



SOA REFERENCE ARCHITECTURE: SERVICE TIER

SOA Blueprint

A structured blog by Yogish Pai

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.

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.

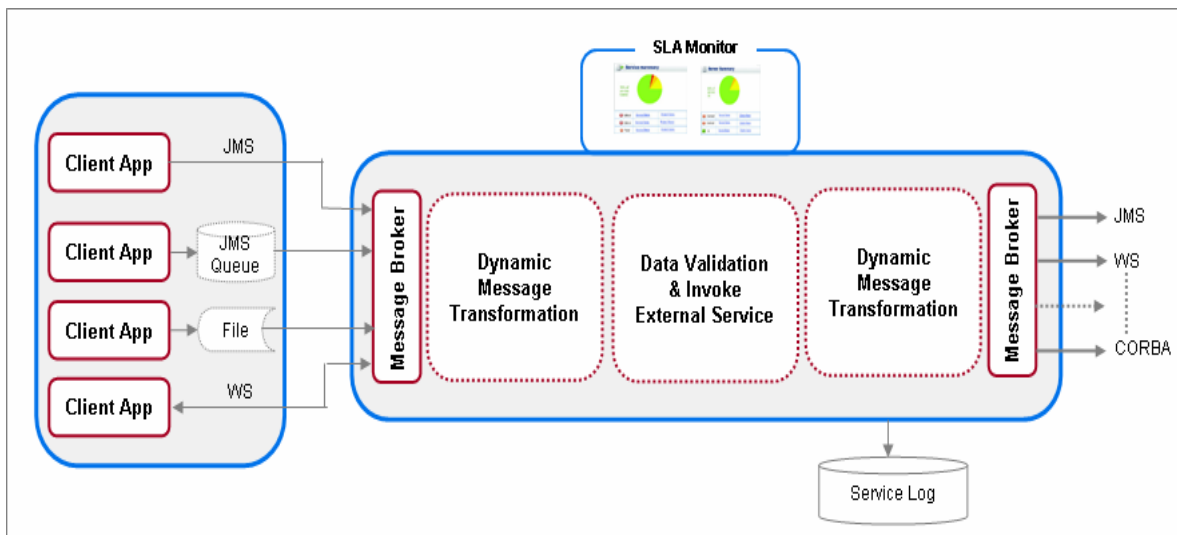


Figure 1: Enterprise Service Bus Architecture



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.



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.

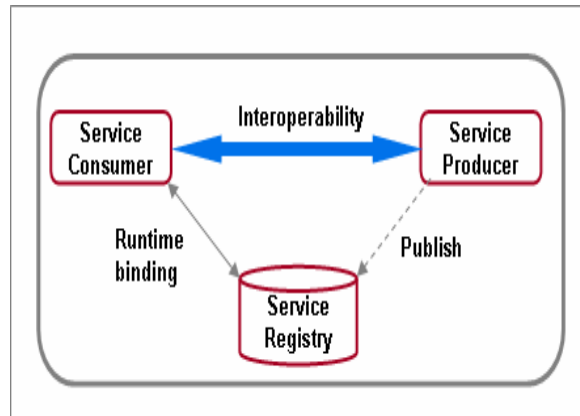


Figure 2: Service Registry

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.



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.





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.



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.

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.

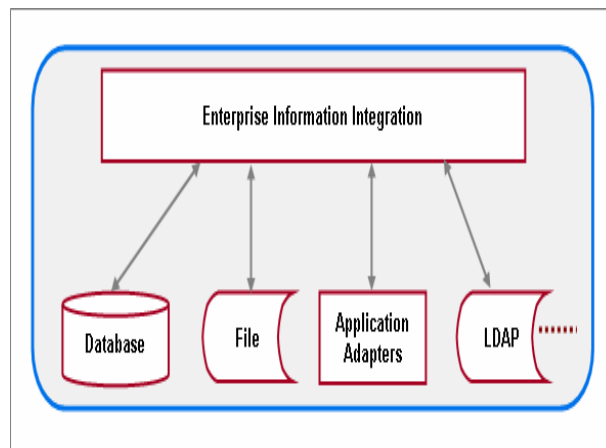


Figure 3: Enterprise Information Integration (EII)

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.

