SOA Practitioners' Guide
Part 2
SOA Reference Architecture

Painting by: Surekha Durvasula

Contributing SOA Practitioners

Surekha Durvasula, Enterprise Architect, Kohls

Martin Guttmann, Principal Architect, Customer Solutions Group, Intel Corp

Ashok Kumar, Manager, SOA Architecture, Avis/Budget

Jeffery Lamb, Enterprise Architect, Wells Fargo

Tom Mitchell, Lead Technical Architect, Wells Fargo Private Client Services

Burc Oral, Individual Contributor

Yogish Pai, Chief Architect AquaLogic Composer, BEA Systems, Inc.

Tom Sedlack, Enterprise Architecture & Engineering, SunTrust Banks, Inc.

Dr Harsh Sharma, Senior Information Architect, MetLife

Sankar Ram Sundaresan, Chief Architect e-Business, HP-IT

Reviewers

Steve Jones, CTO Application Development Transformation, Capgemini Group

Prasanna Deshmukh, Director of Architecture, WebEx Communications

Noam Fraenkel, CTO IT, Mercury Interactive

Ashok Nair, Management Systems Analyst, EAI, Information Technology Services, City of Calgary

Jeff Pendelton, Executive Director, SOA Alliance

Annie Shum, VP SOA Strategy, BEA

Brenda Michelson, Principal Consultant and Analyst, Elemental Links, Inc.

George Paolini, Consultant, Georgepaolini.com

The authors would like to acknowledge the many organizations and individuals that contributed portions of this document, performed substantial editing, or who provided reviews and feedback. In addition, the authors would also like to thank BEA Systems, Inc. for providing the infrastructure and the platform for developing and presenting this guide.

Table of Contents

TABLE OF CONTENTS 3

1 ABOUT THIS DOCUMENT 5

1.1 ABSTRACT 5
1.2 INTENDED AUDIENCE 5
1.3 BENEFITS OF THE SOA PRACTITIONERS' GUIDE 5
1.4 SOA PRACTITIONERS' GUIDE: PARTS 6

2 SOA – REFERENCE ARCHITECTURE 7

2.1 DEFINITION 7
2.2 SOA REFERENCE ARCHITECTURE APPROACH 7
2.2.1 SOA FOUNDATION 7
2.2.1.1 Business Architecture 7
2.2.1.2 Infrastructure Architecture 9
2.2.1.3 Data Architecture 11
2.2.1.4 Information Architecture 11
2.2.1.5 Architectures that Complement SOA 12
2.2.2 ENTERPRISE SOA MATURITY MODEL 15
2.2.2.1 Web Application Development Stage 15
2.2.2.2 Develop Composite Applications 17
2.2.2.3 Automate Business Processes 18
2.3 SOA REFERENCE ARCHITECTURE 19
2.3.1 WEB APPLICATION TIER 19
2.3.1.1 Packaged Applications 19
2.3.1.2 Custom Applications 21
2.3.1.3 Portal Services 26
2.3.1.4 Enterprise Infrastructure Services 27
2.3.1.5 Enterprise (Role-Based) Portal 28
2.3.2 SERVICE TIER 29
2.3.2.1 Service Bus 29
2.3.2.2 Service Registry 31
2.3.2.3 SOA Repository 32
2.3.2.4 Service Manager 33
2.3.2.5 Shared Data Service 34
2.3.2.6 Business Process Management 41
2.3.3 SOA FRAMEWORKS 42
2.3.4 APPLICATION TIER 44
2.3.4.1 Legacy Applications 44
2.3.4.2 Mainframe Applications 44

2.3.4.3 Enterprise Application Integration 44
2.3.5 ENTERPRISE SECURITY 45
2.3.6 BUSINESS SERVICE MANAGEMENT 46
2.3.6.1 Typical Current State of Monitoring 46
2.3.6.2 Future State of Business Service Management 47
2.3.7 SERVICE ORIENTED INFRASTRUCTURE (SOI) 48
2.3.7.1 The Business Value of Services Oriented Infrastructure (SOI) 51
2.3.8 MAPPING SOA REFERENCE ARCHITECTURE TO ENTERPRISE SOA MATURITY MODEL 52

1 About This Document

1.1 Abstract

SOA is relatively new, so companies seeking to implement it cannot tap into a wealth of practical expertise. Without a common language and industry vocabulary based on shared experience, SOA may end up adding more custom logic and increased complexity to IT infrastructure, instead of delivering on its promise of intra and inter-enterprise services reuse and process interoperability. To help develop a shared language and collective body of knowledge about SOA, a group of SOA practitioners created this SOA Practitioners' Guide series of documents. In it, these SOA experts describe and document best practices and key learnings relating to SOA, to help other companies address the challenges of SOA. The SOA Practitioners' Guide is envisioned as a multi-part collection of publications that can act as a standard reference encyclopedia for all SOA stakeholders.

1.2 Intended Audience

This document is intended for the following audience:

- Business and IT leaders, who need to start and manage an SOA strategy across the enterprise/LOB
- Enterprise Architects who need to drive the vision and roadmap of the SOA program and the architecture of each implementation that falls under it
- Program Managers who need to manage a portfolio of sub-projects within an overall SOA business strategy
- Project Team Members, who need to map dependencies and develop a timeline that meets the business expectations
- Vendors who provide solutions and tools for new business capabilities to the business and IT
- Standards bodies which need a better understanding of use cases of how business and IT plan to leverage technology to meet their objectives.

1.3 Benefits of the SOA Practitioners' Guide

This document helps readers to:

- Learn from others: Early adopters of SOA share their best practices, insights, and views on the state of SOA adoption across the industry
- Compare alternatives: Identify and define the key technology components of SOA to establish a baseline reference for comparison of options
- Improve collaboration: A common language clarifies the nature of SOA components defined in this document
- Accelerate implementations: This guide defines the services lifecycle along with the requirements, recommended tools, and best practices for each of the stages.
- Understand the value of standards: This document recommends standards for aspects of SOA
- Avoid potential risks: The guide identifies some problem areas not yet addressed by the vendor community.

1.4 SOA Practitioners' Guide: Parts

There are three separate parts that make up the SOA Practitioners' Guide.

Part 1, *Why Services-Oriented Architecture*? provides a high-level summary of SOA.

This is Part 2: *SOA Reference Architecture*. It provides a worked design of an enterprise-wide SOA implementation, with detailed architecture diagrams, component descriptions, detailed requirements, design patterns, opinions about standards, patterns on regulation compliance, standards templates, and potential code assets from members.

Part 3: *Introduction to Services Lifecycle* provides a detailed process for services management though the service lifecycle, from inception through to retirement or repurposing of the services. It also contains an appendix that includes organization and governance best practices, templates, comments on key SOA standards, and recommended links for more information.

2 SOA – Reference Architecture

2.1 Definition

The SOA reference architecture defines an ideal target architecture for an enterprise or LOB. Some also refer to this as the “future state” or “future vision” of the enterprise. The SOA reference architecture is key to constructing a roadmap from the current state to the target state.

2.2 SOA Reference Architecture Approach

One needs to understand two aspects of SOA to be able to develop the reference architecture:

- The three SOA foundation components
- Enterprise SOA maturity model

2.2.1 SOA Foundation

The SOA foundation components are illustrated in the figure below.

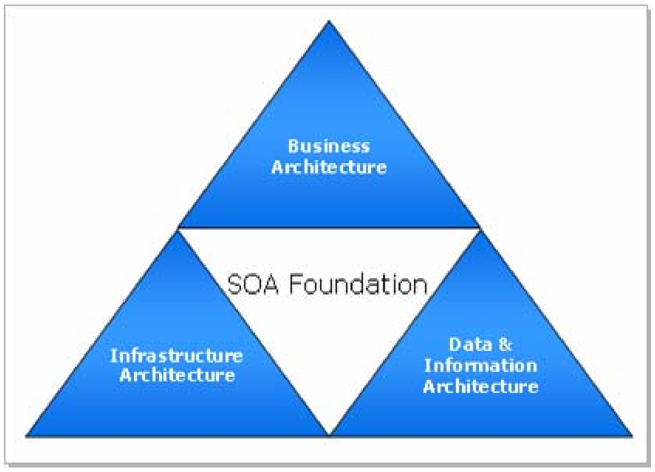

Figure 1: SOA Foundation

2.2.1.1 Business Architecture

Business architecture describes the business strategy, objectives, priorities, and processes to be supported by the SOA. An SOA is only successful if it delivers on the business architecture. Reuse of business processes provides higher ROI than the potential reuse of infrastructure or data components.

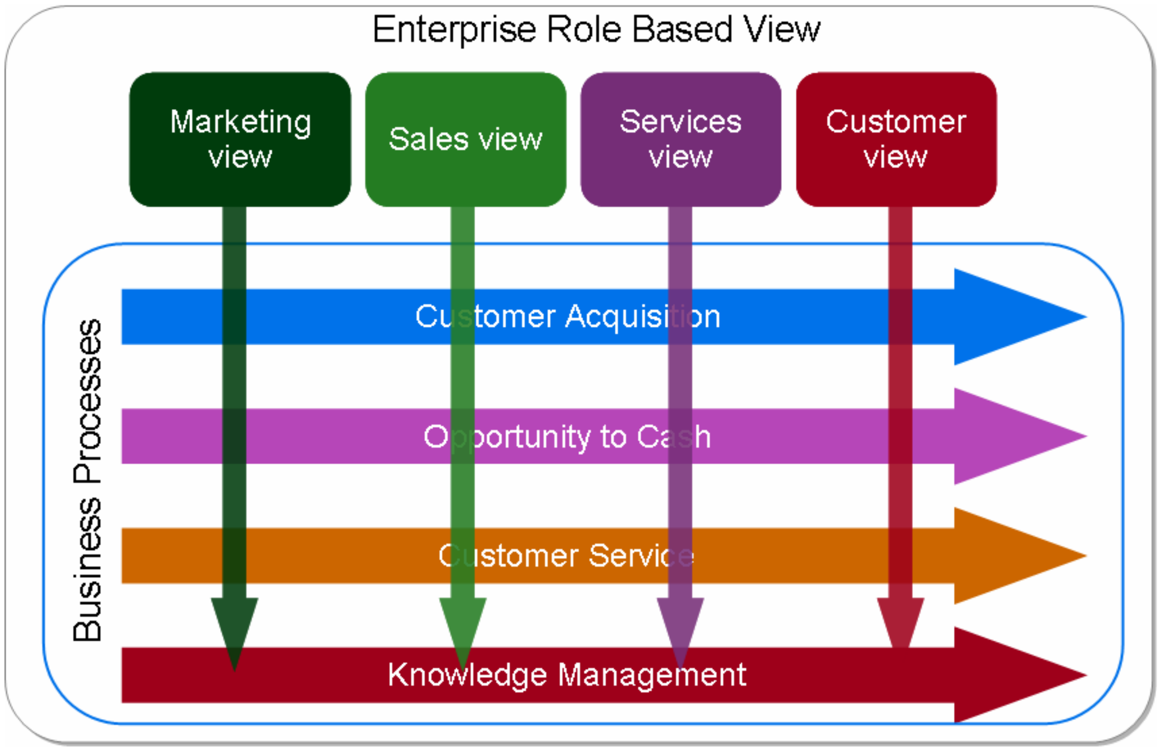

Figure 2: Focus Areas for Business Architecture

Some of the best practices for developing the business architecture include:

- Review the current system specification and the underlying technology
- Map these to the business strategy to identify gaps
- Review the horizontal (business processes) and vertical (role-based view) requirements
- Prioritize the application (services) portfolio to provide these capabilities
- Standardize the user experience across applications
- Define business policies on key aspects such as application and data access and regulatory compliance

Additional reference: Developing a Business Architecture View (TOGAF)
http://www.opengroup.org/architecture/togaf8-doc/arch/p4/views/vus_bus.htm

2.2.1.2 Infrastructure Architecture

This is the engine that enables SOA. It should address all the aspects of the scalable infrastructure from networks, enterprise servers, data centers, and firewalls, to application infrastructure, security, monitoring, and middleware.

The architecture team is responsible for identifying the infrastructure components—the architecture building blocks—required to provide the business capability.

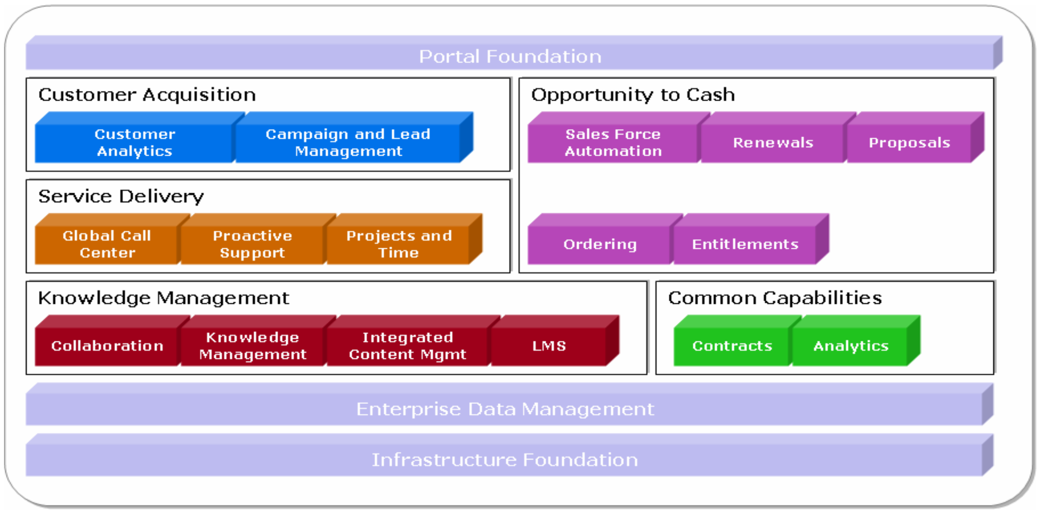

Figure 3: Example of Architecture Building Blocks

The above diagram is an example of the of the architecture building blocks required to provide the business capability with the primary focus being business process. At the same time, the infrastructure architecture also needs to include role-based portal requirements.

