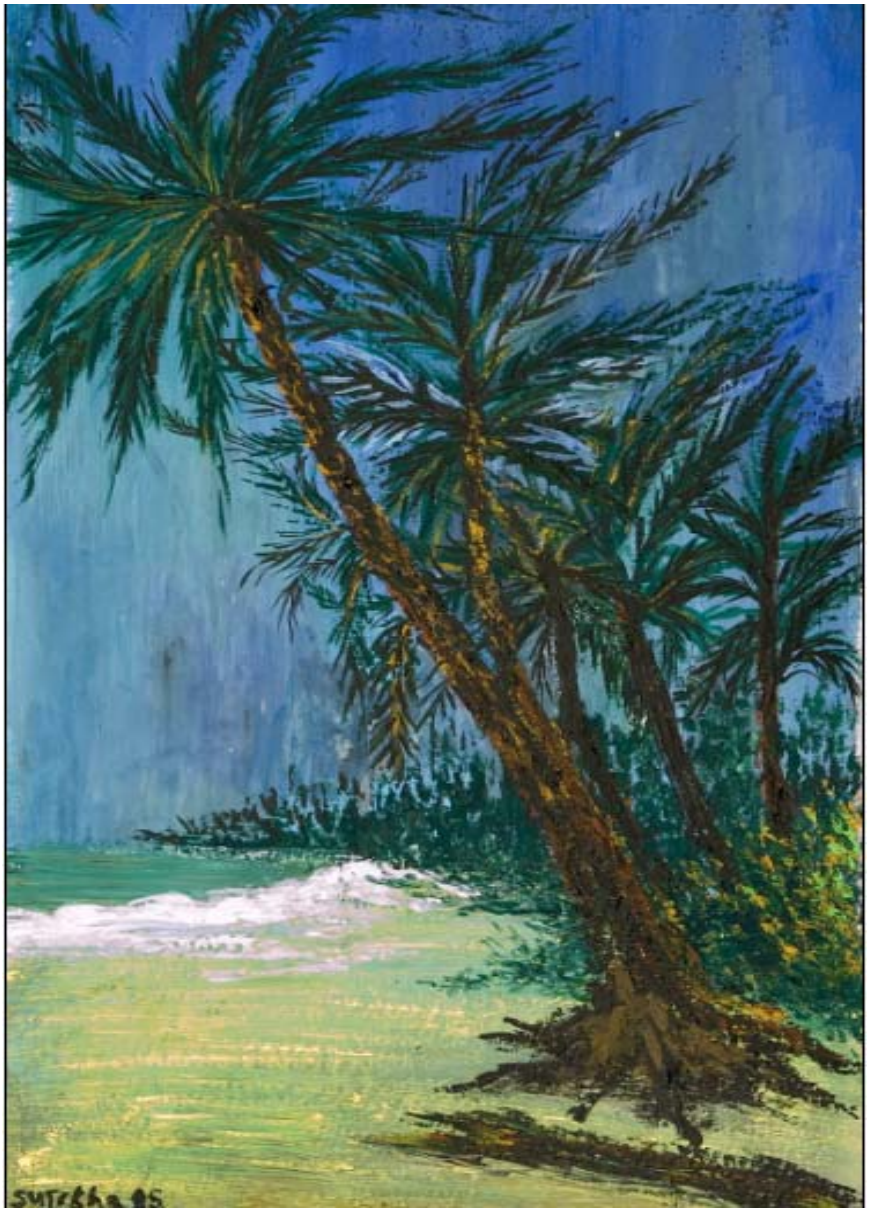
SOA Practitioners' Guide
Part 3
Introduction to Services Lifecycle



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

Prasanna Deshmukh, Director of Architecture, WebEx Communications

Noam Fraenkel, CTO IT, Mercury Interactive

Steve Jones, CTO Application Development Transformation, Capgemini Group

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

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

George Paolini, Consultant, Georgepaolini.com

Jeff Pendelton, Executive Director, SOA Alliance

Annie Shum, VP SOA Strategy, BEA

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

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: Chapters

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

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

Chapter 2: *SOA Reference Architecture* 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.

This is Chapter 3: *Introduction to Services Lifecycle*. It 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 Introduction to Services Lifecycle

2.1 Introduction

After establishing an architecture baseline based on the SOA reference architecture, practitioners should review the services lifecycle. This section briefly describes the service lifecycle and identifies the actors, potential tools, and artifacts associated with each stage of its stages. This document does not cover all the cultural, governance, and organizations changes required to make SOA a success; instead, it focuses on defining best practices for the services lifecycle. The services lifecycle is part of the execution stage in the SOA lifecycle diagram below.

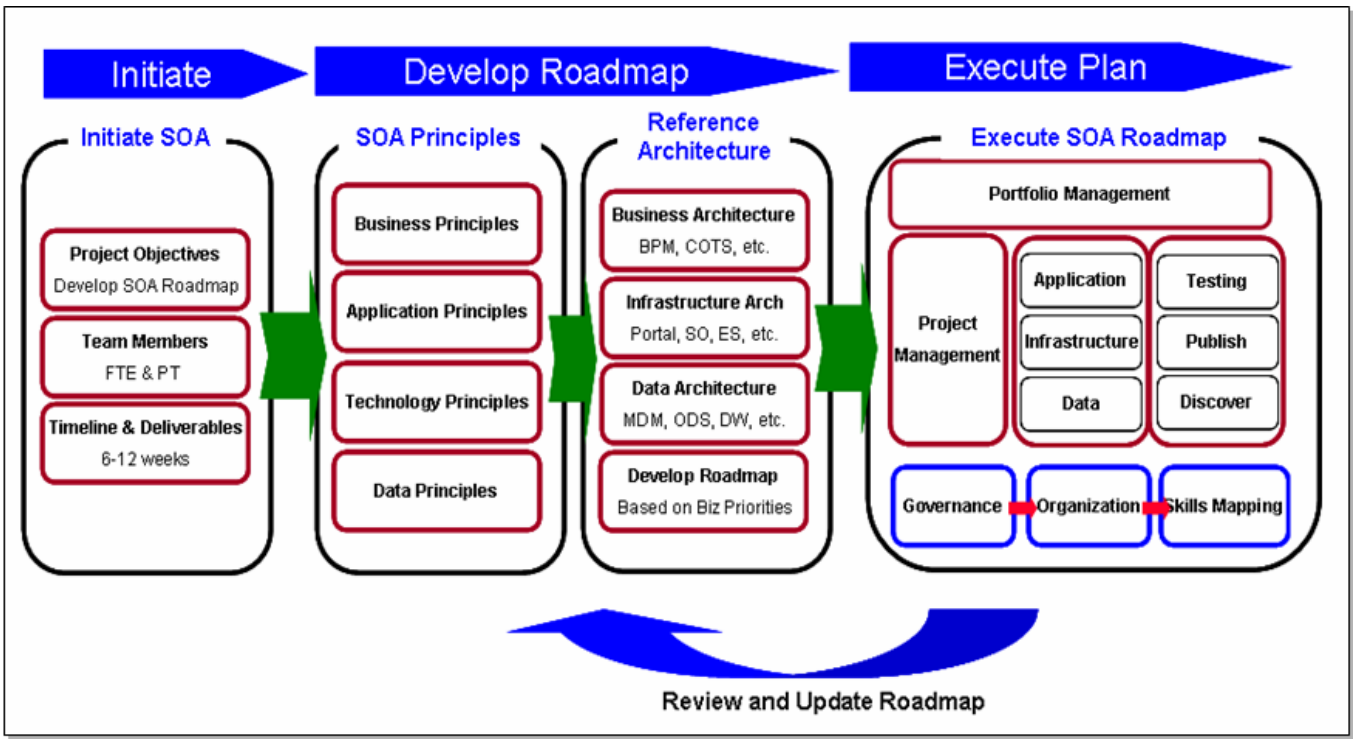


Figure 1: SOA Lifecycle

2.2 Definition

The service lifecycle begins at inception (definition) and ends at its retirement (de-commissioning or repurposing). The service lifecycle enables service governance across its three stages: requirements and analysis, design and development, and IT operations.

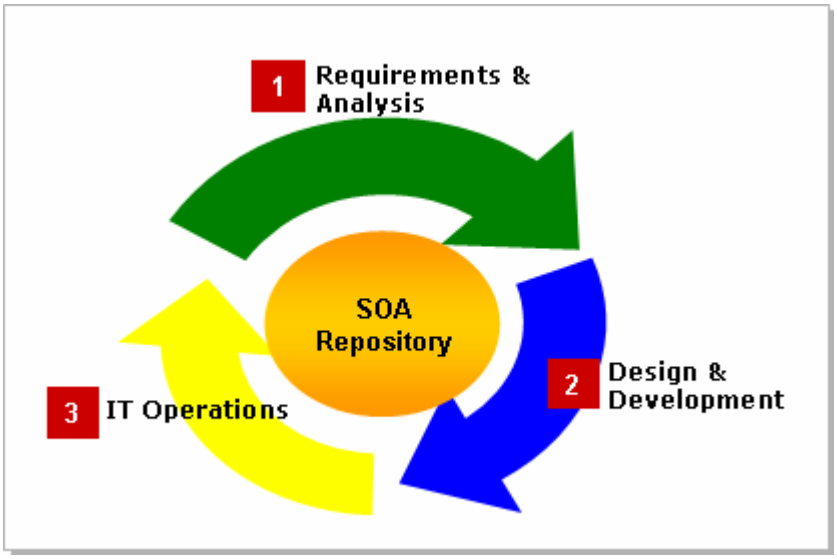


Figure 2: Three Stages of the Services Lifecycle

The above diagram illustrates the three stages and the need for an enterprise service repository to enable service governance.

- **Requirements & analysis:** business initially identifies and prioritizes the business needs. Based on the identified priorities, non-technical staff work closely with business analysts to document the business process, rules, and requirements. High-level requirements include:
    - Visually map business process starting from Level 0 downwards
    - Define each of the business processes
    - Identify business owners for each of the processes
    - Identify objectives and current business services gaps
    - Map Input and output data elements
    - Prioritize business processes and business services
    - Capture all the aspects of business service definitions
    - Simulate user interface and/or business processes.

- **Design & development**: during the design phase, the business analysts work closely with the architect to hand off the business requirements. The architect is responsible for the high-level estimates, design, and handover to the development team. The development teams are responsible for developing, assembling, testing, and handing over the composite application to IT operations. Following are some high-level design requirements:
    - Review requirements and identify alternatives for each business process
    - Design and estimate the components for each of the services, such as portal, integration, infrastructure, data, policy, and business (logical) services
    - Identify reuse opportunities for business services
    - Develop and execute to a detailed project plan
    - Track and report progress to business and IT management
    - Obtain business sign-off at delivery of each business service.
- **IT operations:** this team is responsible for the testing, staging, and production environment with the production environment taking the highest priority. IT operations is responsible for sizing the network and data center. In addition, IT operations is responsible for deploying, monitoring, and providing tier 1 support for all applications supported by IT. Following are some of the high-level requirements:
    - Review requirements and identify infrastructure needs
    - Establish systems environment consisting of development, system integration testing, performance testing, user acceptance, and product environments
    - Assist solutions development teams in systems and application configuration, periodic builds, and capacity planning
    - Track and manage dependencies among services and assets
    - Deploy and manage business services in production
    - Provide application support for business services based on business priority

The details for each of the stages are described later in this section. Following is the high-level IT-process for delivering composite applications to the business.

