Architecture (FEA) and SOA

The U.S. Federal government is the largest organization in the world. The FEA is a bold attempt to provide a business driven architectural framework for this massive organization. The FEA and SOA fit hand in glove as SOA provides an architectural framework to realize the FEA.



The FEA is organized as a set of models, each providing a view on a particular concern that can be related to SOA is follows;

Performance Reference Model (PRM)

The PRM provides a framework for measuring and reporting performance in relation to measurable results. These performance metrics can be directly related to metrics of the services describes in the SOA. The measurements can also be tied directly to the interactions connected with that service. The technical infrastructure of the SOA can then provide the capability to monitor service interactions to record the service performance and "bubble up" the information to meaningful business metrics as defined in the PRM.

Business Reference Model (BRM)

The BRM categorizes the communities, roles and services that are offered within the federal government. In this way the BRM serves as a framework for connecting and integrating the various communities and services that comprise the federal government. BRM categories can be directly related to SOA communities and roles.

Service Reference Model (SRM)

The SRM categorizes the business services provided and used, which correspond well to the roles in a community as well as activities performed by those roles. The SRM is also evolving to encompass a more complete model of services and service interaction.

Data Reference Model (DRM)

The DRM categorizes there data of messages as well as the information behind messages in DBMS systems. The DRM also categorized the providers and consumers of the information as well as the "exchange packets" that carry the information. The DRM exchange packets and the associated information model correspond directly to the documents exchanged in SOA messages.

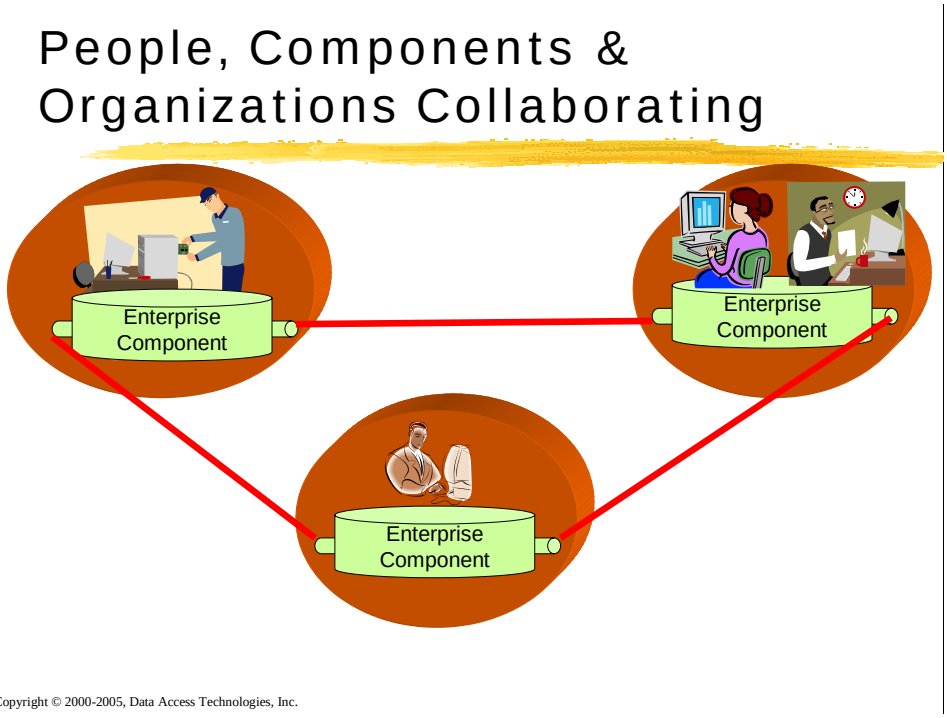
Technical Reference Model (TRM)

The TRM specifies the set of technologies and standards that are needed to implement and deploy an SOA solution.