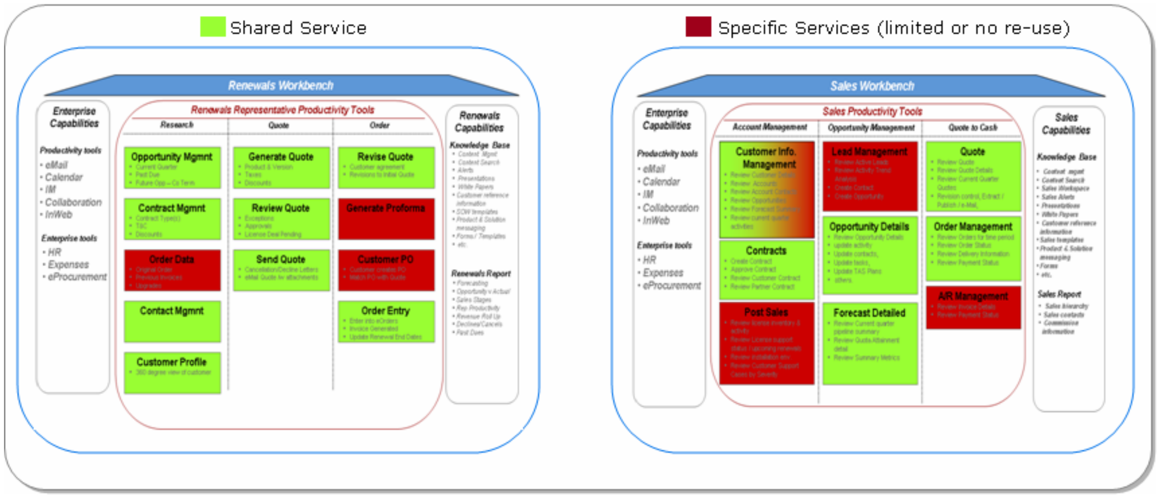

Figure 4: Example of Role-Based Portal

Infrastructure needs to combine architecture building blocks and role-based portals in order to enable:

- High reuse of common services
- Reuse of infrastructure and foundational components
- Reduction in time needed to develop new capabilities.

Infrastructure Components	**Description**
Custom application frameworks Data services Logging services Exception handling Audit service Search framework Notification framework	Common components required for developing custom applications
Security Authentication Authorization Single-Sign-On (SSO) Delegated administration	Security framework that could extend to the enterprise level
Shared data services Master data management Data profiling Data quality service Data matching Data validation Data modeling Analytic services	Data services to support SOA
Portal services Common look and feel Personalization Reporting Localization Web traffic monitoring	Portal services for consistent user interaction and ability to leverage WRSP
Enterprise infrastructure services LDAP E-mail Collaboration (Chat/IM/Whiteboard) Content management Integrated structured and unstructured search	Common services required enterprise wide
Master data management Customer data integration Product master	Capabilities required to provide ability to execute the business processes across applications and business silos

Additional details on the infrastructure components are defined in the SOA reference architecture section.

Additional reference: Infrastructure Architecture (TOGAF)
http://www.opengroup.org/architecture/togaf8-doc/arch/p4/infra/infra_arch.htm

2.2.1.3 Data Architecture

Data architecture deals with the logical and physical modeling of the data as well as data manipulation and data quality. The SOA reference architecture covers each of these areas at length by providing approaches, requirements, and design patterns wherever possible.

2.2.1.4 Information Architecture

Information architecture models key concepts and events for a given business process. The business concepts represent any business entities that need to be exchanged by the processes or applications that support the enterprise.

Information modeling creates canonical models described by XML schemas. Canonical models are very crisp definitions of the business concept attributes, including attribute relationships, value enumerations, value patterns, sequencing of the attributes on the XML document, and whether an attribute is mandatory. SOA uses canonical models to represent both the request and response documents traded by the service and also the content payload that is returned to a consumer.

Canonical models that are exchanged by a business application are typically business concepts. For example, “Candidate Product List” may be returned in response to a product catalog search. Canonical models that are sent out or published by a business process are typically business events. For example, “Purchase Order Approval” business event may be published by the Supply Chain Management business process and will need to be subscribed to by the supplier.

Another aspect of information architecture is the definition of key performance indicators (KPIs) that capture business-level information. KPIs help an organization define and measure progress toward organizational goals. KPIs are abstractions of information that report value extracted from a business process.

Additional reference: Wikipedia definition on Information Architecture
http://en.wikipedia.org/wiki/Information_architecture

2.2.1.5 Architectures that Complement SOA

While SOA itself is an architectural style, it is important to understand that other architectures are relevant to a successful SOA strategy. The following sections provide an overview of some architectures that may come in to play in an SOA strategy.

2.2.1.5.1 Model-Driven Architecture (MDA)

Object Management Group's (OMG) model-driven architecture (MDA) (http://www.omg.org/mda/) is an approach that employs platform-agnostic models to understand and manage all aspects of a functioning enterprise: the "what, how, where, when, by whom and the why." (http://www.omg.org/mda/mda_files/09-03-WP_Mapping_MDA_to_Zachman_Framework1.pdf) The primary goals of MDA are portability, interoperability, and reusability, and bridging the gap between business and IT using models

MDA helps reduce the chasm between business and IT by developing computation-independent models (CIM), which strip out the computation-specific details so subject matter experts can more readily understand the model.

Organizations are better off developing platform-agnostic models, because these help organizations retain their intellectual property and also survive the slippery slope of technology platform implementation. Platform-independent models (PIM) with sufficient degree of platform independence support this, and also help decision makers understand the proposed process flow. Once the overall architecture is agreed, these can be transformed into platform-specific models (PSM) using standard transformation rules and exchanged using a standard interchange format.

MDA is based on well-established OMG standards that are also ISO standards, including:

- Unified Modeling Language™ (UML http://www.uml.org/), the ubiquitous modeling notation used and supported by major companies in the software industry that allows business, architectural, structural, and behavioral modeling. In addition, UML Profiles can be developed as "specialization" for particular areas of computing such as EJBs, Web services, Ontologies, Scheduling, Testing, and Information modeling.
- Meta Object Facility (MOF http://www.omg.org/mda/specs.htm#MOF) OMG's foundation specification for modeling languages and transforming models. MOF defines modeling languages (metamodels) and the integration, interchange and management of models.
- XML Metadata Interchange (XMI http://www.omg.org/technology/documents/modeling_spec_catalog.htm#XMI), the standard for storing and exchanging models across tools using XML.
- Common warehouse metamodel (CWM http://www.omg.org/technology/cwm/) is a specification that describes metadata interchange across data warehousing lifecycle, business intelligence, knowledge management, and portal technologies. CWMI will morph into a broader information management metamodel (IMM) that will allow use of UML for data and XML modeling as well. (http://adtf.omg.org/adptf_info.htm#RFIs,RFPs).

2.2.1.5.1.1 MDA as the Foundation of SOA

Due to its maturity and rich functionality for modeling all aspects of an enterprise and software, MDA is well suited to be a foundation for the SOA lifecycle. Many other OMG standards, coupled with MDA, enable a business- and architecture-driven approach to SOA. For an overview, please refer to: http://www.omg.org/attachments/pdf/OMG-and-the-SOA.pdf and http://soa.omg.org/. In addition, MDA can enable development of an integrated SOA registry/repository that supports discovery of services and their components, as well as mapping to business processes and objectives.

Others have commented on the value of MDA for SOA. The following links provide third-party perspective on the value of MDA:

- Model-driven SOA http://www.opengroup.org/events/q405/mdasoa.pdf
- Dr. Chris Harding, The Open Group's SOA working droup director, says in this article, "The combination of SOA and OMG's MDA is what is needed for real agility; Model-Driven Architecture (MDA) is a well-developed concept that fits well with SOA….Can it be made simpler and more accessible, to become a widely used enabling technology for SOA? With a semantic approach, the answer may well be "Yes"." http://www.ebizq.net/topics/soa/features/6639.html?&pp=1
- Dr. Ali Arsanjani, Chief Architect, SOA and Web services center of excellence, IBM, says in this article, "A service-oriented modeling approach provides modeling, analysis, design techniques, and activities to define the foundations of an SOA. It helps by defining the elements in each of the SOA layers and making critical architectural decisions at each level." http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/ , http://www.webservices.org/categories/enterprise/strategy_architecture/how_to_identify_specify_and_realize_services_for_your_soa/(go)/Articles

OMG is now working on a software services profile standard RFP that will enhance UML for modeling services, such as services specification and contracts. For more details, check out the following: http://soa.omg.org/SOA%20ABSIG%20RFPs,%20RFIs.htm

2.2.1.5.2 Event-Driven Architecture (EDA) and Complex Event Processing (CEP) complement SOA

While connecting services in a typical SOA environment occurs in a linear, predictable sequence, EDA allows for multiple, less predictable, asynchronous events to happen in parallel and trigger a single action. In an EDA a notable 'thing' happens inside or outside your business. An 'event system' senses and collects these events and correlates patterns which are disseminated to all interested parties (human or automated) optionally via services. The interested stakeholders evaluate the event(s), and may respond by invoking a service, triggering a business process, or publishing or syndicating further information. EDA helps correlate complex relationships of events based on past trends and future predictions. While EDA is often considered as subset of SOA, it is more correct to view it as complementary to SOA.

For additional information on EDA and its relationship with SOA please follow the article links below:

- Event-driven architecture poised for wide adoption http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,81133,00.html
- Primer: Event-driven architecture http://www.baselinemag.com/article2/0,1540,1606126,00.asp
- Event-driven architecture vs. publish-subscribe systems http://www.developer.com/design/article.php/3499031
- Event-driven services in SOA http://www.javaworld.com/javaworld/jw-01-2005/jw-0131-soa.html
- Combining service-oriented architecture and event-driven architecture using an enterprise service bus http://www-128.ibm.com/developerworks/webservices/library/ws-soa-eda-esb/index.html
- Event-driven SOA is just part of the EDA story http://www.psgroup.com/research_michelson.aspx

2.2.1.5.2.1 Complex Event Processing (CEP) and SOA

Complex event processing (CEP) is an emerging technology that will help companies develop and manage business activity monitoring (BAM), enterprise application integration, network and systems security, and business processes.

According to Professor David Luckham, at Stanford University, an authority on CEP, the goal of CEP is to deliver understandable information about what's happening in IT systems. That information, in turn, can be used for a variety of purposes, such as detecting unusual activity, improving security and recognizing advantageous scenarios in CRM and supply-chain systems.

"The events in IT systems contain untapped information, and CEP lets you extract it and use it in ways you want to," says Luckham. He predicts that CEP will start being incorporated into Web services, middleware, and application servers in 2005. By 2008, he foresees the emergence of CEP standards, languages and complex event-pattern search engines, and ubiquitous adoption of CEP by 2012.

For additional information on CEP, please refer to:

http://www.complexevents.com/ and http://complexevents.com/?page_id=59

2.2.1.5.2.2 What Problems are solved by MDA, EDA-SOA, CEP

By leveraging MDA, EDA, SOA, and CEP, organizations can develop model-driven, agile “sense and respond” systems that can (in near real-time) detect events of business value and trigger services to manage those events. Understanding which applications are optimized for each architecture and how they interface with each other allows for improved architecture. In addition, such an architectural approach can enable end-to-end management of business processes and supported functions.

The standards for EDA and its relationship with SOA and CEP are still in development. OMG SOA SIG has started a sub-group to focus on this aspect of SOA using MDA as a foundation. (http://soa.omg.org/SOA%20ABSIG%20RFPs,%20RFIs.htm).

2.2.2 Enterprise SOA Maturity Model

The SOA maturity model helps enterprises develop a roadmap to achieve their target state.

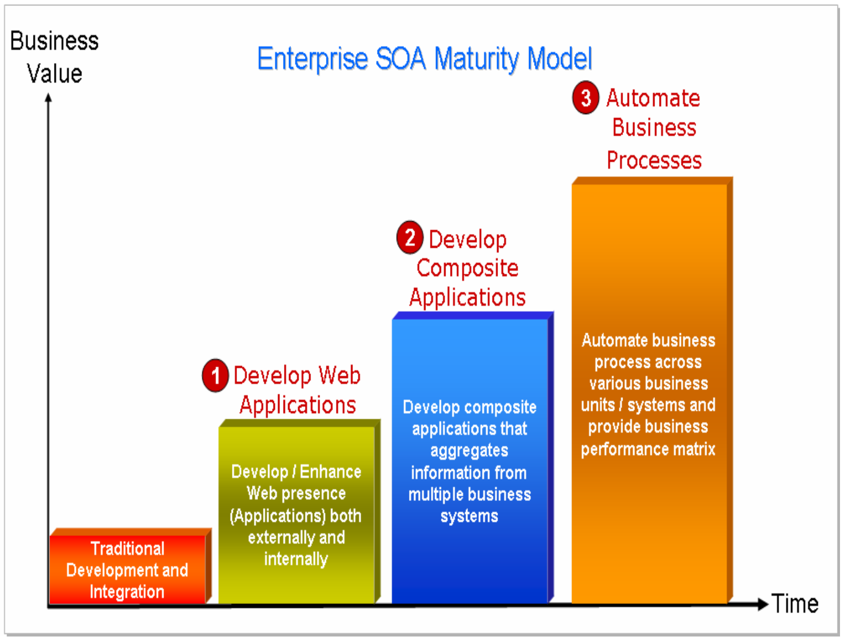
Business Value
Enterprise SOA Maturity Model
3 Automate Business Processes
2 Develop Composite Applications
1 Develop Web Applications
Traditional Development and Integration
Develop / Enhance Web presence (Applications) both externally and internally
Develop composite applications that aggregates information from multiple business systems
Automate business process across various business units / systems and provide business performance matrix
Time

Figure 5: Enterprise SOA Maturity Model

The above diagram illustrates the stages of the enterprise SOA maturity model.

2.2.2.1 Web Application Development Stage

At this stage, teams focus on providing rich client and browser-based business solutions to both internal and external users. They might choose to roll out web-enabled CRM, ERP, or custom applications that support connected and occasionally disconnected operations. In addition, IT organizations typically deploy enterprise-based solutions and services such as content management, search, instant messaging, blogs, Wikis, discussion forums, and white boards.

2.2.2.1.1 Business Requirements

Typically most enterprises would have already deployed external web sites as well as multiple internal web sites and applications to support the diverse needs of each of the business units. The first step is to standardize, share, and integrate these siloed solutions through an infrastructure that provides a common look and feel. This makes it easier for customers, partners, and employees to find the information they are seeking.

During this phase, the team should focus on:

- Unifying user experience on the external site, making it easy for potential users, partners, customers, and analysts to find information they need
- Standardizing the look and feel across all sites (internal and external) as well as across processes and procedures for publishing content

- Creating one my<company name> such as http://my.company.com, site for all employees, contractors, partners, customers to personalize services and content
- Providing secure access to confidential information for all internal and external sites
- Providing a highly reliable, available, and scalable environment
- Enabling the site operations with AJAX to increase performance and user experience.

2.2.2.1.2 Key Challenges

The key challenges for this phase include development of:

- Application support escalation path
- Support for numerous parallel activities
- Leadership and technical quality of team
- Physical environment for development through production, with release management processes and skilled staff resources
- Dedicated production support processes and staffing
- Hosting.