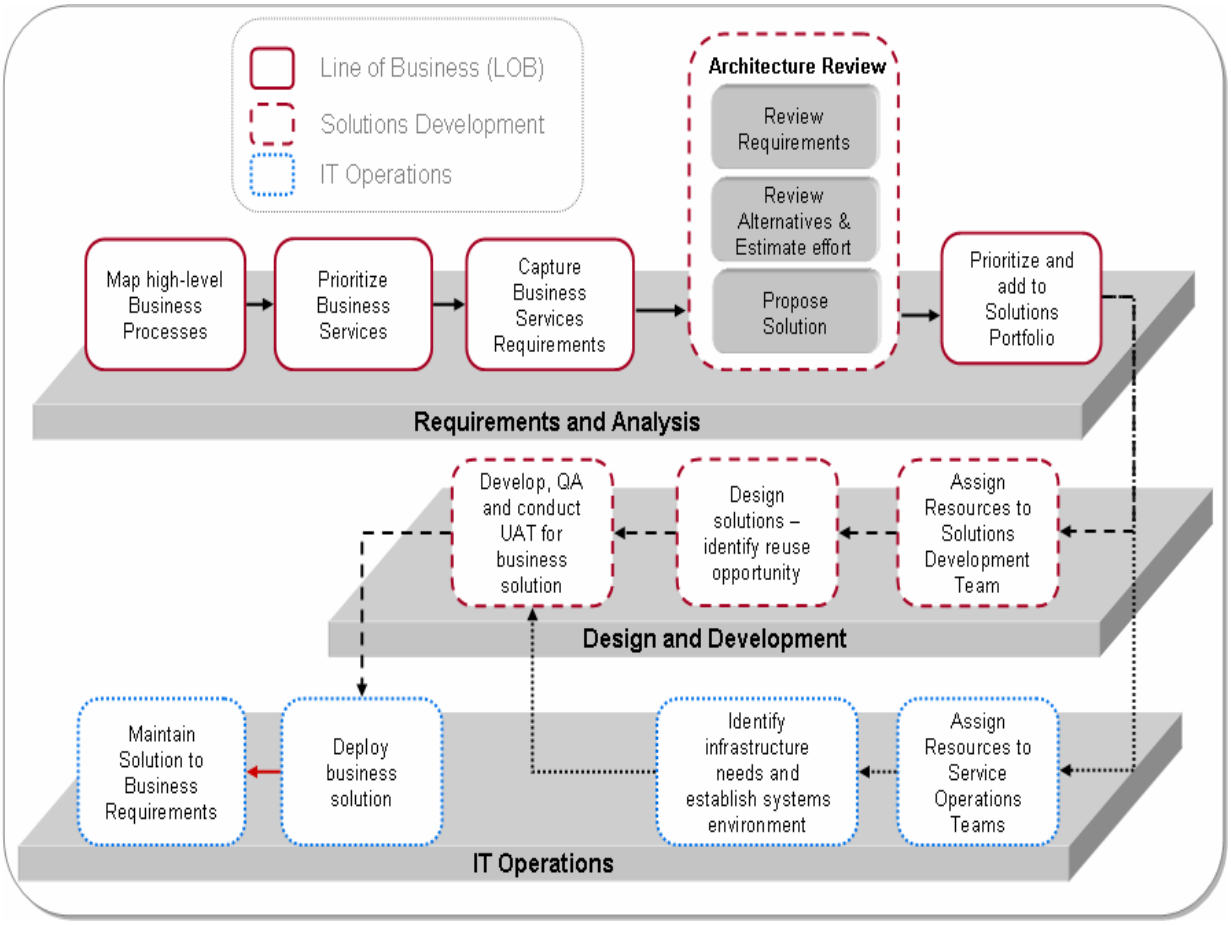


Figure 3: IT-Process of delivering business composite applications

This process also identifies the role of each of the three organizations in delivering the business application.

2.3 Service Lifecycle Governance

Governance is a set of processes, tools, and organizational structure that is essential for delivering on the SOA promise. Effective re-use of services can only be achieved when organizations adhere to standards and follow proper procedures throughout the service lifecycle. As services are shared among applications, organizations must take care in design, development, and deployment of services to ensure that there is no impact on existing consumers of a service.

Services are shared among various organizational silos with conflicting priorities. Effective governance helps ensure maximum re-usability with minimum disruption. The primary responsibilities of the SOA governance function include:

- Publication of SOA standards and best practices
- Definition and execution of processes to promote the use and re-use of services at project level
- To be the custodian of all shared services for the enterprise or LOB
- To be the propagator of standards and best practices across the organization
- Advertise SOA achievements within the organization.

Services governance underpins the entire service lifecycle.

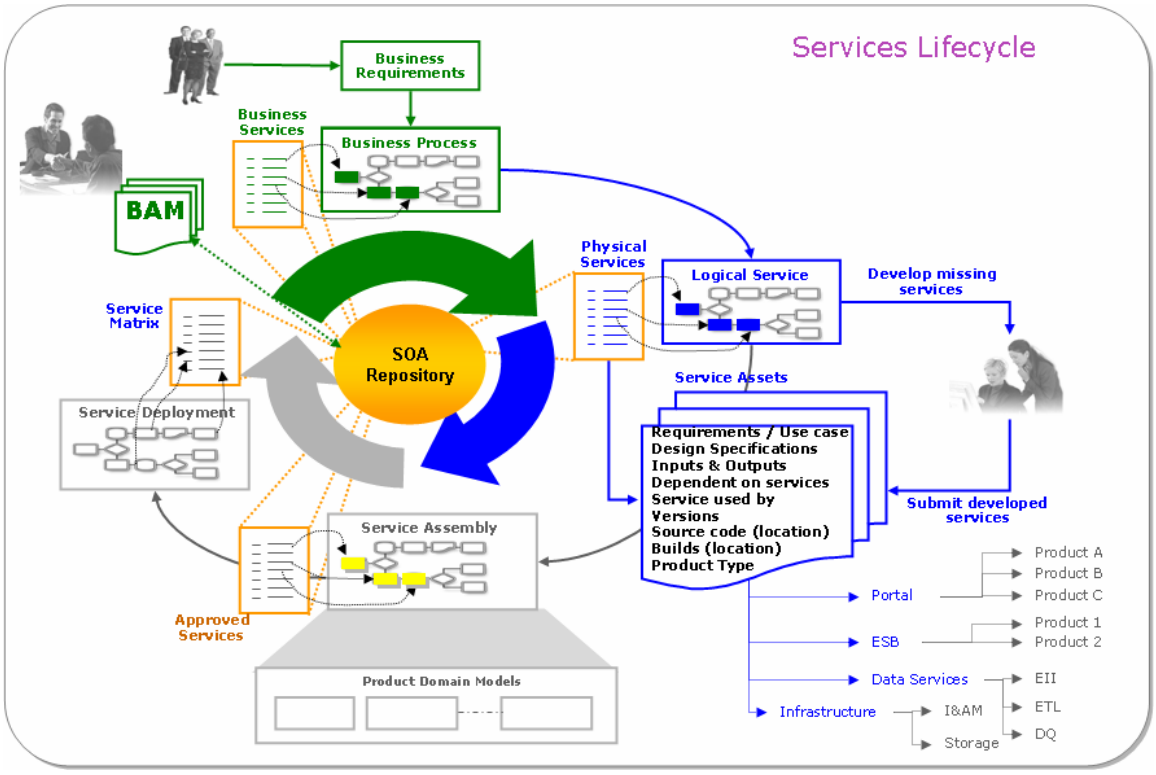


Figure 4: Service Lifecycle Governance

The above diagram illustrates the services lifecycle at a high level and observes the following stages.

2.3.1 Requirements and Analysis

The business analysts work with the business to capture the business requirements, preferably in the form of business processes. For initial SOA projects, teams typically focus on a business process that is

not enterprise- or LOB-wide, but limited to the scope identified by the leadership team while approving the project funding. The team captures the business logic for the composite application being delivered.

Once the team captures the business process, the business analyst identifies any duplication of the process across the enterprise or LOB. The business analyst searches the SOA repository for business processes that the team could potentially reuse. Once this phase is complete, the business analyst uploads the artifacts to the SOA repository, which in turn triggers the governance process.

The governance process should be specific to the organization and the project. Teams should not consider this stage complete until all approvals are in, especially from the business owners.

2.3.2 Design

Business analysts shall pass on the requirements and business processes to the architect to design the application. Each IT organization typically has its own approach or framework for designing applications.

During this phase the architect identifies the services and their implementation. The architect then searches the SOA repository for potential reuse. The architect need not limit the search for services currently deployed in production; the search could be expanded to search for services currently under development by other teams.

At the end of this process, the architect may have identified services for reuse that have already deployed in products, services that need to be modified to create a new version, services that need to be developed, and services that need to be decommissioned.

The architect uploads all the design artifacts to the SOA repository, triggering a governance process that includes approval from enterprise architecture review boards, project managers, and operations. The project manager shall also use this information for distributing the service development tasks.

2.3.3 Service Development

The architect sends development teams the design details, preferably from the SOA repository. The development teams could be distributed in multiple locations, and each team may have a expertise in a business or product domain.

The development teams develop and test the composite application in an iterative manner and upload the artifacts to the enterprise service repository. When the development teams indicate that the service is ready for deployment, they trigger the governance process.

2.3.4 IT Operations

This team is typically responsible for providing the development, QA, staging, and production environment. As the service development organization receives the design details from the architects, IT operations establishes the environment for development. IT Operations often manages the QA environment as well, because it should be identical to the production environment.

The development team typically provides a build to the operations team. For composite applications consisting of services, the development team provides the IT operations teams with the service assembly. The recommended best practice would be to assemble services based on the information found in the SOA repository.

Once IT operations has assembled the services the team deploys those services to the target node. The business analyst and architect would have defined the business, security, and management policies during the earlier stages. It is now the responsibility of the IT operations to monitor and provide metrics to the business to track business KPIs and review IT-SLAs.

The recommended best practice is for the IT operations teams to map the product instance—including hardware, node name, product version, and application version—back to the SOA repository.

2.3.5 Business Dashboard

Business would like to view different types of information that combine data from monitoring systems, operations data stores, and BPM tools. Such information might fall into one of the following categories:

- IT-SLAs
- Business activity monitoring
- Policy management
- Service maturity model (matrix for monitoring the life of the service, and a searchable attribute in the enterprise service repository).

3 Service Lifecycle Stages

This section identifies the actors, tools used, deliverables, lifecycle stage recommendations, lifecycle stage processes, end-user tooling descriptions, and best practices for each of the service lifecycle stages.

3.1 Requirements and Analysis

3.1.1 Actors

- Business personnel (typically business operations from LOB)
- Project managers (business & IT)
- Business analysts
- Architects (optional)

3.1.2 Tools Used

- Business requirements tools including office, business process modeling tool, requirements capturing tools
- Business process modeling tools including BPMN, Visio, and Pro*Activity
- Business rules tools including product rules engines and Word
- User Interface tools including portal simulation tools, Macromedia, Visual Studio, Eclipse, and JSP and HTML editors

3.1.3 Artifacts (Deliverables)

- Design models such as UML, BPM (business process models), data flow models
- Bindings such as JMS, RMI, IIOP, and HTTP(s)

3.1.3.1 Artifact Description

Each LOB defines its own business process, which is captured during this stage of the services lifecycle. Some LOBs may have similar business processes or sub-processes with slight variances. For example, the consumer banking division and the mortgage banking division would be two separate LOBs within a large enterprise with common business processes. Both these LOBs could benefit from documenting and sharing their business processes as well as their key learnings.

The LOB would potentially also run simulations in order to optimize business processes and would use a monitoring and management system to capture and compare actual and simulated results. The business activity monitor provides a dashboard for comparing business results to the established objectives.

The lower levels of the business process definitions would typically be used for developing composite applications. At this level services are identified and mapped to each of the business services (activities). An SOA repository should be the system of record for all these service definitions and dependencies, both for external consumption, and to help IT operations to deploy, monitor, and manage services.

3.1.4 Service Lifecycle stage key considerations

Following are some of the key considerations businesses should factor in during the requirements and analysis stage of the service lifecycle.

3.1.4.1 Business Motivation

Recording business motivation helps map the business process to services. This enables business and IT to have a productive dialog on how to develop and fund the portfolio of services. Mapping helps make the business case for funding the development of the services because it helps businesses understand how services benefit them.