SOA Technical Architecture, Legacy/COTS Wrapping and Components

Joining the business and technical views of SOA

The "joint point" between the business SOA and the technical SOA is the realization that each of the roles interacting at the business level can have a corresponding component in the technical architecture. The components in the technical architecture help realize the processes and interactions defined at the business level. This can be thought of as each business role having an 'automated assistant" that helps facilitate their responsibilities and coordinates the interaction with others using the SOA technologies.



The above picture suggests how people, who may be in various organizations work with their "automated assistant" – the enterprise components and then work with others in the community. In this way the enterprise components are helping to realize the business model.

Other components in the technical architecture may also use SOA for purely technical interactions, for example to manage access to a resource. So the set of SAO interfaces at the technical level will be those derived from business interactions as well as those required for the technical infrastructure.

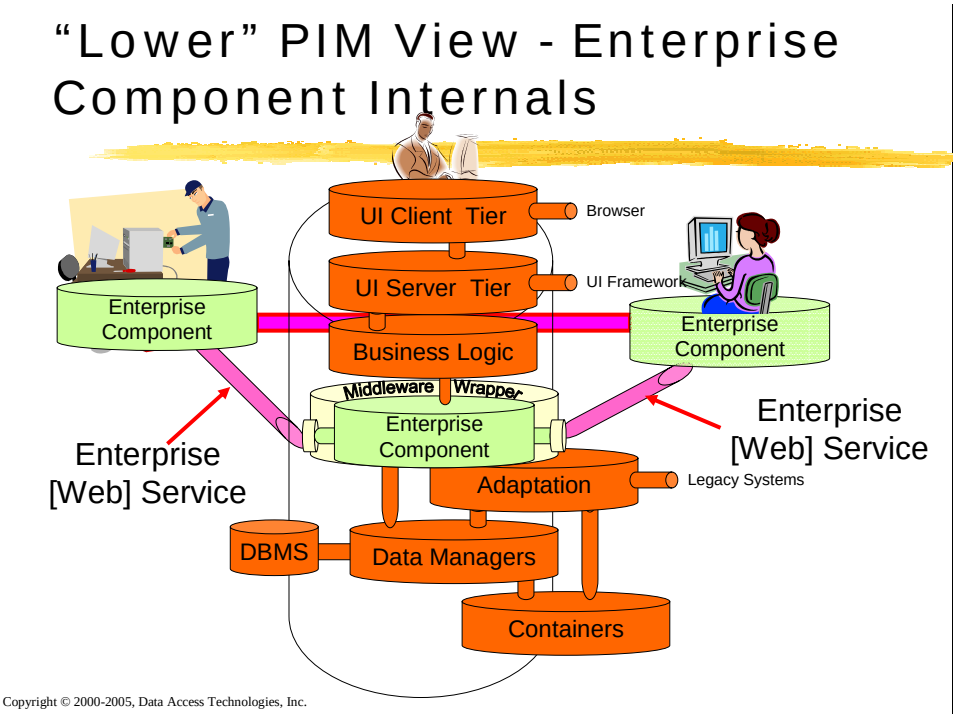
Inside a SOA Component and the ESB

An SOA component is a piece of software that provides and/or uses services via service interfaces. That software may be a wrapper around a legacy systems, it may be produced from a business process model or it may just delegate functionality to their components. Any such implementation requires an infrastructure in which to execute, this is sometimes known as the "Enterprise Service Bus". The ESB provides a variety of capabilities required to implement an enterprise class service component;

- Messaging Infrastructure
- Software "Container"
- Data Storage
- Legacy Adapters
- User Interface
- Workflow Management
- Business Rules
- Identity Management
- Encryption
- Logging
- Registries and Repositories
- Event publishing and subscription

- Transaction Management
- Standards

This implementation of the SOA component is hidden from external roles but it is critically important to have a quality infrastructure for SOA deployment to make the creating, deployment and management of the service components effective.