2.2.2.1.3 Exit Criteria

The team can consider this phase complete when:

- External web site is up and running
- Portal front end has been developed for one or more packaged applications
- One or more custom applications is accessible through the portal site
- Most enterprise services have been deployed
- Business users can request information from multiple applications
- Establishment is complete for the program management office (PMO) and and LOB governance model for deploying application portals
- Business has confidence in delivery timeline and consistently approaches the program office for all major initiatives.

2.2.2.1.4 Success Factors

This phase is successful if:

- Business involvement at LOB level is high
- Sponsorship/executive oversight has been established for all composite applications
- Web-based applications can be rapidly developed and delivered
- Project management is in place, and the team has leadership and a sense of urgency and direction
- Processes have been standardized across the LOB for development, deployment, and status reporting
- The team has developed identified and created experienced resources.

2.2.2.2 Develop Composite Applications

Composite applications aggregate and provide information and data from a variety of sources and channels, and make them available to internal and external users as appropriate. Enterprise 2.0 services can provide appropriate levels of SLA.

2.2.2.2.1 Business Requirements

The business requirement is for IT to adapt to changing business needs. Several business units may approach IT to develop custom applications, enhance their own branding, increase productivity, or provide additional information to their customers, partners, or employees.

Business requirements may include:

- Branding and exposing multiple applications through the portal
- Accessibility of information from multiple applications
- A web-based desktop for users
- Personalized service based on roles and responsibility of the user
- A single standardized look and feel, which can reduce user training requirements
- Reduced maintenance costs from standardizing on one platform
- Reduced operations and support cost, to enable IT to deploy scarce resources for new functionality.

2.2.2.2.2 Key Challenges

The key challenges for this phase include development of:

- Application support escalation path for shared services
- Support for numerous parallel activities across multiple LOBs
- Governance for shared services
- Leadership and technical quality of team
- Physical environment for development through production, with release management processes and skilled staff resources
- Dedicated production support processes and staffing
- Hosting.

2.2.2.2.3 Exit Criteria

This phase is complete when:

- A Program Management Office (PMO) has been created that spans multiple LOBs, and an enterprise-wide governance model for deploying shared services has been established
- Business has confidence in delivery timelines, and uses the program office for all major initiatives
- Multiple deployed application portals leverage the SOA foundation
- Business units debate integration timeframes for applications or data.

2.2.2.2.4 Success Factors

This phase can be considered a success when:

- Business involvement and sponsorship, including executive oversight, is in place for all composite applications
- The team has developed a rapid development and delivery approach
- Project management has developed leadership, a sense of urgency, and direction
- Processes for development, deployment, and status reporting have been standardized across the enterprise for shared services
- The company has developed experienced resources in agile (parallel development) methodology.

2.2.2.3 Automate Business Processes

This is the stage where the applications, data, and infrastructure help users to perform their roles effectively by providing the right information at the right time. At this stage, the enterprise can start achieving higher ROI by consolidating multiple business systems into a single system. Business organizations should now be ready to abandon their point solutions and transition to the target state of end-to-end business process management.

2.2.2.3.1 Business Requirements

The basic requirements for this phase are as follows:

- Business is interested in standardizing the business process across the enterprise
- Infrastructure is consolidated on standards-based technolog, reducing costs
- Standardized business processes are used globally, but allow for some localization

2.2.2.3.2 Key Challenges

The key challenges for this phase include:

- Accomplishing business and IT transformation
- Establishing appropriate governance and organization models
- Implementing packaged applications for perceived short-term gain.

2.2.2.3.3 Success Factors

This phase is successful when:

- Business involvement and sponsorship and executive oversight enable both business and IT transformation
- A dedicated team focuses on business processes
- Business process is the primary focus for the enterprise
- Loosely coupled business services are assembled to automate business processes and can be recombined to provide new business functionaliy.

2.3 SOA Reference Architecture

The following diagram illustrates the SOA reference architecture, which is made up of a web application tier, service tier, application tier, and infrastructure tier.

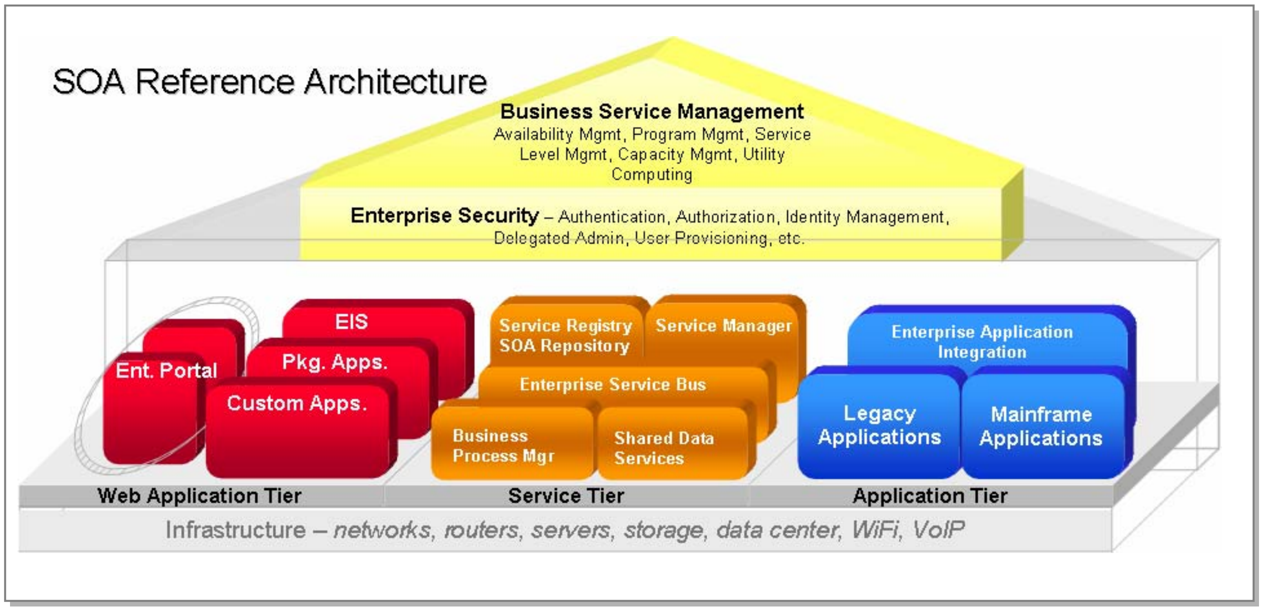
SOA Reference Architecture
Business Service Management
Availability Mgmt, Program Mgmt, Service Level Mgmt, Capacity Mgmt, Utility Computing
Enterprise Security – Authentication, Authorization, Identity Management, Delegated Admin, User Provisioning, etc.
EIS
Ent. Portal
Pkg. Apps.
Custom Apps.
Service Registry SOA Repository
Service Manager
Enterprise Service Bus
Business Process Mgr
Shared Data Services
Enterprise Application Integration
Legacy Applications
Mainframe Applications
Web Application Tier
Service Tier
Application Tier
Infrastructure – networks, routers, servers, storage, data center, WiFi, VoIP

Figure 6: SOA Reference Architecture

Not all IT organizations will need to deploy the entire infrastructure identified in this SOA reference architecture. One of the SOA best practices is to invest in the infrastructure only when it is required to provide business solutions.

2.3.1 Web Application Tier

The primary requirement for this tier is that all the business systems and solutions be accessible from any supported browser. This tier is the user interface or presentation tier and contains business logic for components such as enterprise infrastructure services and applications.

2.3.1.1 Packaged Applications

Typically, enterprises license the “best of breed” packaged applications that meet most of their businesses requirements, and then have their IT organizations and systems integrators tailor the packaged applications to meet their needs. Examples of such packaged applications are customer relationship management, enterprise resource planning, or industry-specific application suites.

Most packaged applications are now based on Internet protocols, which means that users can access many of the functions using any supported browser. Some of the latest applications can expose a limited set of functions as discrete callable services or externally controlled business processes.

Some of the best practices for leveraging packaged applications include:

- Limiting the amount of custom development, making it easier and cheaper to maintain and upgrade
- Attempting to achieve one standard implementation worldwide, thereby reducing integration and maintenance costs
- Leveraging the UI and the business process provided by the packaged applications, wherever possible
- Leveraging published application programming interfaces (APIs) rather than directly accessing the database.

Following are recommended approaches for taking the packaged application through the SOA maturity model:

2.3.1.1.1 Develop Web Applications

- Deploy the latest version of the application that is accessible by any browser; preferably a version that supports appropriate portal standards such as WSRP.
- Expose application services for consumption by custom applications, preferably as web services. This may require an adapter to enable access the application. Some recent versions of applications provide direct access to the application services through integration gateways or web services.
- Provide seamless user experience by incorporating the enterprise look and feel (templates, skins, skeletons, CSS) as well as integrating with the enterprise single sign-on solution.
- Externalize authentication by integrating to the enterprise identity and access manager (typically LDAP).

2.3.1.1.2 Develop Composite Applications

- Identify business objects that could be shared across the enterprise as composite applications
- Send event notifications (triggers) to the composite applications to initiate specific actions
- Modify business processes and user interfaces to enable the composite applications
- Expose additional business services so the composite applications can synchronize with the packaged application.

2.3.1.1.3 Automate Business Processes

- Understand and model business processes to identify opportunities for re-engineering
- Identify re-usable portions of business processes that can potentially be automated by a business process engine
- Expand the number of exposed services and business processes
- Reduce and consolidate the number of applications deployed.

2.3.1.2 Custom Applications

Organizations may choose to develop a custom application, to create a distinct brand and unique experience for their customers and partners. This requires providing a consistent seamless interface to internal and external users. Companies often prefer to develop a custom application rather than customize a packaged application, because:

- Making modifications to the user navigation or user interface for some core transactions is not easy
- As most of the major packaged applications are not based on open or standard technologies, their performance may not scale to the business needs
- Proprietary development model makes it difficult to find resources or rapidly deploy new business capability
- Integration to other technology is not straightforward, resulting in point-to-point integration and possibly poor data quality.

The three options for developing custom applications are:

1. Develop and deploy custom applications on an application server
2. Develop and deploy custom applications by leveraging a portal
3. Develop a thick client either using tools based on open standards or proprietary development tools.

For options 1 and 2, the first step for IT organizations is to determine the approach, infrastructure, and tools for developing custom applications. In addition, IT organizations need to define the governance and organization model to develop the custom solution.

For option 3, thick client custom applications are typically developed using SWING, Visual Studio, or similar tools. Most of these thick clients need to interface with some external systems and the recommended approach would be to leverage open standards such as SOAP, web services, XMPP, or WebDAV instead of directly accessing any external resources. This approach makes it easier for IT organizations to support and upgrade the integration.

2.3.1.2.1 Custom Applications Business Requirements

Most enterprises have already deployed external sites as well as multiple internal sites and applications to support the diverse needs of each of the business units. The first step is to standardize the look and feel of these sites and the infrastructure across the enterprise to make it easier for a customer, partner, or an employee to get the information they are seeking.

The business requirements for this phase include:

- Unify user experience on the external site, making it easy for potential users, partners, customers and analysts to find information that they are looking for
- Standardize the look and feel across all internal and external sites, and the process and procedures for publishing content
- Create one my<company name> site for all employees, contractors, partners, and customers to personalize the services and content
- Provide secure access to confidential information for all internal and external sites
- Provide a highly reliable, available, and scalable environment
- Facilitate branding and accessing multiple applications through a common portal
- Allow users to log in once and gain access to all their services
- Personalize service based on roles and responsibility of the user
- Reduce maintenance cost of maintaining multiple systems and applications by standardizing on one platform or environment

- Standardize on one look and feel to eliminate multiple user training requirements
- Reduce operations and support cost, freeing IT to deploy scarce resources on developing new functionality.

2.3.1.2.2 Custom Applications Architecture Approach

As portals provide a proven set of capabilities in support of the presentation layer, most IT organizations have started standardizing on a portal for developing custom applications.

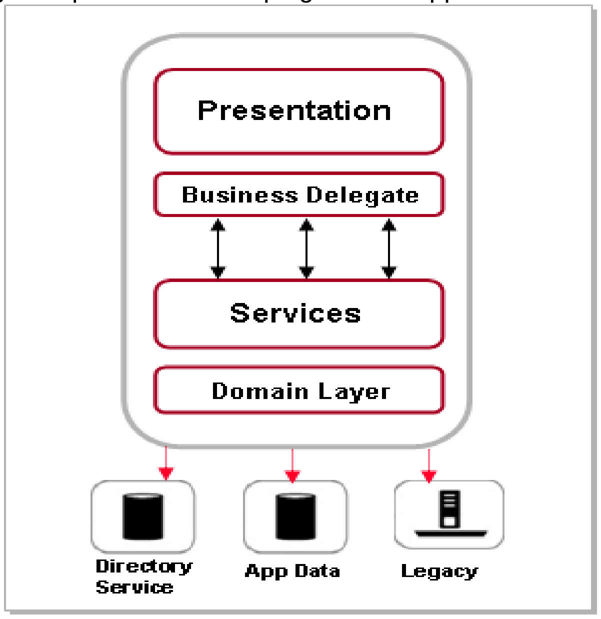

Figure 7: Recommended Custom Application Architecture

This recommended architecture would provide the following benefits:

- Follows SOA principles to promotes re-use at all levels
- Provides capabilities to deliver in weeks not months (once there is a stable framework in place)
- Leverages each product for what it is good at, for example, uses portal for presentation based on entitlements
- Allows business to combine services to deliver new capabilities
- Abstracts the data source and the relationship through a domain layer, thereby minimizing the impact of changes to the source systems
- Loosely couples presentation and the business logic to make it reliable and scalable.

Each of the layers in the proposed architecture plays a specific role:

Presentation layer: a Portal is responsible for handling all presentation services. Portlets drive the user experience where a portlet is a view on an application.

Business delegate layer: business delegates are components responsible for the communication between the presentation and the business layers. They abstract the communication details and complexities involved in making a call to the business layer. This layer includes a model view controller framework that helps users navigate through the web site.

Services layer: the services layer includes the capabilities of the application server. It is composed of stateless functions that expose high-level business functionality. It includes a session facade which is the entry point to the business layer and which abstracts away the details of handling fine-grained business entities from the presentation layers. Most of the business logic can be implemented directly on session facades or on a sub-layer of application objects.

Domain layer: the domain layer also uses the core application server. It is a collection of business entities that defines persistent business concepts. Technologies that handle database storage need to be used in this layer since these components represent persistent states. Entity Beans may be used to implement some of the components of the object model. Alternatively plain old Java objects (POJO) can be used with the help of data access objects (DAO) for persistence. Entity Beans are the preferred mechanism for implementing this layer but a combination of technologies may be required depending on the complexity of the object model.