3.1.4.2 Differentiation from Application Lifecycle

Even though the service lifecycle is iterative, it is similar to the application lifecycle. However, one of the best practices for the service lifecycle is to identify existing services that may provide the required functionality. Designers begin by reviewing what already exists to see if it's applicable; this increases the re-use of existing services and saves time.

3.1.5 Service Lifecycle Stage Recommended Process

Following are some of the recommended templates businesses use to initiate projects.

3.1.5.1 Project Initiation Request

Businesses use this template to submit requests for a project. At this stage the business sponsor of the project evaluates whether the project is feasible and engages the LOB-IT or PMO to assist in this effort.

3.1.5.2 Architecture Statement of Work

Once the business submits the PIR to the LOB-IT or PMO, the IT leadership team engages the business to validate that it meets all the initial criteria. The IT organization then establishes a project team consisting of the project manager, business analyst, and architect to work with the business to estimate the effort involved. Depending on the business priority, the team may or may not be working full time on this estimation.

3.1.6 Best Practices and Requirements

This phase begins as soon as the project is funded and the core team is assembled. Following are the high-level best practices and requirements for this stage of the service lifecycle.

3.1.6.1 Portfolio Management

Business and IT jointly leverage portfolio management tools to manage the portfolio of SOA projects. These tools enable business users to submit their project requests. Tools map requests to the IT governance model and help IT manage applications, integration, data centers, and networks. Depending on the particular tool and objective of the IT organization, tools can also be used for resource planning, skills mapping, and monitoring funding for a project.

Portfolio management software helps IT define a roadmap aligned closely to the business by translating business strategy into high-level business prioritization plans. IT organizations can then execute to this plan, while monitoring it to ensure that there isn't any deviation. Functionality supported by this type of tool includes:

- Mapping all the projects to the original business case to help make sure that they are aligned to the business objectives
- Keeping track of all these assets in a single repository (different from the SOA repository) with the capability to highlight duplicate business logic, dependencies, and resource constraints
- Mapping all the dependencies and being able to send alerts whenever a change violates the business rules
- Managing all change requests and defect tracking, including impact on other projects
- Exporting tasks and projects to third-party tools such as enterprise service repository, project management, and development tools.

Typically only large IT organizations require a portfolio management tool, because of the licensing and administrative overhead cost associated with such a tool. A single instance of this tool could be used enterprise-wide or for a specific LOB, unless regulations require IT organizations to deploy multiple instances of such a tool. The biggest challenge of leveraging such a tool is encouraging adoption; the IT leadership team needs to enforce it.

The first task of all IT organizations is to document and communicate the process and lifecycle for managing project requests . This task is typically the responsibility of the program management office (PMO) or the application development team.

There are typically two steps to initiate a project. The business owners submits a request to IT for initiating a project, and IT engages a technical team to estimate the effort for the projects.

Based on the estimates, the IT board of directors may approve or reject the project. If they approve, a team is assembled under the leadership of a project manager to deliver the application to the business. The best practice is to map the entire application lifecycle management using this tool.

3.1.6.2 Requirements Capture

These tools help capture, prioritize, and track the development of all the business requirements. Following are the some of the capabilities that these tools should provide:

- Create a repository of business requirements for a given project
- Provide a single repository for all projects
- Provide capability to prioritize requirements based on cost, resources, and benefits
- Identify and track cross-project dependencies
- Conduct what-if scenarios and provide impact analysis reports
- Assign owners to requirements and tasks and track status.

It does not matter whether the business analysts capture this information using a requirements capture tool or a product like Microsoft Office. What is important is that the requirements are captured in business terms, without technology vocabulary, SQL statements, or references to packaged applications. In addition, the project team should categorize requirements by business functions and have them approved across different phases.

3.1.6.3 User Experience Simulation

With text-based specifications of applications, business people don't get to see and interact with applications until they're already completed. Making changes at this stage leads to cost overruns, time-to-market delays, and miscommunication between business and IT. Prototyping—either with static screen images or low-fidelity coded wireframes— has largely failed to solve these problems, since business people and end users must "fill in the gaps" to picture the full functionality of the application. And prototyping typically siphons off precious development resources, which increases cost and time to market.

New tools can simulate the user interfaces to all kinds of business applications. These tools offer a simple, drag-and-drop paradigm to assemble rich, high-fidelity simulations of applications—including business logic and data interactions—prior to development. The simulations are easy for business people and end users to understand, providing a true "test drive" of the final product. The output is a visual blueprint for what to build, eliminating confusion, cutting costs, and enhancing user adoption. Best of all, no IT resources are required to build the simulations, which can be done quickly by business analysts, user-experience designers, project managers, and architects.

The following capabilities should be provided by this class of simulation tool:

- Drag-and-drop assembly of high- or low-fidelity simulation screens
- Ability to link data and business logic into the simulation
- Collaborative, team-based definition environment
- Capture and documentation of requirements in context of the simulation
- Built-in support for use-case scenarios and workflows
- Reusable definition assets
- Ability to encapsulate a simulation in a document that can be e-mailed.

People don’t adopt business applications unless they’re easy to use—or at least, easier than what users were working with before. But teams often don’t test applications with end users until they are almost ready to deploy. This can have costly consequences if users demand changes. A new class of tools solves this problem by letting usability experts quickly create and test high-fidelity simulations directly with users in a rapid, iterative process prior to development.

Simulation is now a best practice for development organizations that want to move the process of usability testing to the front of the software development lifecycle (SDLC) process. By working closely with business analysts and developers early in the process of defining applications, teams quickly and inexpensively can fix many of the common problems associated with poor usability. A centralized user experience team can provide the following benefits:

- Standardized best practices around usability and user testing
- Education, training, and mentorship to the rest of the design and development team
- Creation and maintenance of reusable simulation (definition) assets that reflect corporate best practices
- Feedback and direction early in the SDLC
- Process guidelines, style guides.

3.1.6.4 Business Process Modeling

The BPM tool is used to capture the business processes during the requirements stages. The tool should provide the following capabilities:

- **Business process modeling:** capability for business users to easily model processes, define business rules and key performance indicators (KPIs), and simulate, test and develop end-to-end process flows.
- **Business activity monitoring:** real-time and historic analysis and reporting capability. Real-time process monitoring, escalation and management helps quickly identify any problem in the business process quickly to help resolve it.
- **Business process execution:** execution of the automated component of the business. This includes orchestrating all resources—people, organizations, applications and systems—to ensure flawless execution and exception management.

The business process modeling tool can help capture business processes during the requirements stages. This tool is generally used by business analysts to model the business process using common industry notations provided in the tool and based on standards such as BPMN and UML. Typically the business process modeling starts from Level 0 and processes are further defined to lower levels, as required. The business process could be modeled with a view to achieving one of these objectives:

- To simulate a new process through multiple scenarios before committing to the resources for executing it. For example, a line of business is focusing on business optimization because it is entering a new market, rolling out a new product, or starting a marketing campaign.
- To align IT more closely with the business. In this case, the business process modeling tool helps capture and share the business process—and maybe even screen flows—to help build consensus across multiple teams and geographies.

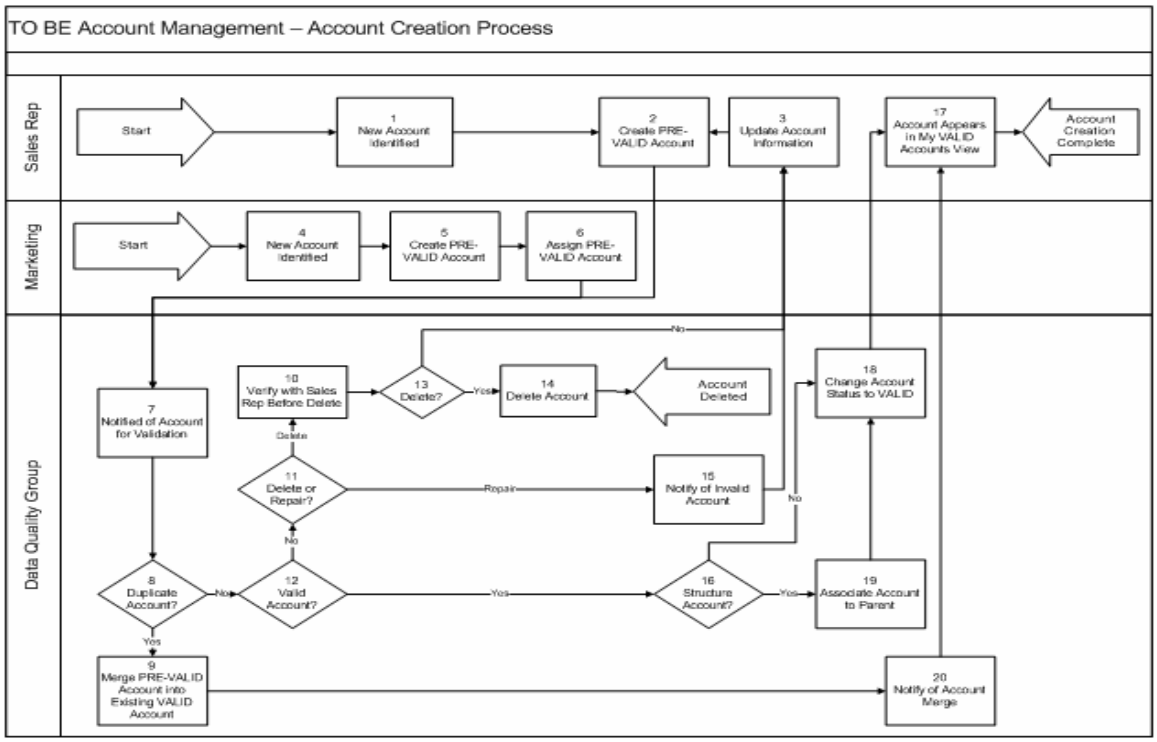


Figure 5: Business Process Modeling

The above diagram illustrates a typical business process modeled during the requirements phase. The business analyst interacts with the business to capture events, manual and automated activities, and

business rules. The architect may participate in these meetings as an observer to better understand the rationale driving the business process.

The business analyst interacts with the architect to define and refine the next level of details during the analysis phase. Once the business analysts and the architects have modeled the business processes to the lowest level that the business can define without going into the technical details, they jointly review business services in the enterprise service repository to determine whether the services already exist or will need to be built. BPM tools help them in this process by enabling them to:

- Capture all business requirements in the form of business processes such as activities, rules, and policies
- Simulate end-to-end business processes to identify bottlenecks and improve overall process
- Review business processes globally and invite other LOBs to participate in these discussions
- Capture local and regional requirements
- Capture both manual and automated processes.

3.1.6.5 SOA Repository

This is the repository that enables automating the governance process across all products. It stores all the relevant product metadata defining the business processes, requirements, and simulation parameters. In addition, enterprise architects can also upload to this repository the enterprise standard documents consisting of patterns and architecture frameworks. The SOA repository should also enable both IT and business to modify the business process to match their internal governance.

3.2 Composite Application Design

3.2.1 Actors

- Project manager (IT)
- Business analysts
- Enterprise architects
- Project architects
- Designers
- Technical leads or lead developer

3.2.2 Tools Used

- Design: Rational, Together Architecture, Eclipse, and others

3.2.3 Artifacts (Deliverables)

- Design models: UML, SCA service assembly model, and others
- Bindings: JMS, RMI, IIOP, HTTP(s), and others

3.2.3.1 Artifact Description

This stage of the service lifecycle generates models that represent the system flow, data flow, enterprise data model (represented in the form of Entity Relationship Diagram(ERD)), application design (represented in UML), activity diagram, and sequence diagram. During this phase, the team also generates the high-level deployment model, identifying the servers, OS, middleware, databases, firewall, and load balancers.

The application designer could be an architect, technical lead, or lead developer, and may decide to use some of the tools in the market. Typically this tool is based on RUP or a variation of RUP and would consist of artifacts such as activity diagrams, use cases, class diagrams, ERDs, deployment models, and deployment models. Architects should share these artifacts with the team for review and approval, and provide instructions to the development teams.