The above picture shows how the technical environment within an SOA component help the users of that component and interact with other components, yet the implementation infrastructure is hidden from the other components via the SOA messaging standards.

Achieving SOA with MDA

An enterprise approach to SOA involves multiple views of the enterprise for different needs and stakeholders. Yes, it is the same enterprise and the people, systems, processes, information and resources have to work together. Traceability is required between the business viewpoints and technical viewpoints – we should always understand the "why" of any design or implementation. There are two major resources that help provide for an integrated architectural approach while maintaining separation of concerns;

Automated integration and transformation of architectures

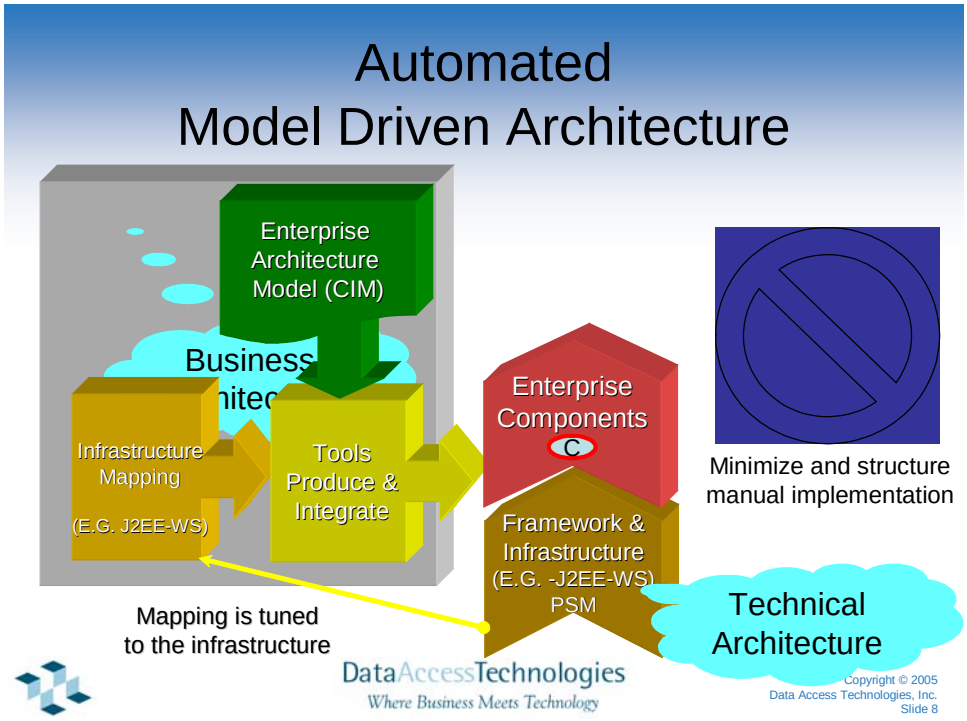
While we need to separate concerns into separate views, there is invariably overlap and dependencies between these views. For example we may want to talk about roles in a collaboration, or role based security or the responsibilities of that role or the service interfaces of that role or the software component realizing that role – in this case the role is prevalent in each of these views, and they are the same role. We want to make sure that as we look at and modify one view our architecture remains consistent across all other views.

In addition to consistency there is derivative information. Given an architecture we may be able to general technical interfaces, BDMS specifications or other technical artifacts. This not only saves time, it makes sure all the hundreds of artifacts that are required for a system are consistent with each other and reflect the architecture.

The Model Driven Architecture[iv] (MDA) standards provide mechanisms for managing the set of models that comprise an architecture as well as automating the transformations between them. In particular, the transformations from a logical architecture to one that is specific to a technology, such as web services or J2EE.

Attempting to manage an enterprise scale architecture completely at the "technology" level, without the benefit of MDA (Or something serving the same purpose)is not a viable approach – it would create yet another fragile and technology specific system not tied to business needs.