2.3.1.2.3 Custom Applications Framework Components

Custom application framework components extend services that are inherent in the application server platform. Framework components include:

Data services: the persistence layer provided for the applications. The container management is robust enough to leverage CMP for most simple transactions, but DAO should be offered as an option for handling complex transactions.

Logging services: services used by applications to record and trace errors and activity. Types of message to be logged include debug messages to trace any issues, error or fault logging for diagnostic purposes, and activity logging for audit trail and usage analysis. Every enterprise should standardize the logging services used by applications, ideally leveraging the features provided by JDK 1.4 onwards. If the logging service is generic across the enterprise, it will enable the staff to more effectively determine performance or transaction bottlenecks. Logging services should standardize the mechanism, communicate it to the entire development community within the enterprise, and ensure compliance with the standard. No specific code needs to be developed for this service.

Exception handling: mechanism to manage and communicate exceptions. This is similar to logging services in that the team should leverage standard application server capabilities. The team needs to decide what mechanism to use and communicate it to the entire development community within the enterprise. No specific code needs to be developed for this service but it would be useful if the team provided examples of handling exceptions.

Deployment/Application configuration: document that details configuration. This involves standardizing the mechanism of building and deploying an application in every environment, including development, QA, UAT, staging, and production.

Monitoring: standardization and documentation of monitoring procedures. Since operations staff must monitor platforms and applications and proactively resolve issues, most departments within IT have already identified and deployed a monitoring tool. The challenge is to standardize and integrate their monitoring procedures and technologies to ensure consistent data that encompasses all systems. An enterprise may need to purchase or develop and deploy an additional specialized monitoring tool.

Search framework: shared functionality for searching data. Most portal applications need to present data in a tabular format to the users. Instead of each developer attempting to resolve this problem, a team could develop a "search framework" to be leveraged across applications and portals. The following diagram illustrates an architecture for a search framework.

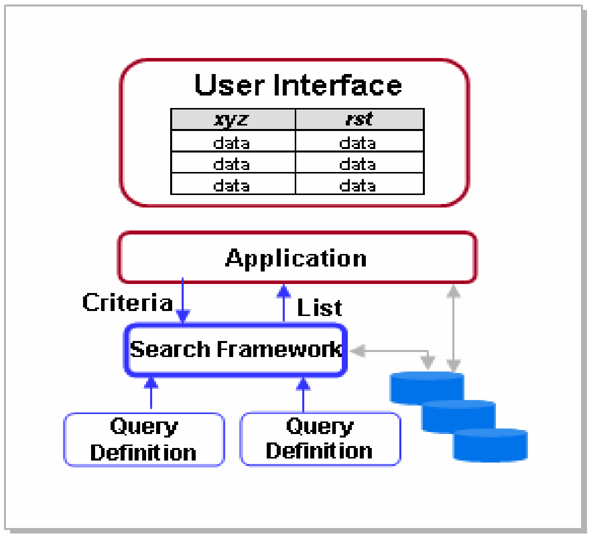

Figure 8: Search Framework

The search framework provides:

- Dynamic query generation based on user input
  - Sort order, joins, etc.
  - Total search results for display purposes
- Consistent mechanism for handling searches
  - Character escaping and wildcard interpretation
  - Pagination
- Abstraction of all database access code from application
  - Criteria used as input
  - Search results required in standards such as java.util.List
- Queries that reside on external files
- Utilities to handle common UI tasks
  - Pagination
  - Criteria persistence.

Notification framework: single notification client to all applications. This should support synchronous and asynchronous interface to the notification engine and also provide capability to send notification through multiple channels.

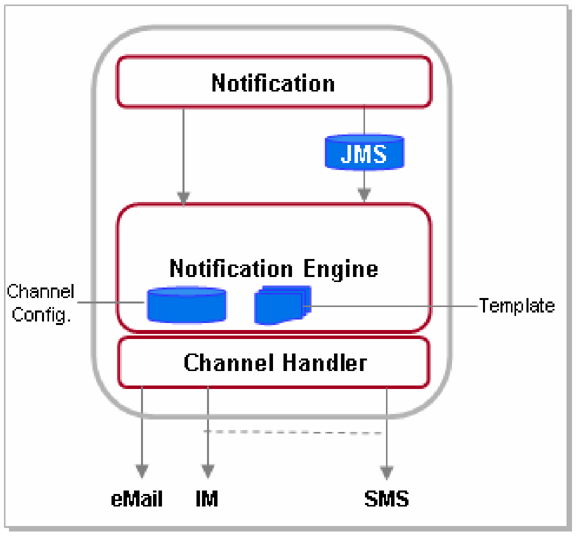

Figure 9: Notification Framework

The interface to the various channels could be developed as required by the business.

Service proxy framework: abstraction of service implementation details. Teams can deploy services either locally or remotely without having to program the calling application with implementation details or location of the service. Instead, the service locater determines the location of the service and calls it in the appropriate fashion. This supports multiple proxies such as EJB, web services, and service bus proxies; additional proxy types can be developed as required. This could also be leveraged by the business delegate to separate the presentation layer from the service layer.

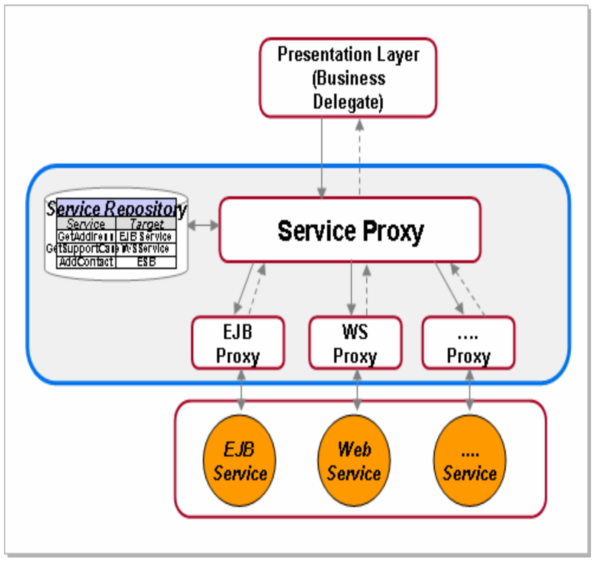

Figure 10: Service Proxy

Security framework: enterprise-wide layer that provides security capabilities. Most application project teams develop their own security layer because current enterprise security solutions do not meet all their business needs. A security framework that supports the client side can reduce, if not eliminate, the need to develop custom security code for each applicaiton. Following are some of the functions a security framework should provide:

- Single sign-on (SSO): capability to log in once and be able to traverse from application to application without having to login again
- Access Control: a set of security features that addresses three main areas:
    - Authentication: determining the identity of the user interacting with the applications
    - Authorization: determining if a user is allowed to perform a particular action
    - Auditing: tracking the actions performed by the users.

Several secondary services are also required such as registration, entitlement granting, and entitlement querying. These features should be provided as a generic framework that can be used and reused by different applications, each with slightly different needs but all having the same basic requirements, including:

- Identity management: Having multiple stores for managing the access control information for a set of applications or services may result in severe management problems. Identity management helps by centralizing the access control management capability, as well as provisioning the users across the enterprise.
- Consolidated user profile: Portals provide this capability to enable the application to extend the base profile. This capability extracts the user profile from multiple data sources, such as the base profile and the application-specific profile from the application-specific repository.
- Registration, delegated administration, provisioning, repository: These are security extensions built on top of access control to meet the application-specific business needs. Alternately, these could be packaged solutions integrated with access control.

Portal products provide most of these capabilities off-the-self. IT organizations will have to develop and support this capability if they do not own a portal for developing custom applications.

2.3.1.3 Portal Services

Portal services manage the presentation tier of the application. As the presentation is generally based on entitlements, there is a need to support this capability.

Presentation: the portal presentation capability provides the skins, templates, skeletons, and style sheets for each of the application teams. This should also include some sample applications to help jumpstart development and leverage the portal navigation capability, both for vertical navigation bars as well as horizontal tabs.

Personalization: the portal provides personalization services, such as portlet layout and background template selection. This paper will address additional personalization in context of the application in the profile management section of Enterprise Security.

Authentication: all portal products provide this capability. The best practice is to externalize this service. Most enterprises have implemented global directory services (such as LDAP) within the enterprise. The custom application framework should provide an authentication interface and externalize the service.

Single sign-on (SSO): the enterprise should provide a seamless user experience by not requiring multiple logins. This framework component should not only support custom applications, is should also support packaged applications and enterprise services.

2.3.1.4 Enterprise Infrastructure Services

These are services, based on applications, that could potentially be leveraged by the entire external and internal user community. Most enterprise services are infrastructure components and some of them provide the capability for users to leverage them as an application. Following are some examples of enterprise services.

Directory service: this is the standard directory service provided enterprise wide, generally in conjunction with the e-mail service. Most enterprises implement a meta-directory for managing identity across the enterprise.

Personal information management: this is the standard e-mail, calendar, and address book functionality and includes access to this information from any channel.

Collaboration: this provides capabilities such as white board, conference calling, instant messaging, discussion forums, news groups, and workspace.

Enterprise content management system: this is the infrastructure service for driving custom applications such as knowledge management, asset or contract management, and collaboration. The recommended approach is to leverage the APIs provided by the portal or the content management provider. All the leading content management systems provide the capability to develop templates for uploading and authoring content as well as workflow for managing approval process.

Following are the best practices for implementing an enterprise content management system:

- Define the taxonomy up front, ideally creating one that is enterprise wide.
- Create a single document base or repository, enterprise wide. This may not be practical, but is a good goal.
- Publish all content to one single location in production and configure all applications to retrieve content from that location, for a reduced TCO.
- Train authors and content approvers in using the system.
- Partner closely with the content management system provider by engaging their architects for every project, especially during the design phase of the project.
- Leverage the pre-built portlets to author, review, and manage the content.
- Engage a specialized function person from either your SI or content management system (CMS) provider to map your business processes to CMS workflow.

Search service: this provides capabilities for any external or internal user to find the information they are authorized to access. There are several search solutions:

1. Key word search, which is the standard search capability that most users are accustomed to
2. Natural language search, which is generally targeted towards a non-technical or Internet savvy user who has just been introduced to technology and wants to find information by asking questions using their local language
3. Federated search, which enables search of structured and unstructured data types.

The integration of a search engine is straightforward. The search engine processes XML or HTTP requests and returns the results in the order requested.

The search engine goes hand in hand with the content management system. Following are some of the best practices for getting the most from a search engine.

- Create one taxonomy enterprise-wide for the content management system
- Define meta-tags for the content and leverage them in the portal to present content to the users based on their entitlements
- Use search engines to crawl and create multiple collections and sub-collections as required
- Leverage federated search between various business units, if required
- Leverage portal tags and entitlements to protect secure contents

- Store secure content at the application server level.

For large sites, the time taken to crawl the entire content repository can be an issue. Depending on business needs, a company might resolve this by creating multiple collections and including all the collections in the search criteria, performing partial crawls on a periodic basis, setting up multiple search engines, and leveraging federated search. It helps to develop the architecture and the process in collaboration with the search solution provider.

2.3.1.5 Enterprise (Role-Based) Portal

By implementing the web tier components defined in this document, enterprises would achieve the "current state" as illustrated in the diagram below.

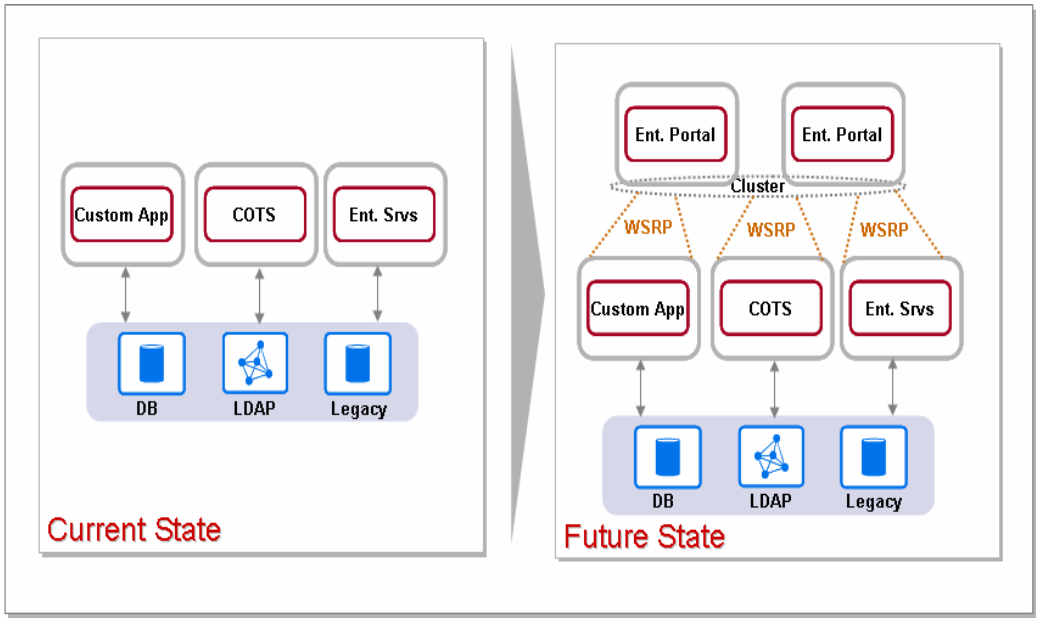

Figure 11: Enterprise (Role Based) Portal

In this state, IT organizations can rapidly deploy business solutions in the form of customer applications, packaged applications, enterprise services, or a combination of these components. The custom application framework enables business to provide an exceptionally good user experience. However, this has the following drawbacks:

- Re-branding the user experience would potentially require changes to all the sites.
- Users still need to know the URLs for each of the sites. By adopting some best practices, this can be reduced but not eliminated.
- This model is likely to result in redundant hardware and software for each of the point solutions. This is because each of the business units would like to schedule their own maintenance windows and the only way to facilitate this is to have dedicated infrastructure for each of the point solutions.

The target state is to leverage the concept of federated portals to create an enterprise-wide role-based portal. The advantages of this approach are as follows:

- Single point of entry for all employees, customers, partners, and other users
- Application (Portlets) access based on the role of the user
- Consolidation of hardware and software infrastructure
- Always-on capability
- Simpler re-branding of sites

- Multi-channel delivery provided by the federated portal by leveraging services.

2.3.2 Service Tier

The service tier is the primary enabler of the SOA and includes the components described in this section. It enables integration and business process automation across the enterprise. This tier is based on the SOA principles of coarse-grained, loosely coupled, and standards-based services. It helps IT respond to changing business needs by providing global solutions with reduced application and infrastructure complexity, increased reuse of business services, and service orchestration capabilities.