3.2.4 Service Lifecycle Stage Key Considerations

3.2.4.1 Enterprise Architecture Framework

IT organizations should standardize on an architecture framework that defines architecture standards, development processes, design patterns, and tools. Most IT organizations will already have adopted one of the standard application lifecycle management processes and modified it to fit its own needs. In addition, there are other well known architecture frameworks such as Zachman, Federal Enterprise Architecture (FEA) and The Open-Group Architecture Framework (TOGAF). IT must adopt an architecture framework enterprise-wide to be successful in implementing SOA.

3.2.4.2 Services Classification Framework

A services classification framework helps provide a basis for design and development of services and is essential to achieving a flexible architecture. Services could be classified into multiple categories such as:

- SOA reference architecture, including shared data service, business process, and portal service

- Portfolio of services, including quote-to-cash services and mortgage approval services
- LOB services, including sales and support services.

After the services have been identified, the team classifies them based on the defined standards. Classification helps business managers, project managers, and the development team to identify what services are being developed and for what purpose. This makes it easier for the project manager to distribute development tasks to the appropriate teams.

3.2.4.3 Service Granularity

Service granularity refers to the level of abstraction or how much functionality the service covers. Teams can apply the concept of granularity either to the service itself or to the service methods.

In determining service granularity, architects need to consider performance requirements. Fine-grained services such as Enterprise Java Beans (EJBs) are typically easier to understand and implement, since in many cases much of the work is already done. However, if performance is a consideration, then aligning services with existing EJBs may *not* turn out to be the optimal solution. A fine-grained architecture that relies on multiple request and response pairs (a "chatty" protocol) may offer slower performance. In order to reduce the effects of network latency, system I/O, and thread/process wait states, it is much better to create a coarse-grained service that internally composes multiple business domain services and uses fewer messages.

Service granularity must take into account legacy system interfaces that are (and must remain) unaware of the new protocol. Architects must plan carefully to avoid any changes to legacy systems when adding a new data channel in the form of the Web service.

Finally, in determining granularity architects must consider the possibility of future changes to the underlying implementation. In general, businesses benefit from using coarse-grained facades or patterns to hide the fine-grained services beneath them. The goal is to insulate services from changes to the underlying implementation by designing them at a level of granularity that will allow for future expansion with little impact to the clients.

3.2.4.4 Reuse Strategy

Reuse of a component starts at the design layer. Architects should design components so that the client use of that component only executes or inherits the methods that are needed to perform a given task. Shared components must be written as standalone elements that perform a task while hiding its complexity. Architects should consider using the facade pattern, as it tends to encapsulate each member of a related task or class within a common interface so that client code may use those tasks interchangeably. Since a task is usually coded as one or a few isolated methods, encapsulation emphasizes reuse of methods that perform a single task. Reusing a single task is easier than reusing objects containing code and data, which may perform multiple tasks.

3.2.5 Service Lifecycle Stage Recommended Process

This phase begins as soon as the project manager or business analyst hands over the requirements to the technical team. The team then begins composite application design, typically by following these steps::

1. Architect downloads requirements from SCM/SOA repository, reviews them, and submits them to business analyst for further clarifications, if required
2. Architect selects tools to model and design the application
3. Architect identifies services and may search the SOA repository to identify potential reuse
4. Architect defines services, implementations, bindings, and dependencies
5. Once the design is complete, architect uploads all design artifacts to the SOA repository for approval
6. Architecture review board—consisting of enterprise architects, project architects, business analysts, and the project manager—reviews design to ensure architecture meets business requirements and is consistent across the enterprise.

The SOA repository should provide capability to generate service templates for each service category based on patterns defined by the enterprise architects.

3.2.6 Best Practices and Requirements

This phase begins as soon as the requirements for the composite applications are approved. This section describes the high-level best practices and requirements for this stage of the service lifecycle.

3.2.6.1 Service Orchestration (Modeling)

During the requirements and analysis stage of the services lifecycle, the team should capture all requirements in the form of business processes, including business rules and policies. This enables reuse, at both the business process and services level. Most BPM tools provide two separate tools: a BPM tool for the business analyst to capture requirements, and a service orchestration modeling tool for the architect. The service orchestration modeling tool typically includes the BPM tool and enables the architect to develop the service orchestration or flow model for each business service or activity. Most of these tools either generate the code or create metadata for executing the logic.

Best practices for service orchestration modeling include:

- Leverage the BPM tool for service orchestration modeling
- Adopt open-standards based tools for modeling (such as BPMN) and orchestration (BPEL)
- Have business analysts focus on business process modeling and architects focus on service orchestration modeling
- Have architects develop the services orchestration model for composite applications even if the business analysts haven't used a BPM tool to define the business requirements
- Upload all service orchestration models to the SOA repository.

3.2.6.2 Service Composition

Most leading software vendors have agreed to a new standard called the service composition architecture (SCA). The SCA standard defines services, service dependencies, service implementation, services composition, and the deployment and runtime aspects of developing composite applications. In addition, open-source projects are under way to develop the Java and C++ runtime for SCA, such as the Tuscany project under Apache.

The SCA standard defines the following:

- How to auto-generate code based on the metadata via the mechanisms of byte-code enhancement, dependency injection, and aspect-oriented programming (AOP)
- Mechanism to extend the annotations (controls framework)
- Annotations that come out of the box for Java "plain old Java objects" (POJOs), business process execution language (BPEL), Web services, and .NET or J2EE components
- Configuration parameters and deployment options for a service
- List of mechanisms to instrument and monitor a service.

Additional information about this proposed standard is available at http://www.osoa.org

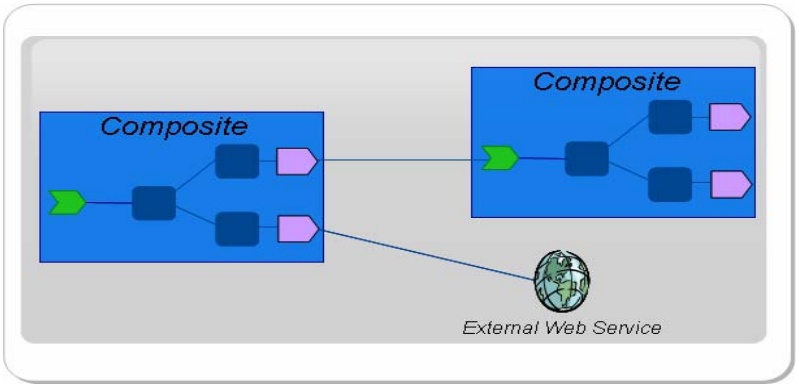


Figure 6: Service Composition Architecture

The above diagram demonstrates at a high level the service assembly model, in which multiple composites form an application. The composite consists of component services.

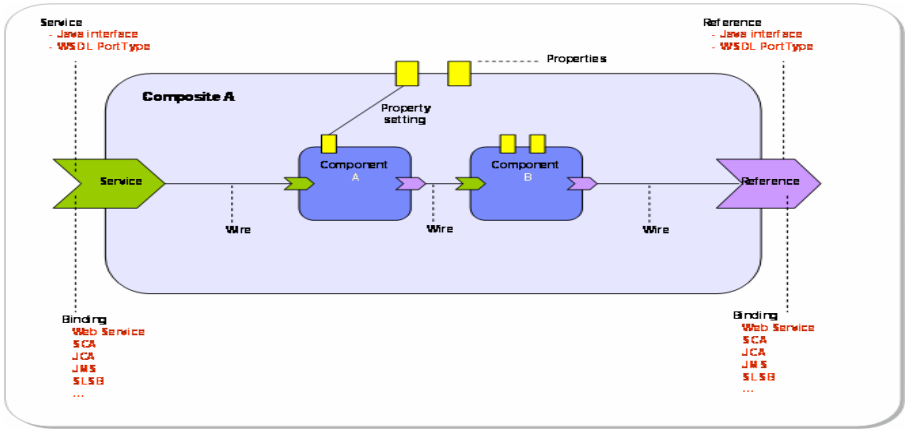


Figure 7: Service Composition Architecture

Multiple tools enable architects to compose services. Architects need the following capabilities:

- Map each service to business activities. This is called service orchestration, and it is the lowest level of the business process.
- Compose services based on SCA. Architects define the service, implementation, properties, interfaces, and bindings based on SCA. The development team then leverages this service model for developing and modifying the service.
- Identify which business rules could be externalized and embedded in code for better performance. This is only appropriate for rules that are relatively static; rules must often change to meet the needs of the business. A rules engine should interpret the rules accurately based on the rule parameters.
- Define the business, management, and security policies associated with each of the services
- Identify and define measurement matrixes for enabling business groups to review their KPIs.

3.2.6.3 SOA Repository

The SOA repository is the system of record for all enterprise metadata, service definitions, and dependencies. All metadata generated by the tools used by the architect/designer is uploaded to the SOA repository. It provides information required by the consumer as well as additional details required by IT operations for managing the service. The repository defines not only what metrics to capture while monitoring a service from a support perspective but also provides for technical services that are provided by vendors. These technical services can provide dashboards that trigger events such as service alerts for business users.

Teams can also use the SOA repository for service governance, to support the process of submitting designed artifacts to appropriate reviewers for approval. Only after all approvals have taken place does the project manager assign tasks to developers and IT operations.

While there are currently no standards for repositories, several vendors have repositories with capabilities for managing SOA metadata. While there are several 'hub' repositories that are available for complete enterprise metadata management, some vendors concentrate solely on SOA repositories that store a subset of metadata, such as SCA models, BPEL, XPDL, JPD, and WSRP.

Following are the best practices for the SOA repository:

- Search the SOA repository for potential reuse, including business processes, policies, and services
- Review the SLA and service details before identifying the service for reuse
- Upload all the architecture framework documents, enterprise and application models, and standards to the SOA repository
- Define the SOA repository workflow to enforce both design time and runtime governance across the enterprise
- Leverage the service registry (UDDI) for integrating the SOA repository with other repositories within the enterprise.

3.2.6.4 Service Registry

The service registry is based on UDDI and the system of record for all deployed services. It contains metadata including service definitions, service dependencies, and service interfaces. Service registries are now also being extended to act as repositories, although they do not store the assets themselves and are focused more on runtime metadata, a subset of metadata from an SOA repository.

The best practices for the service registry are to search it for potential reuse of services, and review the SLA and service details before identifying a service for reuse.

3.2.6.5 Information Modeling

There are different types of modeling associated with information modeling

- Reference data modeling
- Master data management (such as CDI and PIM)
- Shared data services
- Use of domain standards where applicable (HR-XML, XBRL, ACORD, MDDL, and RIXML for standards-based service payloads).

IT organizations rarely model reference data, the common data such as country names, state names, and state codes. However, partial standardization of reference data may be required to provide a consistent user experience across the enterprise.

Following are some of the best practices for information modeling

- Identify all systems, data flow, and attributes between systems, business owners, and system of records, then make it mandatory for every project team to update this model
- Identify the data flow while capturing the business requirements
- Identify shared data services and master data entities while reviewing the application design
- Develop the ERD for each custom application
- Update the ERD whenever a packaged or custom application is modified.

Refer to SOA Practitioners Guide Part 2, *SOA Reference Architecture* for information on master data management and shared data services.

3.2.6.6 Application Modeling

Most teams use modeling tools based on UML, as well as tools that enable architects to model sequence diagrams, activity diagrams, data flows, system flows, and network diagrams.

Packaged application vendors provide the capability to design and model their products and would recommend that IT organizations adopt the documented best practices associated with those applications.

3.3 Service Development

3.3.1 Actors

- Project manager
- Architects
- Development teams
- Release management
- IT operations

3.3.2 Tools Used

- IDE (for example, Visual Studio, Eclipse, JBuilder, JDeveloper, and BEA Workshop)
- Packaged application development tools
- Regression and performance testing tools
- Build tools (for example, Maven, Cruise Control, ANT, and shell scripts)
- Source control management systems

3.3.3 Artifacts (Deliverables)

- Source code
- Product specific metadata for configuration as well as service execution
- Java documents
- Release notes

3.3.3.1 Artifact Description

The artifacts produced during this stage include service definition, configuration, contract, and application configuration, as well as management, security and business policies at the service, application, department, and enterprise level. The application configuration also includes implementation configuration.