By making each of the SOA viewpoints a model in the enterprise architecture and "binding" these together with SOA, we have a full life-cycle and traceable architecture directly connected to the systems that support it.

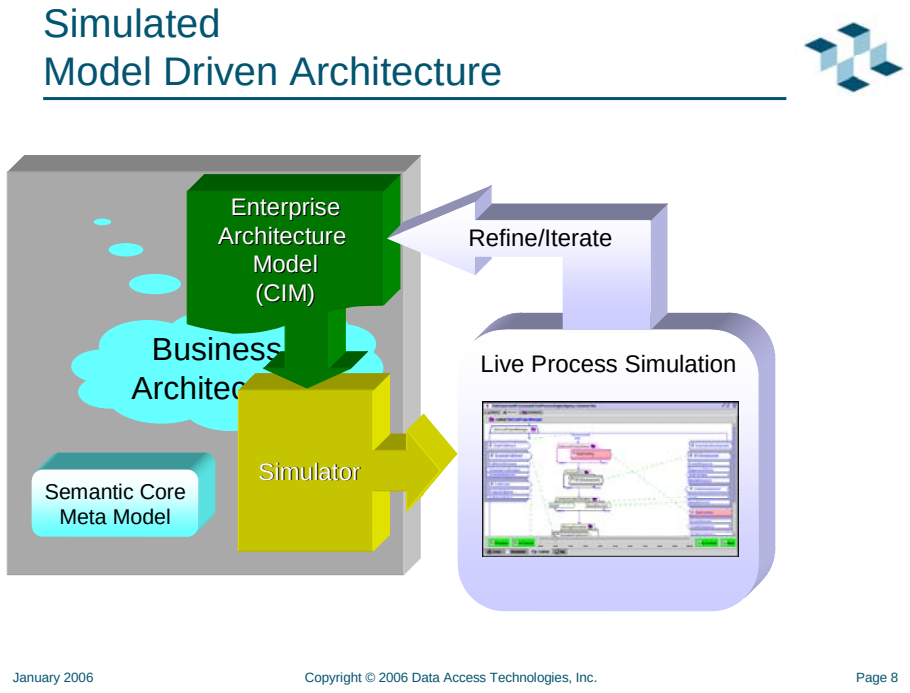


The picture, above, illustrates how the enterprise architecture combined with a technical architecture can be "mapped" to a set of enterprise components using MDA technologies. The depth of what is generated (Vs. what has to be hand implemented) varies, but at minimum the technical SOA can be produced, showing how all of the enterprise components interact across the SOA technical infrastructure. What this achieves is an automated "*re-integration of concerns*" based on the infrastructure mapping.

Simulation

Simulation provides for validation of architectures and testing of components prior to insertion into mainstream systems. Simulation for the business user is a way to see architecture "live" and watch information flowing through the processes of the enterprise.

Simulation in this context is based on the same architecture that will eventually drive the system, business processes and workflows.