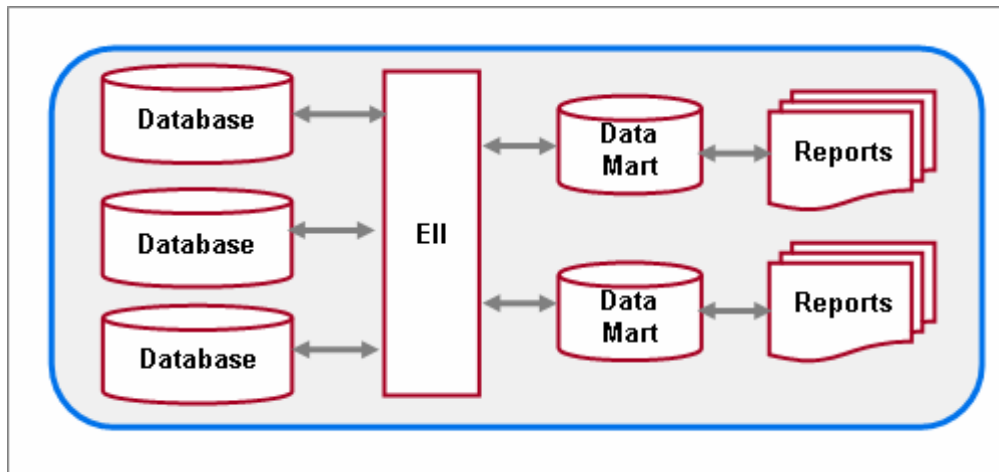


Figure 4: Leveraging EII for Data Marts

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.



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.

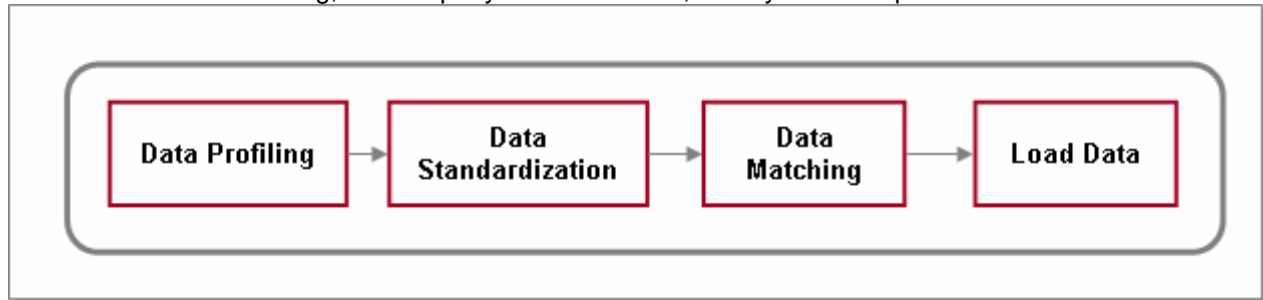


Figure 5: Process of Improving Data Quality

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.

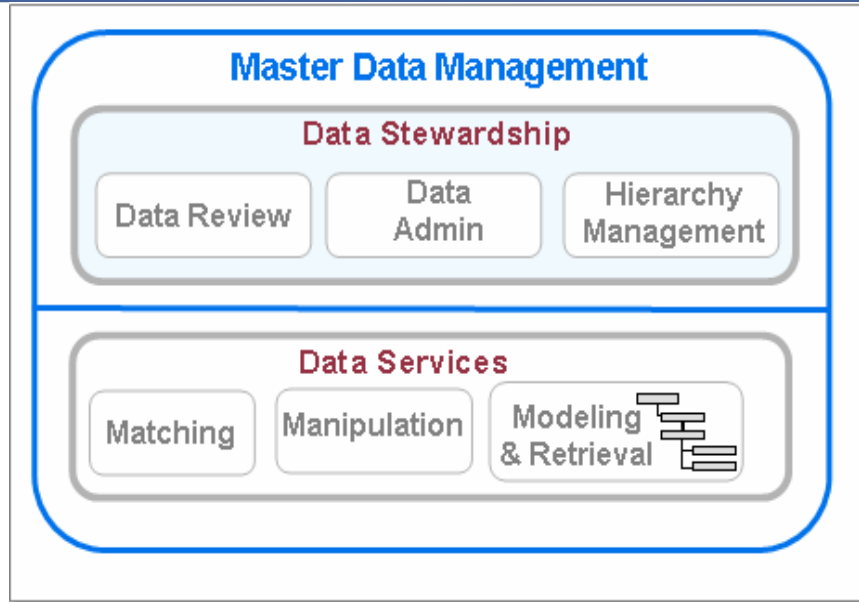


Figure 6: Master Data Management

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.



Best Practices

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

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.