3.3.4 Service Lifecycle Stage Key Considerations

Tasks for this stage of service lifecycle include:

- Define criteria for when to outsource development of a service
- Perform technology review process prior to using a new technology within IT
- Assess skills required for the entire lifecycle of the service prior to development of service
- Map each service back to the business requirement
- Evaluate existing services to eliminate duplicate development.

3.3.5 Service Lifecycle stage recommended process

Developers can start developing services as soon as project managers or architects provide them with the design documents. These could be office documents or models based on enterprise architecture standards.

Developers can also start developing or modifying services as soon as business or IT operations opens a change request(CR). In this case the architect may not be involved. The application support technical lead or developer will review the change request and make the changes. Depending on the level of change and the IT PMO process, governance may be triggered when the developer uploads the metadata change to the SOA repository.

Service creation typically follows this process:

1. The application design or the change request is approved for development.
2. In case of new development, the designer may already have created service templates and assigned them to the developer. The developer downloads the service templates and completes them.
3. In case of new development where no templates exist, the developer either uses a development tool to create the service or searches for a similar service in the SOA repository and copies it over as a template.
4. In case of a change request, the developer downloads the code and modifies it.
5. In all cases, the developer also downloads the requirements and design artifacts for review and may modify them after clarifications from architects, project managers, or business users. The service governance process is triggered whenever a developer modifies the requirements or the design artifacts.
6. If the architect has wired services based on SCA, the developer could leverage an SCA-based tool to double click on the component to confirm that the service is created.
7. If the service being created is to be consumed by services being developed by the other teams, the developer should define the service and develop and configure the service simulation. The service producer does this by defining responses to sample requests and publishing them for consumption. This enables consuming services to test out the contracts prior to the service actually being developed.
8. If the service being created is consuming a service being developed by the other teams, the developer can leverage the service simulations produced by the other team.
9. The developer goes through multiple interactions to develop and test the services. As the regression test is generally end-to-end, the developer can either use the simulated testing environment for unit-level testing or move the service through the various testing stages.
10. The developer periodically synchronizes the metadata with the SOA repository. The SOA repository validates all the metadata before accepting it and also triggers the service governance process, if required.
11. The developer generates documentation, such as Java documents and release notes, that are synchronized with the SOA repository.

This describes the traditional services creation and maintenance process. Most software vendors now also provide business users the ability to create composite applications using a portal. Business users can create composite applications by loosely coupling various user interactions and interactions processes, and dynamically creating forms based on service definitions. The business user can perform most of these tasks directly in production. Tools provide the capability to develop, test, and deploy the service, and synchronize the metadata that drives these composite applications to the SOA repository.

3.3.6 Best Practices and Requirements

These checklists cover what an SOA team should be looking for in development and testing tools, and how it should leverage the SOA repository.

3.3.6.1 Development tools

- Standards-based
- Ability to import/export metadata based on standards
- Backward compatibility
- Advance notice from vendors of major changes
- Ability to generate code from models, wherever appropriate
- Integrative development environment
- Easy tooling and running configuration capability
- Debugging capability
- Hints during development
- Document generation based on comments and code (Example: Javadocs)
- Integration with development and testing frameworks, for example: JUnit, Cactus, and SOA framework components
- Easy creation, modification, or review of a service by double-clicking on the service icon
- Service composition model for existing services, independent of the implementation.

3.3.6.2 Testing Tools

- Ability to define test plans
- Ability to develop test scripts, map them back to the test plans, and develop them for automated regression testing
- Capability to stub functionality for unit level testing
- Capability to simulate services
- Regression and performance testing capability.

3.3.6.3 SOA Repository

- Check in and check out metadata from SOA repository (SCM is the system of record for all source code)
- Upload all metadata to SOA repository at the end of each development milestone
- Leverage SOA repository for obtaining approval through the various development stages
- Leverage SOA repository for facilitating reuse, architecture standards review, and design pattern adoption.

3.4 IT Operations

3.4.1 Actors

- Project manager
- Architects
- Release management
- Build teams
- IT operations

3.4.2 Tools Used

Tools used include product deployment tools such as Maven, Cruise Control, Ant, shell scripts, and IDE-based tools such as Eclipse, Visual Studio, and packaged application deployment tools.

3.4.3 Artifacts (Deliverables)

Chief deliverables are the product domain model, packed application configuration, builds, and service dependencies.

3.4.3.1 Artifact Description

The primary output of this process is the product domain model which teams can leverage to assemble the builds for each of the services (if required), assemble all the metadata for products, and map the URLs that are physical end-points for running the services. Teams develop the physical deployment model from the logical model (SCA). Once the team has assembled the services for deployment, it needs to create the network topology, server configuration, service configuration, and server and server asset management. In addition, the team must also create runtime artifacts such as monitoring and event logs for correlation later downstream.

3.4.4 Service Lifecycle Stage Key Considerations

There are two key considerations for the stage of the service lifecycle: service deployment, and service managing and monitoring.

3.4.4.1 Service Deployment

Composite applications consist of a service or portfolio of services that can be implemented on various technologies. Release management requires planning builds at a service level and also providing the capability to assemble services for a target node. In addition, IT operations needs to put together plans and processes for modifying service and instance configuration parameters while assembling or provisioning the service.

IT operations might require a service assembly management server to help manage the various builds, assemble the services for target nodes, and manage assets. There are products in the market that provide this capability, but unfortunately there aren't any standards on service packaging or service provision. This is potentially an area for the SCA standards to address.

3.4.4.2 Service Management and Monitoring

Service management and monitoring can be sub-classified into two more areas:

- **Server management:** configuring the deployed services and server instances, ideally with a universal administration console. This should be consistent across all products and all the vendors if possible.
- **Server monitoring:** applying the monitoring policies to capture the required matrix for each of the services. This matrix should be based on the KPIs and management policies defined during the requirements and analysis stage of the services lifecycle. Teams also need to collect and correlate monitoring events to ensure the business service level agreements (SLAs) are met and also to publish the aggregated management summary information to the service registry and/or SOA repository.

3.4.5 Service Lifecycle Stage Recommended Process

This phase begins whenever the development team is ready to conduct integration testing, user acceptance testing, or performance testing, or is ready to deploy the application to the production environment. IT operations should follow these steps:

1. The development team completes the development and modification of services and performs unit-level testing as well as integration testing using the simulation capability.
2. The development team identifies the services and assembles them into a project that is ready for testing or deployment to the release management team.
3. The build or release management team uses product tools to assemble builds for deployment. These will prompt the user to enter all the physical end points and application parameters. In the testing environment, the user should be able to enter the URLs of the test services already running in the QA environment or point to the simulated services. When services are ready for deployment or production, users enter the URL for the services.
4. If the service is being decommissioned, the team creates a new project to decommission the service and redesign all services dependent on it. Sometimes, not all dependent services are defined in the SOA repository. Following is a process to mitigate risk.
    a. Create a project to redesign and deploy all services dependent on the service to be decommissioned.
    b. Keep the service targeted for decommission up and running and monitor it to confirm that it is not being consumed by any other service.
    c. Deploy the project and create a new project to decommission the old service.
5. IT operations should use the deployment tool to assemble the services project for deployment.
6. Once IT operations deploys the service into production, the team can leverage various tools to monitor the service, application, middleware, OS, hardware, and network.
7. IT operations is responsible for keeping the services up and running. IT operations escalates support problems caused by application-specific issues to the application support team.

.

3.4.6 Best Practices and Requirements

When the development team is ready to deploy the application to the QA, staging, or production environment, they can benefit from knowing about these high-level best practices and requirements for this stage of the service lifecycle.

3.4.6.1 Release Management Tools

- Leverage existing application lifecycle management tool or a source control management system to support release management
- Leverage open-source tools such as ANT, Maven, or Cruise Control for periodic builds
- Leverage SOA repository for service governance
- Review market for new products that might help build and deploy applications at the individual service level
- Centralize the release management team either at the LOB or enterprise level
- Involve a release management team right from the inception of the project.

3.4.6.2 Deployment Tool

- Leverage existing deployment tools and procedures
- Search for automated application service configuration and provisioning deployment tools if the IT organization does not already have this capability; enterprise management system vendors may supply such tools
- Leverage deployment tools to track of all the changes made to the runtime environment for compliance requirements
- Coordinate deployment with the program management office
- Use deployment tools to reassemble services from one network topology to another.

3.4.6.3 Enterprise Management Systems

Enterprise management systems should provide the following capabilities:

- Application management
- Asset management for network elements, OS, software, and patches
- Business service management
- Configuration management (both service and server)
- Monitoring
- Complex event processing
- Correlation engine
- Automated actions
- Diagnostics and root cause analysis
- Unified console
- Business dashboard
- Contract management.

See SOA Reference Architecture – Business Service Management section for additional details.

3.4.6.4 SOA Repository

Organizations should leverage the capabilities of their SOA repository for the following:

- Manage network topology
- Manage service deployment network
- Review service dependencies and SLAs along with CMDB for capacity planning and network topology development
- Upload all metadata to SOA repository at the end of each deployment
- Obtain approvals during deployment and management stages.

3.5 Business Dashboard

3.5.1 Actors

- Executives
- Business operations
- CIO staff
- LOB-IT executives
- Business analysts
- Project managers
- Architects
- IT operations

3.5.2 Tools Used

- Data marts
- Operational data stores
- Business intelligence tools
- Portal-based dashboard

3.5.3 Artifacts (Deliverables)

- Star schemas and snowflake schemas
- Dashboard usage models
    - Scorecard metrics definitions
    - Rollup and drill-down rules and business algorithms
    - Data lineage and origin-of-information rules for audits and synchronizations

3.5.3.1 Artifact Description

A business dashboard is a shared presentation service. Artifacts capture the metadata necessary to roll up and drill down business rules.

CWM is the metadata modeling notation and framework used to capture the models for the data warehouse and datamarts. XMI is the modeling exchange or interchange notation built on the XML standard. Together, CWM and XMI allow the metadata captured in the data warehouse and datamart schemas to be exchanged across multiple vendor tools in XML, for use by other modeling tools and EII-based services.

The metadata repository includes the business rules for the roll-up and business algorithms used for deriving metrics, for leverage by the semantic translation layer of most BI tools. These metrics form the basis for business scorecards that report on key business drivers or aspects of the value chain. A business dashboard brings one or more related scorecards together to show the impact to a business driver or value chain business process. For instance, the scorecard could report on improvement in vendor relationships or support staff productivity. A single dashboard could incorporate both of these to monitor the business driver of customer satisfaction. Additionally, there may be a related scorecard on repeat sales that also records customer satisfaction gains.

Most business dashboards also provide drill-down capabilities. BI tools leverage lineage and information derivation paths to provide more detailed information about an aspect of the scorecard.

3.5.4 Service Lifecycle stage key considerations

Building a business dashboard requires identifying the criteria, metadata and rules that feed into it. Dashboard-based views and scorecards are tied to both the LOB and the user roles, with authorization rules that link to user roles to ensure that the right level of information is accessible to authorized users. Security aspects, granularity of information, and governance rules as defined by SOX and other regulatory bodies may also affect dashboard design.

Dashboard designers need to plan to ensure accuracy of both derived scorecards and the visualization of metrics, as well as lineage information to allow accurate drill-down capabilities. Each user community may have different drill-down requirements that must be factored into accessibility and authorization level rules. Designers must also ensure that granular row-level data can always back up rollups and the derived information. This supports accuracy and auditability.

3.5.5 Service Lifecycle stage recommended process

Typically organizations develop business dashboards when one or more operational and analytic services are already available and have established credibility with LOB users. Dashboards are a composite service for pulling together scorecards and interacting with value chain business processes that may impact the metrics being reported on real-time basis. Some of the drill-downs may lead to operational data or operational data stores. More sophisticated dashboards may subscribe to real-time business events to provide up-to-the-minute information. Business dashboards may also be linked to alerts and exception reporting services such as a BAM service.