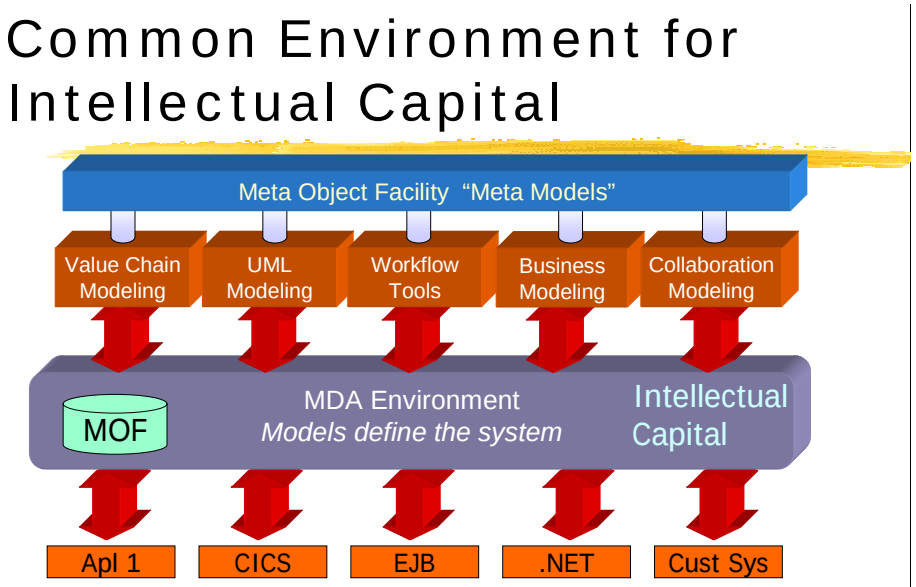


The picture above illustrates how the enterprise architecture can be used to drive a simulator that helps validate the architecture with business stakeholders for refinement. Once refined the architecture can be put into production and the same simulation can be used to test enterprise components.

Enterprise Metadata Repository

An architecture in the sense of an SOA (or any other executable architecture) isn't a "document", it must be in a form that is usable by both machines and people. The machine representation of an architecture is metadata, and these architectures become a critical asset of the organization. They must be preserves, shared, maintained and evolved. Since there are multiple views of an architecture there needs to be an infrastructure to maintain these views, the underlying models and the transformations between them.

Tools and infrastructures should be selected that con integrate with metadata repositories using open standards, such as OMG-"Meta Object Facility" and "XML Model Interchange". In addition, the semantic web technologies have recently become viable as metadata infrastructure, with the added benefit of direct connection to semantic web reasoning capabilities. The integration of MDA technologies and the Semantic web is still in development.



Integration of infrastructure

The picture above shows how the "Meta Object Facility (MOF) can integrate models from various tools, architectures and paradigms and then be used to power the generation of technical artifacts, such as SOA technical specifications (WSDL, Schema, BPEL, Etc). In integrating this information the MOF becomes an infrastructure for capturing, managing and disseminating the intellectual capital of an organization.

Semantic Services Architectures

Where there is a community of some form to agree on an SOA the architected approaches work quite well. Where traditional modeling or SOA technologies have trouble is where there is a requirement to integrate independently developed architectures – where the services are similar but the terms and structures are different, because they were developed independently.

The capability to integrate independently developed architectures based on the semantics of the models is where the semantic web technologies shine. The capability exists (or is in near term development) to use these tools to assist in the integration of architectures, services and the production of adapters.

Another capability that has been promoted is the dynamic (runtime) integration of services. This is not an avenue we are pursuing at this time due to the deep semantics and high overhead that would be required behind such an approach. Fast design-time integration of architectures would provide 90% of the benefit with far less overhead or need for research-level ontologies.

Approaches for using the semantic web with MDA and SOA are treated in more detail in companion documents that can be found on osera.modeldriven.org.

Creating a Service Oriented Architecture

A service oriented architecture starts with an understanding of what some community or organization would like to achieve, the drivers behind the collaboration enabled by SOA. By considering the collaboration of the community the set of services in the architecture has context and the way in which these multiple services come together to solve business problems explicit.

Bottom up and top down

There are frequently "top down" Vs. "bottom up" arguments in relation to SOA and architecture in general. The two views go something like this;

- "This top down stuff ignores the reality of what's on the ground and how we operate – we just need to put WSDL interfaces on our applications".
- "The bottom up approach just digs you deeper into a technology focused solution with no connection to operation needs".

Could they both be right? An architecture, particularly one with multiple concerns has to address both needs. Both the top-down business needs and the bottom-up realities should provide input into an architecture. The architecture is the place where these needs are understood and reconciled to design the best solution. This solution may have technical inefficiencies demanded by the business or business inefficiencies demanded by the legacy systems, all of these are input to the puzzle. The quality architecture is then the synthesis of the requirements from multiple perspectives into a consistent design.