2.3.2.1 Service Bus

The service bus is the key component for delivering a service-oriented infrastructure for IT agility and alignment with business needs. It should have seamless integration with service registry and service management components to accelerate configuration and deployment management and simplify management of shared services across the enterprise.

The service bus should be able to receive any synchronous or asynchronous message in any protocol and route it to the destination based on configuration rules. In addition, it should provide the capability to transform the message to the format required by the destination. As this controls the message flow between the consumer and the producer, the service bus is in a unique position to manage, monitor, and enforce the service levels.

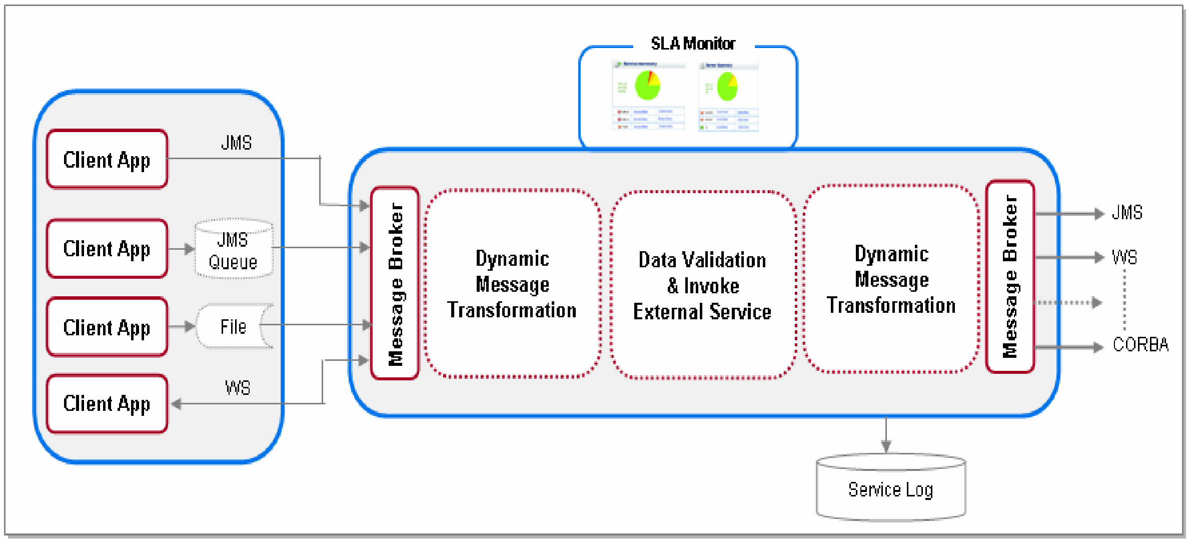

Figure 12: Enterprise Service Bus Architecture

The above diagram represents the enterprise service bus (ESB). The ESB acts as a dynamic and configurable message and service broker. It provides the following capabilities:

- Message brokering between heterogeneous environments
  - Supports asynchronous, synchronous, publish and subscribe messaging
  - Supports synchronous and asynchronous bridging
  - Supports multiple message formats including SOAP, SOAP with attachments, XML, structured non-XML data, raw data, text, and e-mail with attachment
- Heterogeneous transports between service end points
  - Supports multiple protocols such as file, FTP, HTTP(s), multiple JMS providers, RMI, web services, CORBA, DCOM, and e-mail (POP, SMTP, IMAP), SIP.
- Message transformation to enable the consumer to talk to the producer, but does not provide a fully fledged transformation engine
- Configuration-driven routing
  - Message routing based policies or call-outs to external services to support complex routing
  - Support for both point-to-point and one-to-many routing scenarios, enabling request-response and publish-subscribe models
- Monitoring
  - Service monitoring, logging, and auditing with search capabilities
  - Capture of key statistics for message and transport attributes, including message invocations, errors, performance, volume, and SLA violations
- High availability
  - Supports clusters and gathers statistics across the cluster to review SLA violations
  - Simplifies service provisioning
  - Deploys new versions of services dynamically through configuration
  - Migrates configured services and resources between design, staging and production
  - Supports multiple versions of message resources that are incrementally deployed with selective service access through flexible routing
- Configurable policy-driven security
  - Supports the latest security standards for authentication, encryption-decryption, and digital signatures
  - Supports SSL for HTTP and JMS transports
  - Supports multiple authentication models
- Policy-driven SLA enforcement
  - Establishes SLAs on a variety of attributes including throughput times, processing volumes, success/failure ratios of message processes, number of errors, security violations, and schema validation issues
  - Initiates automated alerts or enables operator-initiated responses to rule violations using flexible mechanisms including e-mail notifications, triggered JMS messages, triggered integration processes with a JMS message, web services invocations with a JMS message, or administration console alerts.

Following are some best practices for the service bus:

- Adopt the service bus whenever the number of services is more than 50. One definitely needs a service bus when the number of services exceeds 150.
- Start small by targeting a single composite applications or divisional business process that spans multiple systems.
- Consider having multiple LOBs manage their own service bus based on their policies, and a service bus at an enterprise level that could act as a broker for sharing services across the various business units.
- Decide between deploying a vendor-provided service bus and an internally developed abstraction layer.

2.3.2.2 Service Registry

SOA requires services to be coarse-grained, loosely coupled, and standards-based. As services are developed and deployed there must be a catalog of services available for architects, developers, operations, and business managers.

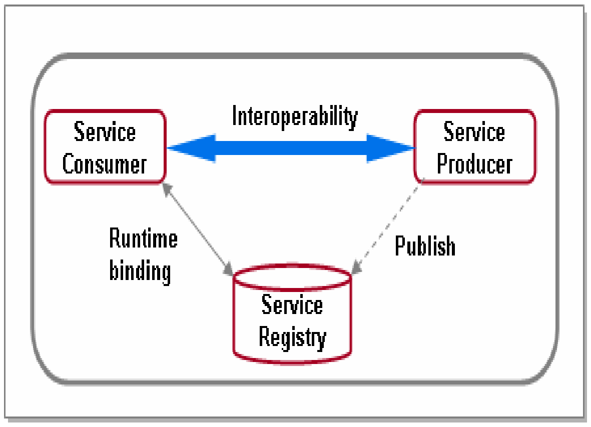

Figure 13: Service Registry

The above diagram illustrates the architecture of the service registry. The service producer publishes the service to the service registry which is leveraged by the service consumer for runtime binding. The registry also acts as the system of record for the business policies that it enforces at runtime.

The service registry should provide the following capabilities:

- Core services, including replication, UDDI data store, and security
- Information services, including data validation, SOA mappings, advanced classification, and business data access service
- Lifecycle services, including approval and change management, change notification, business service discovery, and QoS management
- Web-based business service console for configuration
- Platform-independent open architecture that interfaces with leading enablement, management and security products.

Best practices for the service registry include:

- Start small and grow over time
- Replicate every implementation of the service registry to the enterprise service registry
- Provide service browsing capabilities for architects, developers, and operations to facilitate re-use and identify service dependencies
- Maintain service contract information along with the service definition
- Version all services.

2.3.2.3 SOA Repository

An SOA repository is a key component for managing metadata throughout the lifecycle of a service, from inception through retirement. Its primary purpose is the storage of detailed metadata in order to manage and govern the assets prior to deployment. SOA repositories store service definitions, so teams can check what services have already been defined within the enterprise. SOA repositories also store other types of metadata, including process mappings, business rules, entities and relationships, orchestrations, transformations, reference data models, business activities and events, audit requirements, role and authorization mappings, and governance rules and policies.

The repository also helps reduce “service sprawl” by identifying and managing dependencies in order to maximize reuse. When the repository contains metadata from all an organization’s products, it provides architects and IT decision makers a valuable overall view of their services, systems, and data dependencies. To empower business, IT and operations in making the right decision, the SOA repository should store as valued assets standardized governance processes, shared business and technology best practices, and development guidelines.

SOA repositories help govern SOA assets during the design stages. They enable the sharing of metadata across different stages of the SOA lifecycle, and provide an optimal location for triggering approval workflows as assets are populated within the repository. Repositories can help ensure that assets go through the appropriate approvals as they move through the lifecycle, reducing ‘maverick development’ of services and encouraging higher rates of reuse.

SOA repositories also provide a central location for managing policies that are associated with services for runtime enforcement, such as routing, security, and SLAs. The dependency tracking and impact analysis capabilities of repositories help teams manage change to policies or other assets and proactively analyze the impact a change will have to other assets.

An SOA repository should provide the following:

- Ability to publish and discover metadata (service, business process, user interaction, etc.)
- Sophisticated metadata search by category, composite application name, service description, scope (such as division or department), or any other metadata type tracked within the repository
- Service dependency mapping, to track both which assets a service depends on and which assets depend on this service
- Notification of changes in metadata
- Extensible metadata taxonomies, so that businesses can customize taxonomies based on their own requirements
- Authorization procedures to control who can create and manipulate metadata contained within the repository
- Maintenance of version information for all assets
- Impact analysis for proactively measuring the impact of a change prior to making the change
- Open, extensible interfaces for synchronizing with development environments
- Synchronization with other external stores, such as a registry or other repository
- Design-time semantic validation of the metadata based on the design-time policies
- Approval workflows for promoting or rejecting metadata
- Federation and partitioning for multiple synchronized repositories.

Depending on the number of services and their interdependencies, IT organizations adopting SOA for more than three projects will probably need an SOA repository–especially if they have multiple distributed development teams and a lot of services. As organizations mature to the point where they are reusing services for development of composite applications, processes, or services, they will need a repository to enable management and governance for reusability.

2.3.2.4 Service Manager

As the SOA implementation matures in an enterprise, the need for an overall service manager increases. The primary function of this service manager is to manage, monitor, and report on all the services enterprise wide. Following are some of the capabilities that a service manager needs to provide:

- Manage and maintain the service level enterprise-wide
- Map and maintain service hierarchy across the enterprise and provide dependency matrix to operations
- Detect and manage exception conditions
- Review and monitor business transactions, and provide the capability to review in-flight transactions
- Manage service lifecycle and validate before deployment
- Provide non-intrusive service discovery across multiple systems
- Manage and integrate with multiple service bus and service registry infrastructures
- Leverage the existing monitoring infrastructure.

2.3.2.5 Shared Data Service

Shared data service provides several key capabilities:

- **Electronic data interchange (EDI),** the transfer of data between different companies using networks
- **Data manipulation**
  - Extract, transform & load (ETL)
  - Enterprise information integration (EII)
- **Data quality**
  - Data matching engine
  - Data stewardship
  - Workflow.

EDI and ETL are traditional approaches, especially useful for handling large volumes of data in batch mode. However, one of the SOA requirements is the ability to invoke some EDI/ETL capabilities as services. For example, an electronic fund transfer must populate an operations data store from the source systems triggered by an event.

2.3.2.5.1 Enterprise Information Integration (EII)

EII refers to software systems that can take data in different formats from a variety of internal and external sources and treat them as a single data source.

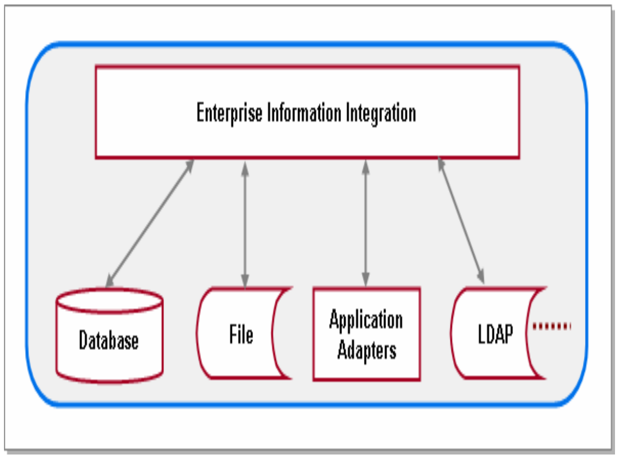

Figure 14: Enterprise Information Integration (EII)

These capabilities should be provided by EII:

- Data modeling across multiple sources
- Query (read and write) development to extract information from multiple data source.
- Support for multiple data sources such as database, file, application adapter, LDAP, and web services
- Data transformation
- Data validation
- Exposure of data services to client applications using RMI or web services.
- Adherence to standards such as SQL, XQuery, XML, web services, JDBC, and J2EE.

Even though the service data object (SDO) standards have been defined to simplify and unify the way in which applications handle data, the industry has not yet clearly defined the standards for EII. Vendors all have their own extensions that deal with reading, updating, and inserting data to each of the data stores.

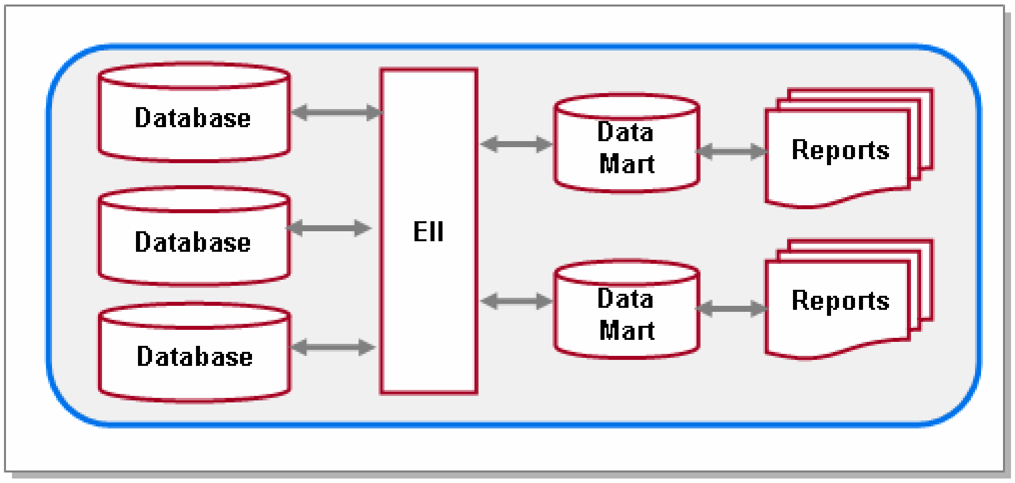

Figure 15: Leveraging EII for Data Marts

The best practice is to use business intelligence (BI) tools to provide analytic reports from data warehouses, ODSs, or data marts. ETL is batch oriented, so the business typically gets a delayed report, which may not always represent the current state of the enterprise. To deliver real-time information, IT organizations are starting to leverage EII tools to extract the data in real-time from the source systems directly into the BI tools. All major BI vendors support this approach, which is also driving the convergence between the ETL and EII tools.

This convergence does have an impact on an organization’s culture, especially as most data architects, data analysts, and DBAs are more familiar with ETL tools than with EII tools. IT organizations should plan on training staff and bringing in EII experts before undertaking such projects.