3.5.6 Best Practices and Requirements

3.5.6.1 Business Intelligence

Business intelligence solutions may include the following:

- Analytics tools for decision support systems (DSS)
- Alerting engines
- Business event analytics
- Executive dashboards and business score cards.

3.5.6.2 Portals

Technologies such as JSR 168 portlets and WSRP-enabled federated portals help embed or consume DSS-based analytics views and scorecards. These technologies support shared presentation services.

3.5.6.3 Correlation between the IT Service QoS Monitoring Metrics and Services SLA needs

In order to perform end-to-end monitoring and reporting of Quality of Service (QoS), organizations need metrics that correlate response time and availability with the SLA for service turn-around time. In other words, the service contract specification must be charted against the service runtime execution metrics of response time, availability, and performance metrics.

Most organizations end up manually correlating business process and business-function QoS metrics with SLA expectations and service contracts. Application server Implementations and ESBs report QoS metrics into a database for mapping back to SLAs in the service registry or a repository. This mapping can take place at the database level or using XQuery when the SLA and the runtime service performance metrics are expressed in XML

To automate this process, organizations need a common semantic language and metadata for specifying SLAs, QoS, and runtime performance metrics. Also, the capture schemas for the SLAs and the runtime performance metrics may need to be standardized to allow automated correlations and mappings. With this level of automated mapping, a business dashboard can present a composite scorecard of service monitoring metrics and SLAs . This allows the business to evaluate how service runtime impacts the service turnaround time.

4 Appendix

4.1 IT Stakeholders

Depending on their roles, individuals within companies may define SOA differently. This makes it important to spell out the roles of IT stakeholders who may be involved in SOA.

4.1.1 IT "Board of Directors"

Just like all major corporations, IT also needs a "board of directors." This function is usually performed by key executives or their representatives. The board's objectives are to set direction, approve initiatives and projects, and resolve conflicts. Some organizations refer to these boards as information services steering committees (ISSCs), information technology review boards (ITRBs), or information technology leadership teams.

4.1.2 Chief Information Officer

The CIO is responsible for all the aspects of IT. Some organizations have multiple divisional CIOs reporting to a global CIO. Divisional CIOs have a lot of autonomy in some organizations, but the recent trend is to consolidate the enterprise architecture and enterprise shared services team under the global CIO for faster adoption of SOA and a consistent approach across the enterprise.

4.1.3 Program Management Office (PMO)

The PMO is responsible for orchestrating projects across the enterprise or LOB. It acts as the primary contact for all cross-functional activities and ensures that each project team follows the standard process defined by the enterprise architects.

4.1.4 Business Sponsor

The business sponsor champions the application within the business and is ultimately responsible for ensuring that the project is successfully adopted. Typically an officer (VP or above) of the company, this person is usually responsible for securing the funding and completing the business transformation.

4.1.5 Project Team

The project team is tasked with delivering a business capability. The team develops the strategy for delivering the capability and investigates available options. Typically every project team has a project manager. Project management responsibility may be shared between one person from the LOB and one from IT.

4.1.6 Architects

There are multiple categories of architects.

4.1.6.1 Enterprise Architects

Enterprise architects define standards, processes, and design patterns, and identify new technology.

4.1.6.2 Project Architects

Project architects design business solutions or applications.

4.1.6.3 Information / Data Architects

Information or data architects ensure a consistent approach and model for handling information and data across the enterprise.

4.1.7 Business Analyst

The business analyst captures business requirements, policies, and rules.

4.1.8 Architecture Steering Committee

Almost every project or program has a steering committee. Especially for SOA, there needs to be a steering committee for enterprise architecture too. Members of this committee include the IT leadership team and key stakeholders from business operations.

4.1.9 IT Operations

The IT operations team is usually responsible for the data center, security, networks, and tier 1 support, “keeping the lights on.” This team is not usually responsible for application-specific issues.

4.1.10 Business Operations

The business operations team defines and documents operations processes. Each member is responsible for an area of the business, for example, sales operations makes sure that all requests from sales are reviewed and accepted if appropriate.

4.1.11 Chief Technology Officer (CTO)

In some organizations, the CTO is responsible for enterprise architecture within IT. The CTO may also act as the CIO.

4.1.12 Enterprise Shared Services

The enterprise shared services team develops shared services for the enterprise. Sometimes this is a dedicated shared services team, sometimes it is part of the architecture team, or sometimes each project team develops its own shared services.

4.1.13 Chief Process Officer (CPO)

A CPO, typically a peer of the CIO, may report to the COO or president of a company, and is responsible for defining business processes across the enterprise.

4.1.14 Chief Security Officer (CSO)

The CSO is responsible for enterprise security strategy and implementation. He or she usually reports to the CIO.

4.1.15 Project Managers

Project managers are responsible for delivering projects on time and under budget. They could be either from a LOB or IT. The standard best practice is to have one person responsible for project delivery, but increasingly, companies are making a business project manager and an IT project manager jointly responsible for getting the project delivered.

4.1.16 Application Support

Organizations may choose to support applications with a dedicated centralized team, an application support team within a LOB, a permanent project team with developers rotating to application support roles, or they may choose to outsource application support.

4.2 SOA Governance and Organizations

SOA governance and organization best practices is complex. This section focuses only on a few observed best practices.

4.2.1 SOA Development Organization

4.2.1.1 Introduction

Best practices recommend develop the organization model by defining the target state, developing the roadmap, and identifying components that need to be built and how. Governance and organization end up being a natural outcome of this exercise.

An alternate approach is to build the organization model based on the tasks and not the components that need to be built. This section briefly describes some of the organization models that enable faster adoption of SOA.

4.2.1.2 Traditional Development Approach

Most IT projects fit into one of these categories:

- **e-business solutions:** portal applications for both internal and external users
- **Packaged applications:** best-of-breed point solutions
- **Integration:** integration of applications, portals and data across the enterprise or LOB
- **Infrastructure:** data center, networks, servers, and software platforms.

The business sponsors and IT leadership team prioritize projects and monitor progress periodically.

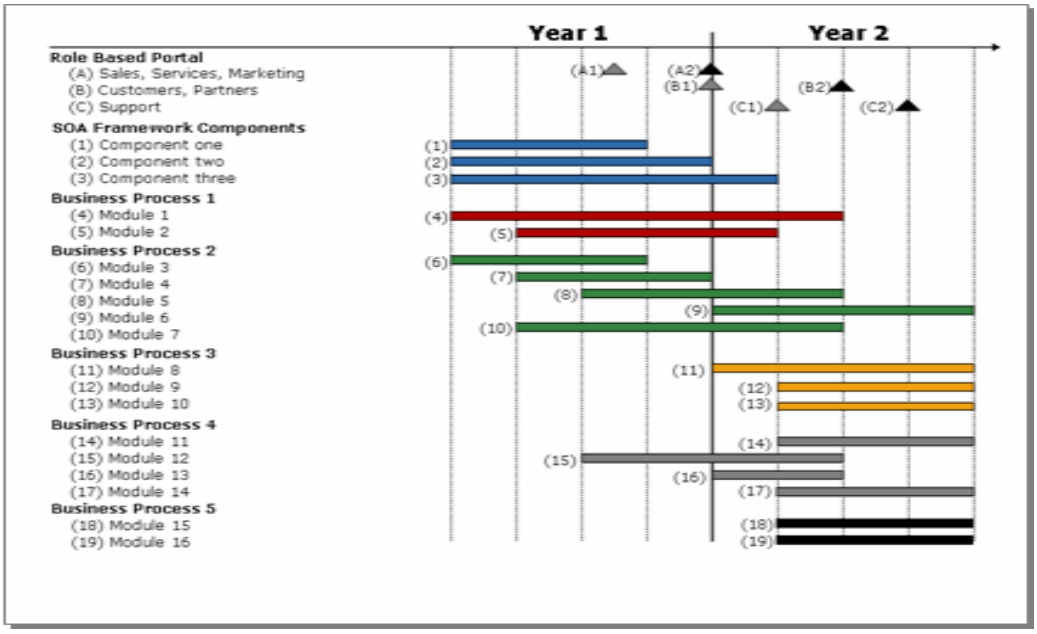


Figure 8: Typical IT roadmap

The traditional development lifecycle applies consistently across all the initiatives as shown below, but resource allocation varies based on the initiative category. The following diagram illustrates a typical development lifecycle.

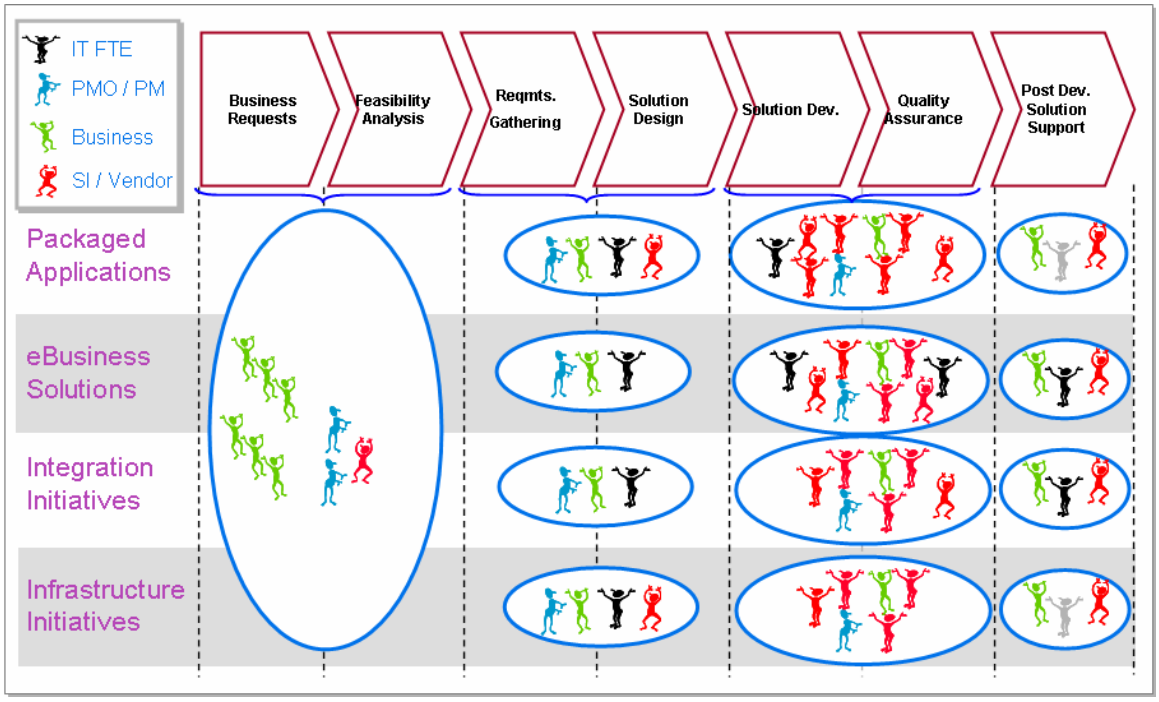


Figure 9: Typical IT Development Lifecycle

Typically a team consisting of business operations, PMO, project manager, and the IT staff of business analysts and architects review the high-level business requirements of each initiative. Based on their analysis, the IT staff provides the joint leadership team with a recommended approach and estimates for funding approval. Even though the lifecycle is the same, the approach is different for each type of initiative.

- **Packaged applications**: the typical approach is to outsource development to a system integrator or the product vendor and later off-shore support of these packaged applications to reduce cost. This is considered a best practice because packaged applications are proprietary and developing these skill sets in-house as that wouldn't provide a strategic value to the company. While enterprises could standardize on one packaged application vendor and build an in-house team, this is not a practical solution for large corporations. The corporation would be too much exposed to the business issues of its vendor. Besides, most LOBs want packaged solutions specific to their own needs.