If there is an existing system that can have a service retrofitted that serves a community, go for it – but understand that service in terms of that community and who it is serving. This then brings the legacy into the architecture instead of making a separate effort. The top-down view may then inform that service, suggesting features that may be needed or others that serve no purpose.

The social and political process

The process of creating an SOA is more of a social process than a technical one. The task is to understand the requirements of collaboration, the communities, the information and the mutual needs. There are frequently disagreements over terms, where responsibilities are assigned, the future of existing systems, build, vs. Buy, what needs to be common Vs. specialized, how much to invest in transformation, etc. The time to reach an architecture by consensus or through authority is the primary activity. Creating the technical interfaces follows easily once this task is done.

The manageable approach to getting early value while allowing the inevitable political process to continue is, again, separation of concerns. By focusing on manageable collaborations that can be in0plemented with a few services there can be both early success and a path to a more extensive architecture. "Big bang" approaches are not recommended. The power of combining SOA and MDA is that the architecture evolves in a controlled way over time, with ever increasing value.

Components & Viewpoints of the SOA

The components and viewpoints of the SOA have been spelled out in this paper but are summarized here, the following are required for most, but not all SOA specifications;

- Collaboration, Roles & Interactions
- Process
- Information
- Components
- Technical Infrastructure
- MDA mapping
- Metadata Management

SOA Technical Infrastructure

A number of basic principles have been defined for implementing the Enterprise Collaboration Technical Architecture on an Enterprise Service Bus. In this paper we will focus on the requirements of the technical infrastructure rather than attempt to specify a complete infrastructure.

Loosely coupled independent roles

The architecture will be based on the concept of business roles that need to collaborate, and the implementation of the roles and collaborations between those roles using technology. The roles in the collaboration architecture must have their contract of interaction well defined and complete. There should be no coupling between roles other than that defined in these contracts.

(Note - in some circumstances, roles will be coupled together when implemented. Where integrated vendor suites are used, a number of roles may well be effectively grouped together into a “super-role”. Some existing “legacy” systems may present several roles with tight back-end coupling. For practical and economic reasons these situations may continue, but these should be de-coupled where practical).

Coupling of business process

While the roles are independent, the business processes in which they are participating must be well defined across all participating roles, like any contract among independent parties.

Service and event based architecture

The architecture will be based around the concept of re-usable services and service specifications, both at the business (computational) level and at the engineering (infrastructure) level. One special point to note here is the need for a rationalized set of business service definitions, which are easily accessible and searchable. The concept of

services applies to one-to-one interactions where the initiating roles need are aware of the identity of the other role. In cases where one to many or anonymous interactions apply, event publication and subscription mechanisms will be supported.

Contract of interaction

Due to the requirements of loose technology coupling and high business cohesion, the contract of interaction becomes architecturally significant. The semantics of this contract must be sufficient for the business and technology requirements. The maintenance of such contracts becomes critical and highlights the importance of repositories for management and access to them.

Thus the technical infrastructure must include repository capabilities such as may be found in MOF, ebXML or UDDI.

Technology independence

The interaction media and execution platform technology and product set are an important decision, but it would be naive to assume that any such technology choice will remain constant or may be imposed on all parties of a business process. There is substantial and long-term investment in defining, implementing and integrating business processes and the support technology. The architecture must enable technology-independent specification of collaborations that are bound to the technology as late as possible in the development process.

Open Standards

Related to the previous point, the architecture and related technology choices need to be based on open standards as much as possible and practical.

Distribution Transparency

Business components should not know whether they are distributed or not. Calls out from the components should all be “local” from the components perspective (they may be to a proxy, which would then send the message or execute a remote call as appropriate)

Location Transparency

Individual components within the architecture should not need to be aware of the physical location of other components.

For these reasons the specification and implementation of business logic components should be as independent as possible from specific media, data formats, middleware or platforms.