2.3.2.5.2 Data Quality

Enterprises need to focus on data quality because:

- The quality of data directly affects the quality of information
- Poor data quality causes inefficiencies in the business processes which depend on data
- Critical decisions based on poor-quality data can have very serious consequences
- Poor data quality can reflect adversely on the organization, lowering customer confidence
- If the data is wrong, the company could lose time, money and its reputation.

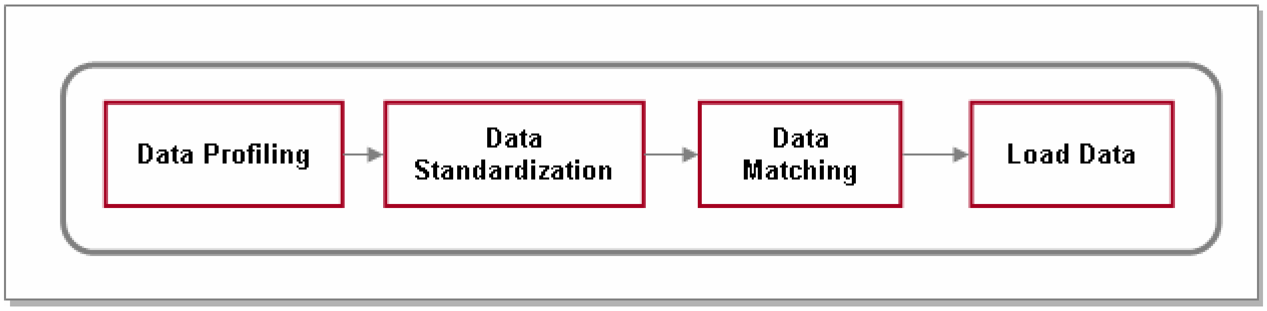

Figure 16: Process of Improving Data Quality

To improve data quality, organizations need to focus on:

- **Data profiling:** the first step towards creating a high-quality data environment is to understand the level of data quality in the current environment. Measuring the success of a data quality improvement initiative depends on correctly assessing the state of data quality at the outset.
- **Data standardization**: organizations must define and apply the business rules for data standardizations.
- **Data matching & load data:** IT must match standardized data with existing cleansed data and either create or update existing data. In addition, IT must expose shared services in order for data matching to be leveraged by client applications.

In addition to data quality infrastructure, enterprises also need a master data management (MDM) capability. With so many different applications at work, organizations may find they have multiple representations of entities within and outside the enterprise. MDM consolidates and rationalizes representations of key entities such as customers and products to ensure accurate, consistent information.

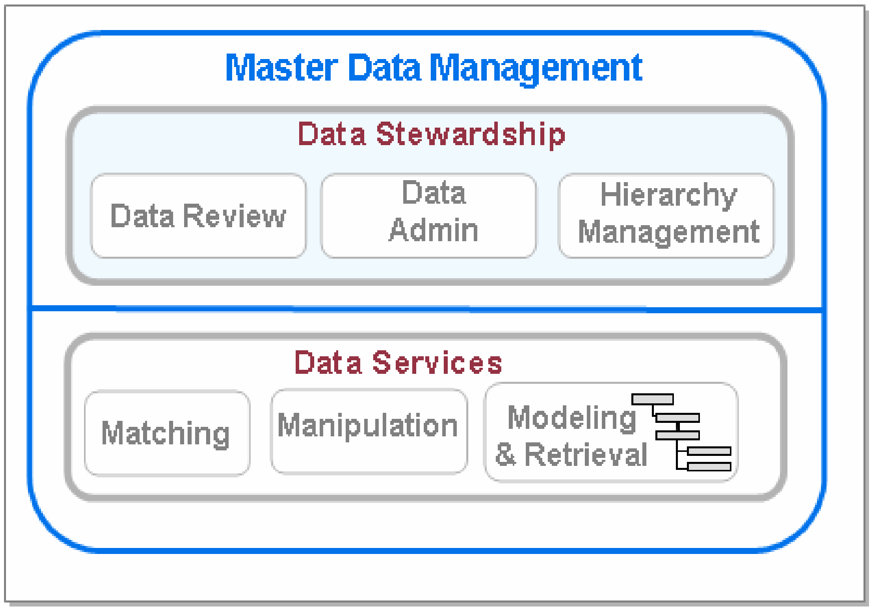

Figure 17: Master Data Management

MDM consists of data services and data stewardship.

- **Data stewardship** is an application that enables users to review exception data, administer data, and create multiple hierarchies for reporting purposes.
  - **Data review** provides the capability to review linked and unlinked data and take corrective action
  - **Data administration** enables staff to modify or manually override a match, modify matching and data survivorship rules, and manage users. In addition, it enables teams to leverage data enrichment rules from third-party sources such as Dun & Bradstreet.
  - **Hierarchy management** enables staff to create master data hierarchies based on industry, geography, revenue, and organization.
- **Data services** is a set of configurable rules that provides the infrastructure for MDM. Its capabilities include:
  - **Matching** using fuzzy logic to match and standardize master data based on the business' predefined matching rules.
  - **Manipulation** for data movement and tweaking. In addition, it provides basic workflow for matching, cleansing, and standardizing data as well as enforcing data survivorship rules
  - **Modeling & retrieval** enables the business to modify the MDM with limited or no IT involvement. Traditionally, ETL or custom tools provided retrieval functionality, but now companies often use EII tools for this.

2.3.2.5.3 Best Practices

Following are some of the best practices for shared data services.

2.3.2.5.3.1 Implement Master Data Management

Companies need to focus on overall business processes to achieve the full benefit of SOA. As data is the source of truth for the enterprise, it is very important to map the enterprise data flow as it relates to business processes. The following diagram is an illustration of the importance of mapping key data elements to the business processes.

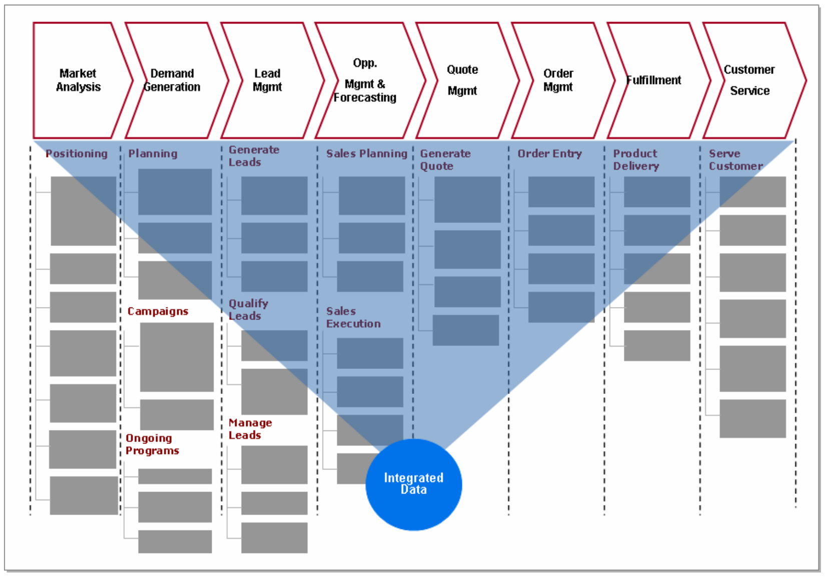

Figure 18: Integrated Data

It is not easy to develop the integrated data model before adopting SOA. The recommended approach is to phase the data model in gradually as you implement SOA, one business processes at a time. Business needs to prioritize the business processes while IT works in parallel to develop an enterprise data model that maps the data flow to the business processes.

Companies quickly realize that while there are many processes and sub-processes, there is a limited set of enterprise data objects—such as customers, contacts, and products–that are required across most of them. Data model proliferation is a threat that organizations need to address by implementing MDM solutions.

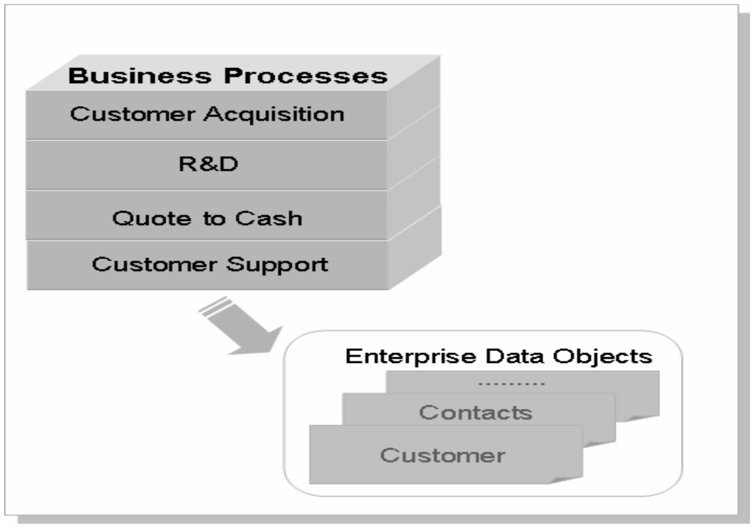

Figure 19: Identifying Enterprise Data Objects

Organizations implementing MDM have learned that:

- There are no shortcuts. Mapping the business process at a high level helps identify the enterprise common objects or entities.
    - Business defines the process
- Resolving MDM (the common objects) is the highest priority.
    - Business defines the enterprise's data needs
- The ODS and data warehouse can be derived using the common objects / models.
    - Teams develop appropriate infrastructure as required
- Business project teams must develop the capability to populate the ODS/data warehouse or MDM solutions.

2.3.2.5.4 Convergence of Structured and Unstructured Data

"Structured data" refers to enterprise data stored in databases, while "unstructured data" is enterprise data stored in different documents.Today most enterprises have a solution for searching unstructured data and can leverage applications to access structured data. However, the users would prefer to review both structured and unstructured data on the same page.

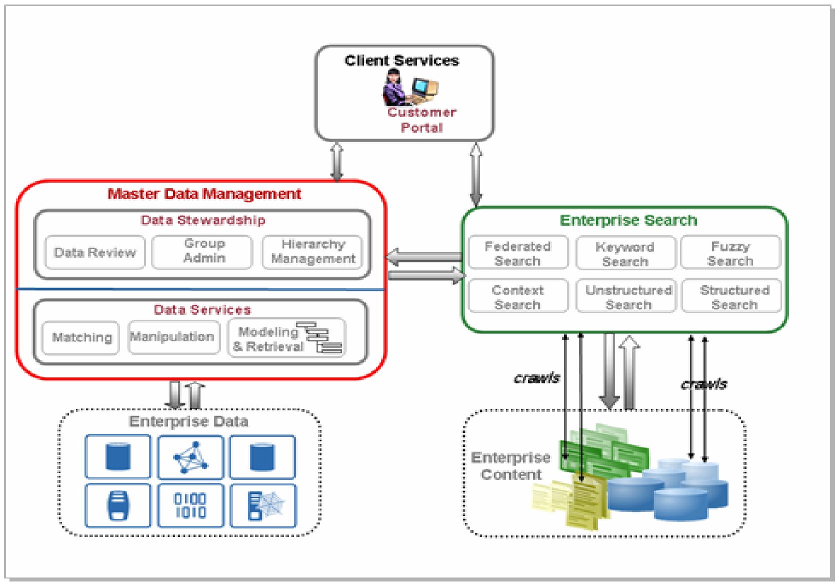

Figure 20: Convergence of Unstructured and Structured Search

Convergence requires three components:

- Portal for presenting information to the user
  - Federated portal leverages WSRP to present structured and unstructured information
- Search engine
  - Delivers unstructured content to the portal
- Shared data services
  - Access structured data, preferably using EII
  - Search for enterprise data objects leveraging MDM.

Each enterprise search component plays a different role:

- **Federated search:** submits search request to external service providers
- **Keyword search:** searches based on key words
- **Fuzzy search:** searches by natural language or pattern
- **Context search:** maintains context of the search without having to include operators
- **Unstructured search:** searches different types of content including 200+ document types
- **Structured search:** searches databases

2.3.2.6 Business Process Management

BPM is used to manage long-running synchronous and asynchronous business processes. While the service bus performs lightweight service orchestration, the following functions and features should be provided by the BPM:

- Visual process modeling to modify views and business process models
- Open standards compliance, preferably BPEL
- Business process orchestration and automation between private processes, public processes, human tasks, and error handling
- Support for nested and concurrent processes for advanced modeling and custom logic as required to enable rapid customization
- Optimized process performance to provide flexibility of configuration for state-full (long-running) and stateless (short-running) process design patterns as well as synchronous and asynchronous process execution
- Status monitoring to show users end-to-end processes graphically and measure performance against service level agreements
- Process instance monitoring to provide statistics on running processes, drill into individual details, and terminate, delete, or suspend problematic process instances
- Enable task creators, workers, and administrators to interact with running business processes for handling process exceptions, approvals, and status tracking
- User and group management to centralize the assigning of roles, users, and groups working on integration projects
- B2B protocol support for rapid, secure online connection with suppliers and customers using leading standard protocols such as RosettaNet, ebXML, and EDI, with secure messaging, digital signatures and encryption, recoverable and traceable messages, and dynamic configuration updating.

2.3.3 SOA Frameworks

SOA frameworks are reusable services that are the backbone of the SOA. These re-usable services must be enterprise-class, designed well enough to scale under load and meet the demands of a diverse set of consuming applications and stakeholders.

SOA frameworks support the move to an SOA by helping development teams rapidly design, develop and deploy well-designed, modular, flexible, scalable, and supportable web services, web applications, and portlets. As companies start adopting SOA principles to transform their IT architectures, these underlying services must be created to perform as enterprise-class assets.

A framework can be defined as a reusable skeleton application that teams extend in order to build specific services or applications. Frameworks improve consistency in the delivered software. Frameworks control themselves and the application or services that are created on top of them.

Frameworks typically provide a set of high-level programming abstractions and a strong starting point for creating enterprise-class services. They often specify a layered architecture for services that incorporates several design patterns and software engineering best practices. The architecture also specifies the responsibilities of the components in each of the layers and the collaboration between them.

Services inherit the good architecture and best practices built into the frameworks. Using frameworks, a team of average developers can develop well architected services that take advantage of design patterns and best practices. The typical layers that a services creation framework would offer include:

- **Transformation layer:** supports protocol and data-type conversions to support multiple access protocols, while at the same time keeping most, if not all, of the service implementation protocol and access mechanism agnostic.

- **Business logic layer:** holds all the business logic in the system. This includes such abstractions as request, result, UseCaseController, and BusinessPolicy objects.

- **Business data layer:** the layer for domain objects, which are the objects that have a consistent definition across many applications in the enterprise. The business data layer should provide location transparency so that users of the domain objects don’t need to know the exact physical location of the underlying persistent data. This layer should be able to manage persistence to and retrieval from various persistence repositories in the enterprise.