- **e-business solutions:** most enterprises look at this as a strategic investment, because this is one of the fastest-growing channels for customers, partners, suppliers, and employees to interact and collaborate with each other. The preferred approach is to dedicate the company's best architects and developers to creating and supporting these solutions, and to build these skills sets within the organization. Developing in-house expertise makes sense, especially for consumer-focused organizations, because it facilitates making rapid changes to the customer-facing solution to help the organization keep its competitive edge. For some lower-profile business functions, support could potentially be outsourced or off-shored.

- **Integration:** integrating e-business solutions with packaged applications and later with the platform for BPM is complex. Execution depends on the quality of interaction between the various teams, so outsourcing is not the most appropriate approach. Instead, the industry best practice is to embed all the customization into the middleware. The LOB should stick to providing the business processes and rules and resolving open issues related to business definitions. Enterprise architecture teams should make the integration decisions.

- **Infrastructure:** the best approach to infrastructure varies depending on the size of the organization. Organizations should keep this in-house until they grow to over $1 billion in revenue. Larger enterprises may out-source infrastructure to a third party, which then makes it easier to estimate the real cost of infrastructure. Keeping infrastructure in-house could overload IT operations staff, resulting in longer turn-around time and reduced efficiency.

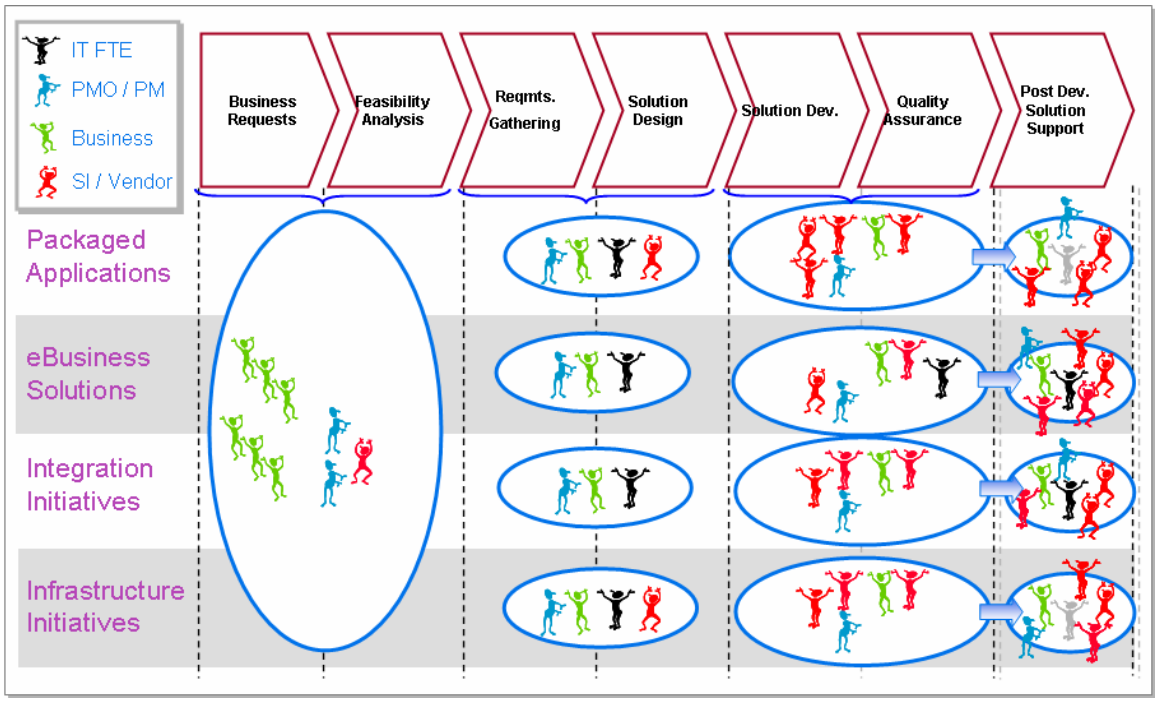


Figure 10: Result of the current Lifecycle approach

One of the major drawbacks of the traditional lifecycle approach is that as IT delivers new capabilities, resources get diverted to support these capabilities, increasing the cost of the capability. IT may need to bring in additional resources to develop new services or provide support for existing services. IT organizations cannot halt this trend but can slow down the cost of new development by outsourcing and later off-shoring the support function. However, this approach can only take the IT organization so far. Enterprises need to develop a different approach for staffing their development organizations.

4.2.1.3 Recommended Approach

The recommended approach is to create teams based on technical capabilities, as follows:

- **Composition team:** wires the services together based on business processes and requirements. The wiring of services not only identifies the capabilities required by each service but also the deployment, configuration, and management model. This team should consist of architects and technical leads.
- **UI team:** develops the business interaction layer which consists of the wire frames, user interaction flows, navigations, and type validations. Development of this layer could be

outsourced, provided there an excellent process in place to capture and document the requirements.

- **Services team**: designs and develops the business logic. The recommendation would be to staff the key positions in-house, especially the architects and key developers. The development itself could be outsourced as long as it is based on the architecture guidelines and managed and reviewed by internal staff.
- **Data team**: develops the shared or specific data services for the other teams. Information and data architects are the key members of the team and should be company employees. They define the enterprise data model, data quality, and common objects such as customers, orders, and products. The development itself could be done anywhere but has to be based on the architecture guidelines and managed and reviewed by the information and data architects. While the other teams receive services from this team, they do not need to understand the enterprise data model.

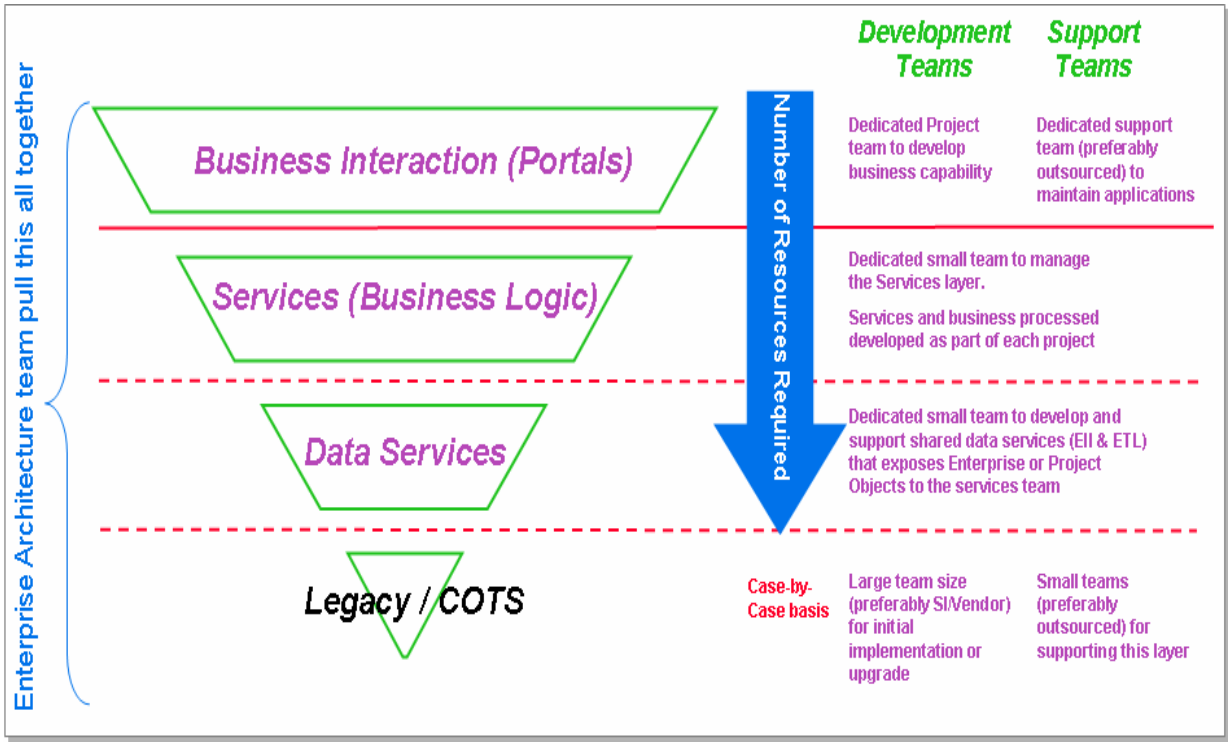


Figure 11: Organization based on capabilities

4.2.1.4 Summary

This recommended approach organizes development teams based on their capabilities. It results in lower costs because it eliminates the need to staff every project team with similar skill sets.

4.2.2 Enterprise Architecture: Roles and Responsibilities

4.2.2.1 Mission Statement

The purpose of an enterprise architecture is to develop IT strategy and provide technology vision aligned to changing business priorities through thought leadership, processes, structure, and value delivery.

4.2.2.2 Enterprise Architecture Responsibilities

Following are the high-level responsibilities of the enterprise architects:

4.2.2.2.1 Develop IT Strategy and Technology vision

- Review the target (future) state of how IT shall handle the enterprise's business processes, information, technology, and applications to achieve the business objectives in phases that enable business to execute the company mission
- Review the current state to identify gaps and develop an actionable roadmap for achieving the target state, including business processes, organization, information, technologies, and applications
- Assess alternatives and make recommendations based on a holistic view of the costs, benefits, and dependencies associated with each alternative.

4.2.2.2.2 Develop Architecture Standards

- Develop and publish the service lifecycle methodology that spans defining the business applications to design, development, deployment, support, and upgrade of underlying technology and applications
- Develop detailed checklist of the roles and responsibilities of each of the teams including deliverables and exit criteria for each lifecycle phase
- Define (lifecycle) processes, such as rapid application development methods, and common tasks such as project management, reuse, metrics, and testing.

4.2.2.2.3 Define Technology Lifecycle

- Identify emerging technologies that support the actionable roadmap
- Define technology standards and where each of them can be used
- Obtain information on emerging software and manage relationships with strategic partners.

4.2.2.2.4 Assist with Skill Planning

Whenever the enterprise architecture team identifies a new technology for adoption, it must identify the new required skills sets for achieving the target state. All other teams shall leverage this as part of their skills planning exercise