Providing for technology independence will allow more flexibility over time, a longer lived systems asset, greater freedom in integrating new business partners and a capability to withstand both business and technology change over time.

Characteristics of an SOA Enterprise Service Bus

This section contains a brief summary of some of the main characteristics of the ESB. Each item is covered in more detail in subsequent sections.

1. Integration of business units (E.G. Analysts, Information Providers (Internal and External) , Information Consumers (Internal and External) and applications will be provided based on the modeling and implementation of collaborations as specified in the Component Collaboration Architecture (CCA), a subset of the OMG Enterprise Collaboration Architecture (ECA), which is part of the UML Profile for Enterprise Distributed Object Computing (EDOC). This specification is intended to be technology neutral and thus should support any infrastructure selected.
2. Collaborations will be based on large-grain document exchange between actors playing roles in a collaborative business process. These collaborations will include synchronous, asynchronous and event (pub/sub) interactions but asynchronous interactions are the primary and preferred mechanism. Document information will be encoded in XML, also making use of WSDL/SOAP packaging where appropriate. Transport will be over HTTP(S) or using a message service (e.g. IBM MQ or MSMQ) as appropriate. *Note: XML is a technology of choice today but the architecture would allow the automated substitution of any middleware technology.*
3. The collaboration architecture will be expected to be able to support long-running business processes, but each interaction will be an atomic transaction.
4. Implementation of services will be secure and robust. Application and role-based security will be supported. Application servers will support reliable messaging, load balancing, logging and fail-over.
5. The overall architecture is based on industry standards as far as possible, however these are developing and fairly immature with respect to collaborations. The basic set of standards being followed is based around the concept of “web services” (XML, SOAP).

In general, discussion will avoid mention of specific products. However, there are a small number of assumptions that are made with respect to technologies that the architecture must be able to support. Examples are that custom components will be able to be built using Java (J2EE Servlets or EJBs) or using the .NET framework; Message Queuing will be able to be supported by JMS, IBM MQ or Microsoft MSMQ.

One additional concept that may also be referred to throughout this documentation is that of a “Business System Domain”. This is defined as “a managed set of services and resources.” It is characterized by the following:

shared resources (which are in a consistent state with respect to each other)

has defined boundaries

provides a defined QOS, reliability, performance

consistent, defined set of contracts between the elements

The purpose of identifying this concept is that it provides a scope for an individual set of work or discussions. This does not necessarily imply a fixed hierarchy, domains can legitimately be defined in different ways for different purposes, e.g. naming, security or business processes. A business systems domain is also a coherent set of technologies working together.

How a SOA Achieves Business Value

Given the SOA approach leveraged with MDA there are multiple advantages to both a more flexible system and enterprise.

- The separation of concerns between providers and consumers of services allows more flexible and better integrated while allowing more autonomy with how the services is realized or used. Processes and technologies can be changed without impacting their "contract" with internal or external stakeholder.
- Capabilities can be in-sourced, outsourced, centralized or distributed based on the best business decision.
- Changes driven by business or technology can change more independently
- The business model can have a direct impact on and even be part of the specification of the technologies that assist the business.
- New, legacy and COTS solutions can be intermixed and change over time providing for a smooth transformation strategy.
- It becomes easer to support internal and external collaboration, joint missions and virtual enterprises.
- Interoperability between systems is not "added on", it is an inherent capability of the SOA architecture.
- Automated development and integration of service components can be partially automated, drastically reducing development, integration and test time.
- Architectures are no longer suggestions to be ignored, but specifications to be implemented – putting the enterprise in control.
- Models can be simulated prior to implementation and deployment to validate the business intent.

[i] "Business" as used in the document is a very general term for the purpose of an organization – it is inclusive of government, defense and commercial concerns..

[ii] An "actor" is any participant in a community such as a person, organization or system.

[iii] OMG & MDA – www.omg.org/mda

[iv] MDA – Model Driven Architecture