- **Integration layer:** a placeholder for connection technology implementations ranging from JDBC to JNI to Java connectors. All the infrastructure code that is needed to access extended enterprise systems—such as ERP systems and content repositories—fits into this layer.

SOA frameworks benefit both developers and the corporation. When developers use frameworks, they:

- Gain a solid foundation to create services, web applications and portlets
- Improve productivity by incorporating design patterns and best practices
- Utilize off-the-shelf features of the frameworks and write less code
- Don't need to understand the nuts and bolts of J2EE standards and specifications
- Don't need to be an expert at object-oriented design and design patterns to benefit from using them.

For IT organizations and the company as a whole, the SOA frameworks deliver:

- A catalyst for getting to a SOA quickly and at a lower cost
- Consistency of design and development across projects
- Repeatability and the ability to guarantee a minimal level of architecture and design rigor.
- Improved business agility as a result of having modular solutions that can be changed easily, often via configuration changes
- Leverage of software engineering best practices among developers with varying skill levels
- More consistent, predictable, and better tested solutions
- Improved mobility of developers among projects.

IT organizations are using SOA principles to aggressively create re-usable services that encapsulate and expose key business processes. By combining a layered architecture, ease of use, and a deep emphasis on good architecture and re-use, SOA frameworks companies to create enterprise-class mission-critical services that are vendor-neutral and portable.

2.3.4 Application Tier

IT organizations' greatest investments are in applications. Even though enterprises will invest in SOA moving forward, the application tier is not going away anytime soon.

2.3.4.1 Legacy Applications

These are the thick-client applications. They may be old versions of packaged applications that have not yet been upgraded.These applications may be best suited to run in the client/server mode.

These applications will typically have some published APIs or logical models for integration.

2.3.4.2 Mainframe Applications

Business solutions that run on proprietary mainframe systems may be integrated with other systems either through a messaging or database gateway. Mainframe software vendors are working to expose all their APIs as web services. However, in most cases, the best approach is to support the business requirements by leveraging middleware to develop an abstraction layer and exposing the services at the appropriate level.

2.3.4.3 Enterprise Application Integration

Enterprise application integration (EAI) typically uses middleware to provide the following capabilities:

- **Messaging:** message-oriented middleware (MOM) to enable resources to publish and subscribe to messages
- **Business process manager (BPM):** proprietary capability to automate business processes
- **Application adaptors:** pre-built connectors to various packaged applications that allow access to the application views or technology adaptors of other technologies such as databases, messaging (MQ, JMS), and web services.

The best practice for EAI is to leverage an integration hub. However, without the appropriate supporting methodology, this may result in a point-to-point solution on the hub. Even though the services tier provides the service bus and BPM, which enables enterprises to adopt SOA and migrate away from EAI, this migration is expected to take a long time, especially when IT organizations have invested heavily in EAI.

2.3.5 Enterprise Security

Currently most packaged or custom applications implement their own security solutions. However, when an enterprise has multiple applications, each with its own authentication implementation, problems can arise when different authentication implementations are updated at different times.

This could be simplified by dividing enterprise security into three major components:

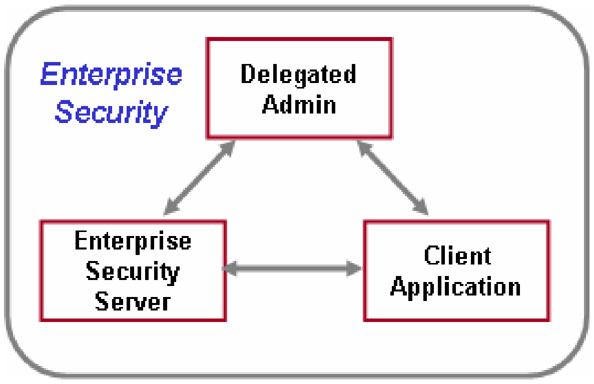

Figure 21: Enterprise Security Components

Delegated administration: user and resource administration application that enables administrators to create, modify, and delete user privileges as appropriate to their role. This application updates the same repository leveraged by the enterprise security server.

Client applications: externalize and leverage the enterprise security services.

Enterprise security server: provides security services such as user authentication, user identity management, authorization, auditing, user profile management, and user provisioning.

- Authentication validates the identity of a user through several authentication mechanisms. Identity can be validated against a password, or through digital certificates or smart cards. In the interests of productivity, users should need to be able to sign on once and gain access to multiple resources, but enterprises need to implement proper controls and policies so that, once authenticated, users can access only the resources they're authorized to work with.
- Identity management maps multiple identities to a single user or links identities across applications to support coexistence of multiple user identities.
- Authorization involves controlling access of users to diverse resources across the enterprise.
- Auditing involves tracking user activities.
- Profile management involves managing the profiles of the users.
- Provisioning automates the time-consuming process of managing accounts and their life cycles. It allows centralized activation, modification, or deactivation of accounts across multiple applications in an enterprise. User provisioning could include explicit granting or revoking of user access to resources and establishing of entitlement policies for the user.

2.3.6 Business Service Management

The primary focus of business service management is monitoring and reporting on operations of the network, infrastructure, application, or business process.

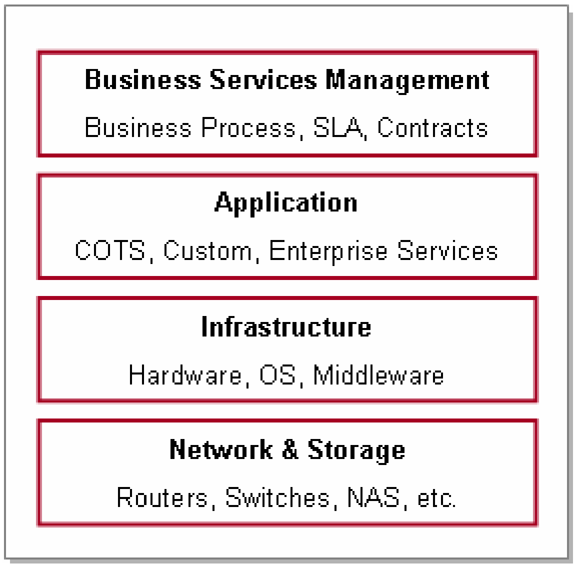
Business Services Management
Business Process, SLA, Contracts
Application
COTS, Custom, Enterprise Services
Infrastructure
Hardware, OS, Middleware
Network & Storage
Routers, Switches, NAS, etc.

Figure 22: Business Service Management

Network and storage: network elements such as routers, switches, PABX, load balancers, etc. Typically operations has a dedicated team to architect, design, and monitor the network with most of the long-haul networks purchased from telecom network providers.

Infrastructure: all the hardware, operating systems, and middleware such as applications, servers, databases, and LDAP. IT operations is responsible for monitoring and providing tier 1 support for the infrastructure layer, sometimes also referred to as data center monitoring.

Application: monitoring of application and enterprise services such as e-mail, search, and content management systems. IT operations provides tier 1 support, while a dedicated team provides application support.

Business service management: monitoring the business processes, policies, activities, and IT service level agreements.

2.3.6.1 Typical Current State of Monitoring

Most enterprises do not have comprehensive monitoring capabilities, resulting in:

- Multiple monitoring systems that overlap in functionality
- Initiative not staffed at the required level, resulting in software still on the self
- Limited or no mechanism to correlate events across business or product silos
- Lack of a unified view of IT configuration to facilitate correlation
- Limited or no view of service model that maps configuration elements to business processes for SLA management
- Limited mechanism for tracking historical availability and performance of services
- Limited integration of ticketing back to correlation to prevent duplicate notification.

Following is an exhaustive list of the capabilities required to support BSM. IT operations do need to review their current state and develop a roadmap for deploying these capabilities. Unfortunately, the biggest challenge for IT organizations is to get funding approval, especially as business tends to prioritize business capability over monitoring and management capabilities.

- **Ticketing:** tracking of problems that have been identified and escalated from incidents
- **Incident:** any alert—not necessarily a problem—received by the monitoring system or the service desk
- **Notification:** forwarding of tickets to the service desk or responsible work group for action; this would include previous requirements identified for paging
- **Diagnostics and root cause analysis:** tools to diagnose the problem in order to identify the root cause; the diagnostic process typically includes a database of solutions or workarounds for known problems
- **Event console:** provides view of the availability and performance management data stored in the monitoring data store
- **Correlation engine:** software tasked with interpreting and manipulating incoming data from different sources
- **Decision engine:** based on correlation result, this software makes a decision about what actions need to be taken, such as logging a ticket, alerting a responsible party, running a script, or doing nothing
- **Monitoring data store:** centralized database containing event data and configuration information, also referred to as CMDB
- **Service-level objectives:** central repository of availability and performance targets associated with applications or business processes
- **Service-level reporting:** the ability to measure actual performance and availability against service level objectives
- **Automation engine:** allows actions—such as log a ticket, run a script, or initiate another recovery process—to take place automatically for known problems
- **Target methods:** process and workflow followed to manage known problems.
- **Automatic scripts:** pre-defined actions to test for or react to known problems
- **Capacity planning:** current and forecasted utilization for configuration items
- **System modeling:** representation of the configuration items that comprise a system
- **Workload simulation:** performance and capacity testing using an automated tool for generation of synthetic transactions against a test infrastructure.
- **Probe:** software that gathers fault or performance information about a device, but is not installed on the device
- **System monitor:** software installed on machines and network devices that gathers availability information
- **End-user monitor:** software to measure the experience of an end user via synthetic transactions or monitoring actual user activity
- **Business availability console:** provides a business perspective of how availability management data is impacting the business

2.3.6.2 Future State of Business Service Management

SOA enables businesses to monitor and manage their business processes through business service management. Businesses must define and capture business process policies during the requirements phase and carry them through to implementation. To support this, IT organizations need to invest in additional software and secure additional funding, both for infrastructure and head count.

2.3.7 Service Oriented Infrastructure (SOI)

As the infrastructure to support SOA, service-oriented infrastructure (SOI) requires the resources to compute, network, and store data: networks, routers, network appliances, servers, storage, data centers, VoIP, and wireless capabilities.

Currently, storage products have consolidated manageability models and operation procedures through the SNIA SMI-S standard. SOI will leverage that standard as part of SOI architecture. In contrast, SOI network integration relies primarily on proprietary protocols. There is no industry-wide interoperable management standard available yet, except the outdated simple network management protocol (SNMP).

For computing resources, there are two work-in-progress standards developed by DMTF. The server manageability standard defined by DMTF Server Management Working Group (SMWG) focuses at the individual system level to define the server resource modeling before pre-boot. The Utility Computing Working Group (UCWG) defines the standard that deals with modeling management of the virtualized server pool. Architects can leverage these work-in-process standards as a foundation and propose additional extensions to take advantage of technologies for better performance and further capabilities.

There are two major components for SOI computing resources. The first is the infrastructure manager, software to aggregate computing resources and abstract them into a virtualized computing pool. The second is the computing resource itself, which combines software and hardware modules that enable the autonomic capabilities for compute resource virtualization.

Once the infrastructure manager aggregates and abstracts the computing resources, IT can choose its partitioning, such as a traditional one-server-one-OS partition, multiple OS partitions inside a compute system, or an OS partition span across multiple systems. Software features within the infrastructure manager enable IT to implement SOI services:

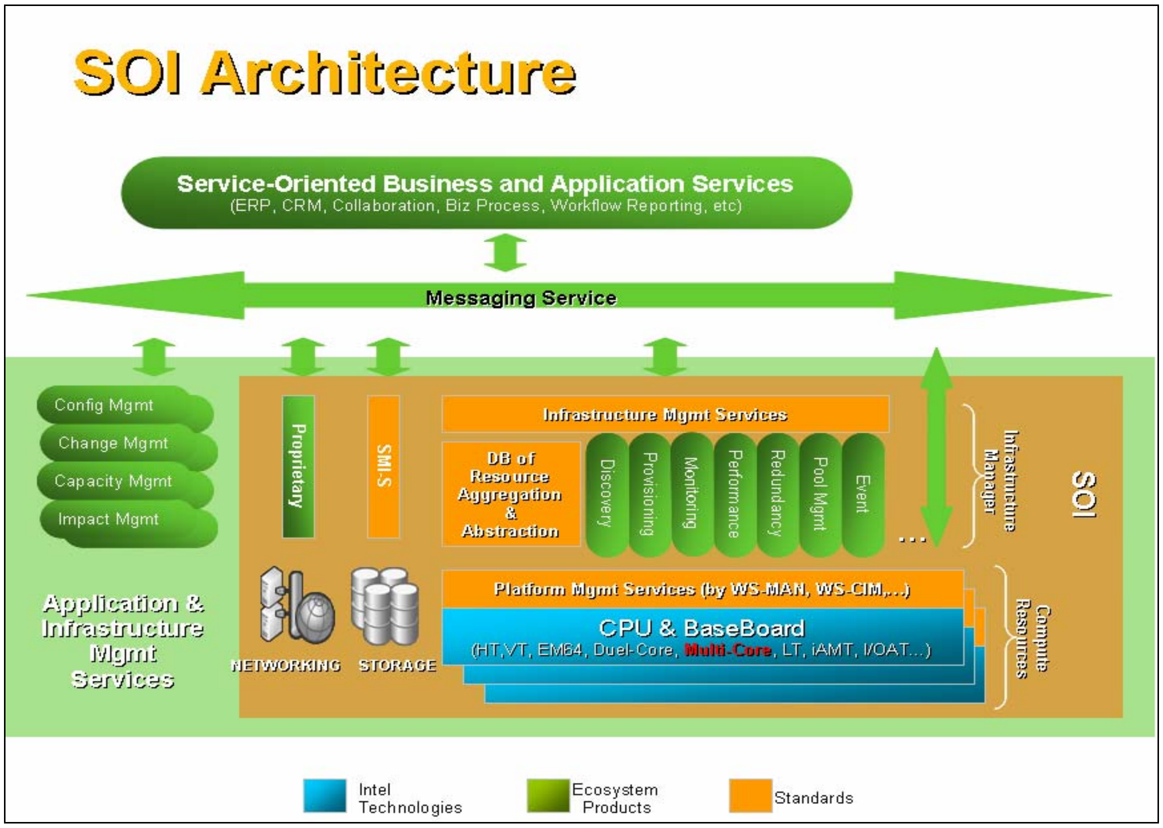
SOI Architecture
Service-Oriented Business and Application Services
(ERP, CRM, Collaboration, Biz Process, Workflow Reporting, etc)
Messaging Service
Config Mgmt
Change Mgmt
Capacity Mgmt
Impact Mgmt
Application & Infrastructure Mgmt Services
Proprietary
SMI-S
Infrastructure Mgmt Services
DB of Resource Aggregation & Abstraction
Discovery
Provisioning
Monitoring
Performance
Redundancy
Pool Mgmt
Event
Infrastructure Manager
SOI
NETWORKING
STORAGE
Platform Mgmt Services (by WS-MAN, WS-CIM,...)
CPU & BaseBoard
(HT,VT, EM64, Duel-Core, Multi-Core, LT, iAMT, I/OAT...)
Compute Resources
Intel Technologies
Ecosystem Products
Standards

Figure 23: Service-Oriented Infrastructure Architecture

- **Database of resource aggregation and abstraction**: this repository is the information foundation of an SOI architecture. It stores computing, network, and storage resources once they've been abstracted into logical models based on the common information model (CIM) of DMTF. Depending on the implementation, the repository might offer only simple resource configuration or it might include interdependencies among compute, storage, and network resources and their operation processes.