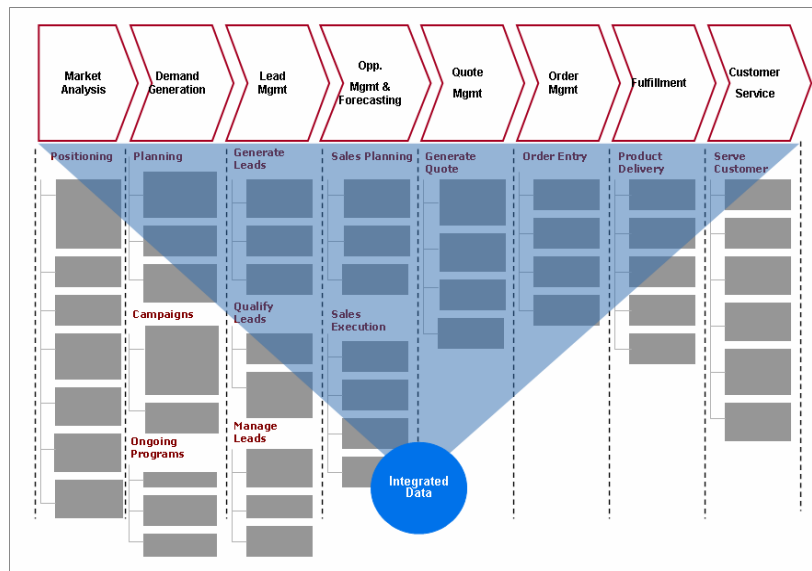


Figure 7: Integrated Data

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.

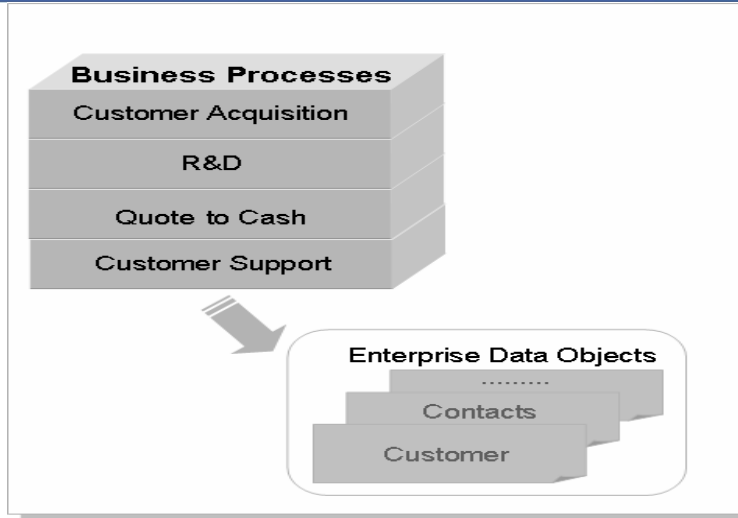


Figure 8: Identifying Enterprise Data Objects

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.

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.

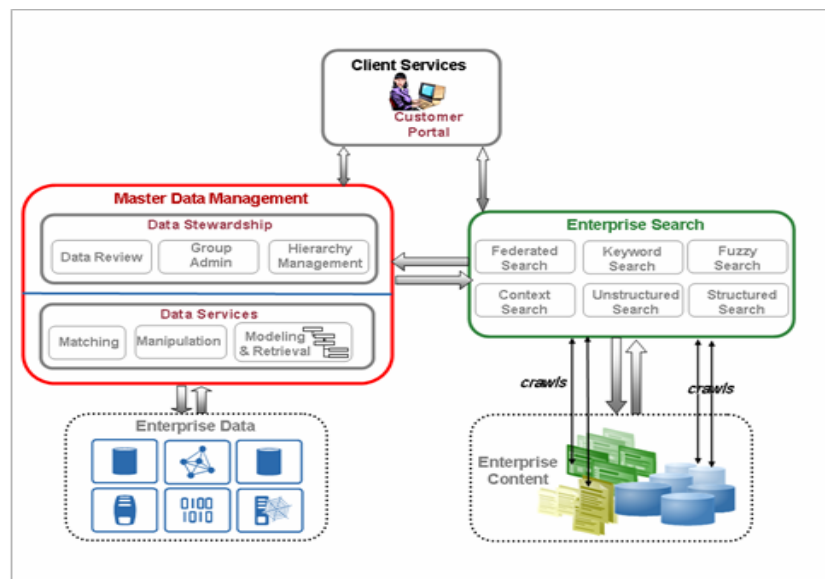


Figure 9: Convergence of Unstructured and Structured Search

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



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.

Copyright

Copyright © 2007.

The author grants a non-exclusive licence to everyone to publish this document in full or part by acknowledging the source. Any other usage is prohibited without the express permission of the authors.