  Once IT has defined the common abstractions of the resources and their interdependencies, this repository can share and synchronize this information among internal and external applications to ensure the interoperability and consistency of resource management. The database also includes operation processes, the generic IT micro-operations that serve as building blocks to standardize IT processes. For example, server provisioning can be simplified by a generic operation that orchestrates configuration procedures across server OS installation, network switch VLAN settings, and SAN storage partitions. Currently, the DMTF Utility Computing Working Group is working on these generic micro-operation processes for server virtualization. As the scope expands, the UCWG or other standard consortiums will standardize more resource operation processes.

  IT staff can leverage these building blocks to develop more complex and custom processes for their own environment. Product vendors can extend operation processes to make better use of their product features. Vendors could differentiate themselves by extending the operation process for "bare metal" provisioning based on active management technology (AMT).

- **Infrastructure management modules**: these software modules provide the execution of services. The services can include automatic resource discovery, provisioning of an OS virtual partition, resource health monitoring and its event notification, and resource pool management. An SOI SDK can help accelerate the development of these management modules by providing software examples and libraries that show developers how to interact with the abstraction and aggregation resource database and how to register or advertise WSDL services to the external applications.

- **Infrastructure management protocol**: this set of XML protocols enables the upper-layer business or system management applications, either web services-enabled or legacy, to interact with underlying resources. SOI management modules encapsulate the complexity, keeping it from the business or system management applications, and maintain a runtime detail internally. This management protocol specification defines the details of the resource object handlers that manage the interaction.

  For example, a J2EE application server may reach its performance threshold and need more computing resource to comply with its SLA. Then, it can send an XML resource request based on the needed jAppServer SPEC benchmark to the SOI to ask for additional computing resources. The request may specify the application configuration and the operation process to join the J2EE application server cluster. However, there is no need to specify the detail system and OS configuration in this XML request because SOI will automatically validate the available J2EE application server platforms and reply with available options in the SOI. This can be based on either a dedicated Xeon server or a shared Itanium 2 4-way server using a specified jAppServer SPEC benchmark and available OS and availability. The J2EE application can select one of the options depending on availability or cost of resources. SOI will maintain the configuration context across computing, network, and storage resources, and encapsulate the detail from the business applications. This is just a starting point for resource performance metrics. It still needs many improvements and extensions with the current SPEC benchmark metrics in order to fulfill the final goal of SOI.

Beyond the infrastructure manager mentioned above, several key hardware and software features on the computing resource itself enable implementation of SOI services.

- **Asset identifiers**: provide a dependable way to remotely and uniquely identify and catalog physical resources attached to a network. This enables an enterprise management console to discover and account for all IT assets connected to the corporate network, including wireless devices.
- **Out-of-band (OOB) accessibility**: allows network access to platform management functions even if the platform is otherwise non-functional because it has not yet had operating system or application software loaded, or due to a software bug, system failure, malware attack or other problem. This does not require another network cable, but provides alternate paths and intelligence within the platform itself. This feature enables remote “bare metal” provisioning, reloading software that has become corrupted, running remote diagnostics, remote reconfiguration, and many other functions.
- **Service processor**: like OOB facilities, a dedicated management service processor supports functionality even when the primary platform processor(s) are not yet provisioned with software or are not functioning properly. A separate service processor also enables more effective monitoring, failure detection, and diagnostics in any platform system state. It also allows the platform to perform autonomous functions such as periodic self-test and diagnostics, automatic hardware and software configuration discovery and reporting, and self-repair actions, such as isolating itself from the network and sending an alert when it detects a virus.

To deliver interoperability among all the hardware and software components that make up the service-oriented enterprise, the SOI must be based on open, industry-wide standards that define common functions, data structures, and interfaces.

2.3.7.1 The Business Value of Services Oriented Infrastructure (SOI)

SOI delivers bottom-line benefits to the enterprise. It provides the basis for greater IT automation which results in higher IT productivity and lower operational costs. SOI also makes it possible to move from the static, “one-box-per-application” approach to dynamic resource allocation in which virtual processing, storage and network resources are allocated to applications as needed. This results in reduced capital costs through better resource utilization. It also yields higher reliability, with fewer service outages and delays, since applications can fail over to available resources without disruption of the application. IT can deliver more consistent service levels since software can automatically allocate additional resources in real time as an application’s workload increases.

Even more important in the long run is SOI’s contribution to overcoming the “complexity barrier” that limits organizations’ ability to design and implement new information-supported business processes. With an SOI in place, a business can reallocate IT overhead dollars to investment in new innovative systems that create true business differentiation. For example, SOI can help enterprises take full advantage of autonomic data sources such as RFID or eForms in real-time whent the data becomes available. SOI enhances the ability to implement and manage event-driven, message-based services and applications inside and across the corporate firewall. Better management of mobile wireless devices allows the business to push services to the edge of the network to make employees more productive and to deliver value-added services to mobile customers. SOI also enables greater business accountability, for example, improving Sarbanes-Oxley compliance in the area of IT asset inventories for accurate depreciation schedules, and providing timely software version and license management. These are high-impact capabilities—one high-tech company saved 40,000 person-hours in just five quarters by using automated IT asset inventory techniques.

From the unique perspective of IT management, SOI offers many benefits:

- Reduced operations workload due to reduced manual reconfiguration of hardware and software.
- High productivity resulting from the ability to perform enterprise-wide platform, network, data, and applications management from a single, standardized management console.
- Better resource utilization leading to reduced capital expenditures to accomplish IT objectives.
- Greater flexibility through dynamic resource allocation, as well as the option to focus on core competencies and move commodity IT services, such as corporate e-mail, outside the firewall to 3rd party providers. The ability of SOI to operate across firewalls means that IT can manage services consistently whether they are sourced within the corporation, or outsourced to external providers through telco or ISP connectivity.
- Simple and cost-effective upgrades thanks to a modular, loosely-coupled SOI architecture. This allows IT to refresh its infrastructure to take advantage of new technologies without complications resulting from hard-wired management dependencies. This eliminates painful “rip-and-replace” (aka “forklift”) upgrades, allowing IT to respond to change in a rapid and graceful manner, and reducing the inertia that causes IT to resist incorporation of new technologies because of expected disruptions
- Support for on-demand usage models which allow more accurate alignment of costs to benefits through pay-for-use chargeback methods.
- Support for managing all client platforms ranging from desktops to mobile hand-held devices, as an integral part of the infrastructure with common platform features and access interfaces.

Developers also benefit directly from evolution to a service-oriented infrastructure:

- SOI removes resource management and security responsibilities from the application code. Infrastructure management protocols ensure these functions are architected into web service-enabled hardware, SOI middleware, and applications building blocks from the bottom up and the inside out in a standardized, consistent manner. Rather than having to make critical assumptions that will invariably change, or embed logic to take care of this in the application code, developers need only specify virtual application requirements. They can then rely on the SOI to ensure that these are met dynamically.

- Developers can use pre-built services for distributed, real-time, event-driven, multi-platform applications because these elements are already architected to be consistent with, and take advantage of, a standardized SOI. Because developers can easily reuse components they have built for other services and applications, they are able to build and modify applications more quickly, and therefore can deliver more innovation and business value.
- Developers work more efficiently in heterogeneous environments, including cross-enterprise Internet/Web applications, because they're using building blocks that are web-based, service-oriented, and virtualized, meaning they conform to industry-standard capabilities and interfaces.

2.3.8 Mapping SOA Reference Architecture to Enterprise SOA Maturity Model

The following diagram represents common adoption SOA pattern. Each IT organization should introduce technology components in the manner that fits its particular needs.

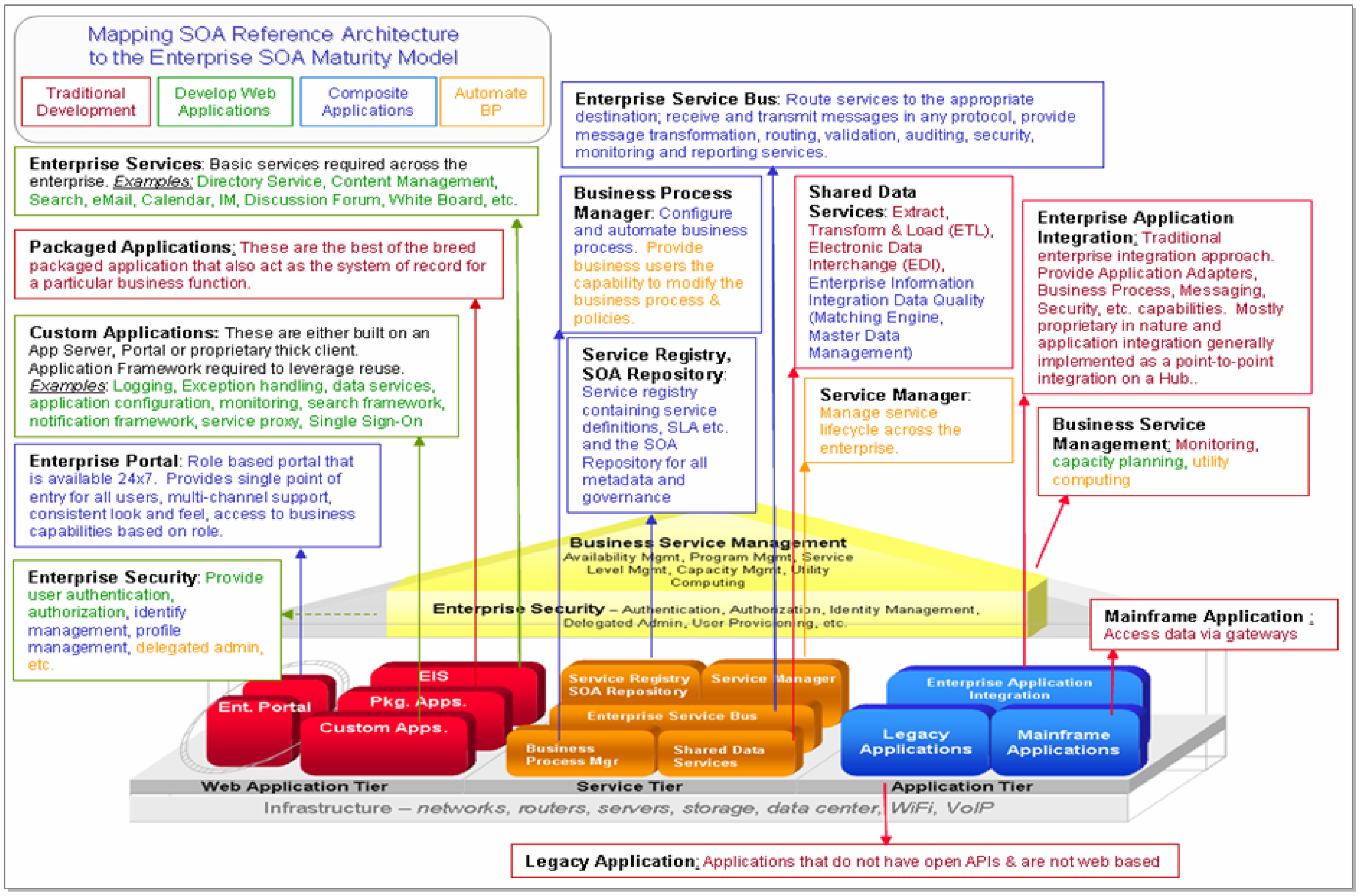
Mapping SOA Reference Architecture to the Enterprise SOA Maturity Model
Traditional Development
Develop Web Applications
Composite Applications
Automate BP
Enterprise Services: Basic services required across the enterprise. Examples: Directory Service, Content Management, Search, eMail, Calendar, IM, Discussion Forum, White Board, etc.
Packaged Applications: These are the best of the breed packaged application that also act as the system of record for a particular business function.
Custom Applications: These are either built on an App Server, Portal or proprietary thick client. Application Framework required to leverage reuse. Examples: Logging, Exception handling, data services, application configuration, monitoring, search framework, notification framework, service proxy, Single Sign-On
Enterprise Portal: Role based portal that is available 24x7. Provides single point of entry for all users, multi-channel support, consistent look and feel, access to business capabilities based on role.
Enterprise Security: Provide user authentication, authorization, identify management, profile management, delegated admin, etc.
Enterprise Service Bus: Route services to the appropriate destination; receive and transmit messages in any protocol, provide message transformation, routing, validation, auditing, security, monitoring and reporting services.
Business Process Manager: Configure and automate business process. Provide business users the capability to modify the business process & policies.
Shared Data Services: Extract, Transform & Load (ETL), Electronic Data Interchange (EDI), Enterprise Information Integration Data Quality (Matching Engine, Master Data Management)
Enterprise Application Integration: Traditional enterprise integration approach. Provide Application Adapters, Business Process, Messaging, Security, etc. capabilities. Mostly proprietary in nature and application integration generally implemented as a point-to-point integration on a Hub..
Service Registry, SOA Repository: Service registry containing service definitions, SLA etc. and the SOA Repository for all metadata and governance
Service Manager: Manage service lifecycle across the enterprise.
Business Service Management: Monitoring, capacity planning, utility computing
Business Service Management
Availability Mgmt, Program Mgmt, Service Level Mgmt, Capacity Mgmt, Utility Computing
Enterprise Security – Authentication, Authorization, Identity Management, Delegated Admin, User Provisioning, etc.
Mainframe Application: Access data via gateways
EIS
Ent. Portal
Pkg. Apps.
Custom Apps.
Service Registry SOA Repository
Service Manager
Enterprise Service Bus
Business Process Mgr
Shared Data Services
Enterprise Application Integration
Legacy Applications
Mainframe Applications
Web Application Tier
Service Tier
Application Tier
Infrastructure – networks, routers, servers, storage, data center, WiFi, VoIP
Legacy Application: Applications that do not have open APIs & are not web based

Figure 24: Mapping SOA Reference Architecture to Enterprise SOA Maturity Model