4.2.2.2.5 Information and Data Architecture

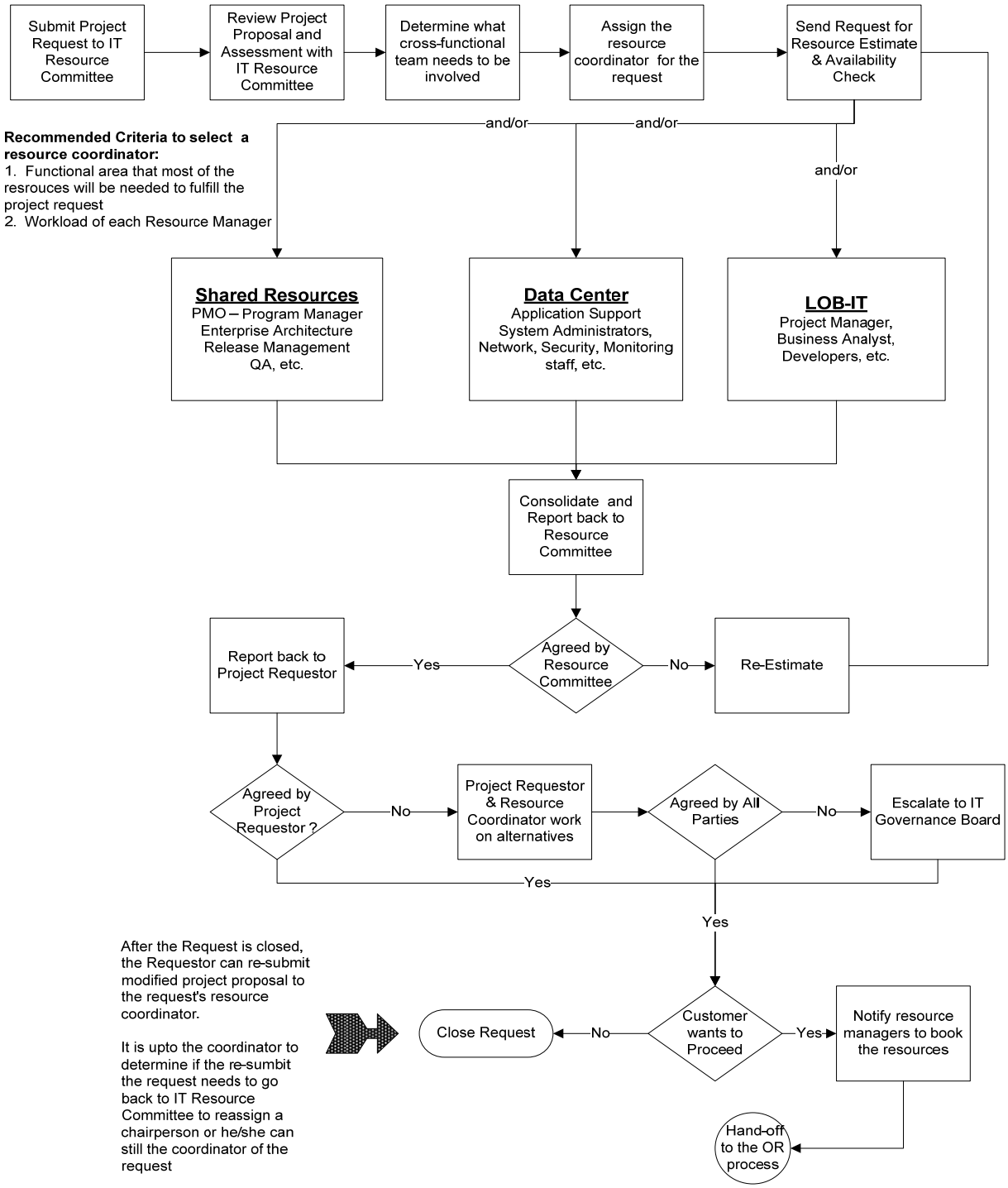
The enterprise architecture team partners closely with the business to develop an enterprise data model, especially for common objects such as customers, products, and orders. The team must also develop models for the data warehouse, datamart, shared data services, and roadmap.

4.2.2.2.6 Integration Architecture

The role of the enterprise architecture team is similar to city planning. They may not define every building, but they define the connections and infrastructure, in this case, the key systems, modules, components and relationships between major components of the IT infrastructure.

4.2.3 IT Enterprise Resource Management Process Flow

Following is an example of a high-level IT enterprise resource management process flow.



4.2.4 Project Initiation Request Form

REQUESTOR - Name:	Request Date:
PROJECT - Name:	Additional Stakeholders/Beneficiaries:
Business Sponsor Name:	Request # (LOB-IT Use):
Business Sponsor Organization:	Request Status (LOB-IT Use):

REVIEW & APPROVAL (LOB-IT Use)		
Business Programs Office:	Review Date:	
LOB IT Staff:	Review Date:	
LOB IT Review Board	Review Date:	

BUSINESS PROBLEM DEFINITION & JUSTIFICATION

BUSINESS PROBLEM:	*Describe the current business problem this project or tool enhancement request would address. Please include a description of the business process it would support.*
BUSINESS JUSTIFICATION:	*Provide qualitative and quantitative information to support investment. Include organizations affected; number of users, productivity gains, revenue gains, cost savings, etc.*
BUSINESS OBJECTIVES:	*List objectives you want to address; including, where appropriate, how they tie to company or organizational goals*
BACKGROUND INFORMATION:	*Provide any additional information about this request, including previous efforts to resolve; who was involved; new solutions or software you have considered; potential process or organizational change impacts, etc.*
LOB IT INFORMATION:	*Provide any information about this request that is related to IT.*

4.2.5 Request for Architecture Work

Business analysts hand the request for architecture work (or request for estimation) to the architects after capturing the high-level requirements from LOB. This request shall contain the statement of work and information from the project initiation document.

Statement of Work

A short description of the work required.

Project Request and Background

The business is to provide the information detailing their request with any additional background information that shall help architects estimate the effort.

Architecture Statement of Work

The architecture statement of work contains the estimates for the proposed phases, timeline, milestones, and cost.

Project Request and Background

Project Scope, Benefits & Cost

SCOPE:

Define the scope of this project or in phases of this project. Include:
a. any business process changes
b. system functional features that support or automate the above changed processes.

PHASE 1

- ❑
- ❑
- ❑

NOT IS SCOPE - BOUNDARIES:

*Identify what will **not** be in the scope of this project or in phases of this project. Include:*
a. any business process changes
b. system functional features.

NOT IN PHASE 1

- ❑
- ❑
- ❑

ROI (Return On Investment) – Benefits & Costs

BENEFITS:

***DIRECT BENEFITS** - List the measurable, direct benefits that this project will deliver to BEA upon completion and that justify the ROI of this project. Address the key benefits in terms of: operations cost reduction, revenue generation, productivity gain, and customer satisfaction.*

- ❑
- ❑
- ❑

- ❑

INDIRECT BENEFITS *- Identify other indirect or intangible benefits and advantages this project will deliver.*

- ❑
- ❑
- ❑

COSTS

RESOURCES *- Estimate the required resources as listed below for a successful implementation of this project and ongoing system operations and support.*

ALREADY BUDGETED

- ❑ PERMANENT & CONTRACT HEADCOUNTS:
- ❑ SOFTWARE & HARDWARE:
- ❑ IT INFRASTRUCTURE – NETWORK & TELECOM:
- ❑ TOTAL BUDGET:

NOT BUDGETED

- ❑ PERMANENT & CONTRACT HEADCOUNTS:
- ❑ SOFTWARE & HARDWARE:
- ❑ IT INFRASTRUCTURE – NETWORK & TELECOM:
- ❑ TOTAL ESTIMATED BUDGET:

Risk Management

DEPENDENCIES ON OTHER PROJECTS:

List any dependencies on other projects for the start and completion of this project.

- ❑
- ❑

CRITICAL SUCCESS FACTORS:

Identify any critical success factors that need to be considered from the onset of this project.

- ❑
- ❑
- ❑
- ❑

ACCEPTANCE CRITERIA RISK:

List the project acceptance criteria

Identify any potential risks that may negatively impact this project.

- ❑
- ❑

Roles and responsibilities

DEFINE BUSINESS STAKEHOLDERS

List all business stakeholders and their roles within the project

- ❑

- ❑

DEFINE IT STAKEHOLDERS

List all IT stakeholders and their roles within the project.

- ❑ IT Sponsor
- ❑ Project Manager
- ❑ Architect
- ❑ PMO
- ❑ System Administrator/DBA
- ❑ Application Support

Architecture Approach (mapped to the Vision)

Architecture descriptions, models mapping back to the reference architecture

High-Level Project Plan and Schedule

Post deployment effort required to keep it running and supporting the applications (business logic changes)

4.3 Simplified common vocabulary

Following is a short list of common vocabulary to be used by business, business analysts, architects, project managers, developers, QA, and operations staff.

Term	Definition
Business process modeling	Model, capture, and simulate business process requirements
Business events	An external change that invokes a business process
Business concepts	Enterprise objects, such as customers or products
Business applications	Business applications logic implemented as packaged applications, custom applications, and business processes
Business service	Each discrete business activity within a business application
Logical service model	A one-to-one mapping of physical services to business services to encourage reuse at business service level, instead of at the physical services level
Physical service	The discrete coarse-grained service, categorized by function

4.4 Relevant SOA Standards

4.5 Service Component Architecture and Service Date Object

- Separation of the business logic—embedded in the components that make up the service—from the transport and protocol binding needed to access service components.
- Clean separation of the service components' core business logic from the protocols used by the components to interact with or reference external services.
- Assembly of the services at runtime using dependency injection techniques
- Definition of configuration and deployment metadata for service assembly
- Container guidelines for annotation-driven auto-generated implementation of service implementation layer components
- Runtime binding of clients to the service using metadata entry points and binding of components to external services using metadata external service references
- Metadata configuration policies to define service access and service dependencies, defined by entry points and external service references.
- Trading of document style information using SDO as both parameters and return values
- Metadata that defines service policies (WS-policy for many interoperability features) such as security, transaction, and reliable message policy
- Constant service implementation: the business service offered is fixed while the service delivered can be altered using SCA external references and properties
- Assembly of two different components from the same base implementation by using different property values and by using a different binding to two separate target services (as long as the service interface is the same)
- Configuration of a reference through wiring an implementation to a target service
- Configuration of properties through setting different property values
- Runtime access binding with multiple ways to access the same implementation code base through Web service, JMS messaging, EJB, database stored procedure, and EIS binding
- SCA annotation such as <implementation. java> and <implmenation.ejb> to support different types of implementations using the same service interface
- SCA definition of how the container provider interprets the annotation to provide implementation support for implementations that are not native to the SCA
- SCA to complement BPEL so that BPEL processes coordinate services and service activities while the SCA provides the wiring of services to their implementations
- WS-BPEL implementation through SCA container support of <implementation.bpel annotation
- SCA definition and support for BPEL-style long-running asynchronous transactions that allow non-blocking invocation of services, callbacks, and compensatory transaction support and reliability features
- Support for conversational services and service sequencing to provide business functionality
- Definition of models for connecting geographically distributed services and partner service consumption using external service reference binding at runtime
- Definition of a set hierarchy for assembling complex services that can all be configured, deployed, managed and monitored independently without changing the core implementation base.

4.5.1 SCA Extensions

- Interface-level extension for specification of native interface definitions (Java and WSDL interfaces are base-level interfaces provided by SCA)
- Binding-level extensions for providing binding using SMTP or session initiation protocol (SIP) to access the service components

4.6 Java Business Integration

- Defines the integration environment based on the concept of containers and plug-ins that adhere to industry-standard Service Provider Interface (SPI)
- Defines SPI for integrating both service engines and protocol binding components
- Defines how to build service engines once and bind them to multiple communications protocols using vendors' binding components
- Separates service engines (business capabilities such as process engines and vertical industry transformation packages such as AS/2 or eb-XML or HIPPA transformations) from communications protocols (such as SMTP, WS-I, and JMS)
- Builds binding components by default to read WSDL from services deployed in service engines and help with the exchange of "normalized messages" between service consumers and service providers
- Has services use opaque "normalized message services" to transfer document-based information or payload between the service engines and binding components deployed in a JBI container
- Has protocol-specific binding components interact with a "normalized message router" to communicate with the service engines that have the "rightful service provider" to then transport the message payload
- Enables vendors that specialize in creating industry-vertical business processes to build services using existing service components and keep them in the same JBI container as long as their interface adheres to the service engine SPI

4.7 Traditional Data Movement Technologies

4.7.1 Electronic Data Interchange (EDI)

Electronic data interchange (EDI) is the computer-to-computer exchange of business data. In EDI, information is organized according to a format agreed by both parties, allowing a "hands-off" computer transaction that requires no human intervention or rekeying on either end. All information contained in an EDI transaction set is, for the most part, the same as on a conventionally printed document.

Organizations have adopted EDI for its potential to enhance efficiency and increase profits. Benefits of EDI include:

- Reduced cycle time
- Better inventory management
- Increased productivity
- Reduced costs
- Improved accuracy
- Improved business relationships
- Enhanced customer service
- Increased sales
- Minimized paper use and storage
- Increased cash flow.

The EDI standards are developed and maintained by the Accredited Standards Committee (ASC) X12. The standards are designed to work across industry and company boundaries. Changes and updates to the standards are made by consensus, reflecting the needs of the entire base of standards users, rather than those of a single organization or business sector. Today, more than 300,000 organizations use the 300+ EDI transaction sets to conduct business.

Source: http://www.x12.org

4.7.2 Extract, Transform and Load (ETL)

ETL stands for extract, transform, and load. “Extracting” is moving data from multiple sources. To “transform” it, companies cleanse it and reformat it to the format required by the destination. They then load it into another database, a datamart or a data warehouse for analysis, or on another operational system to support a business process.

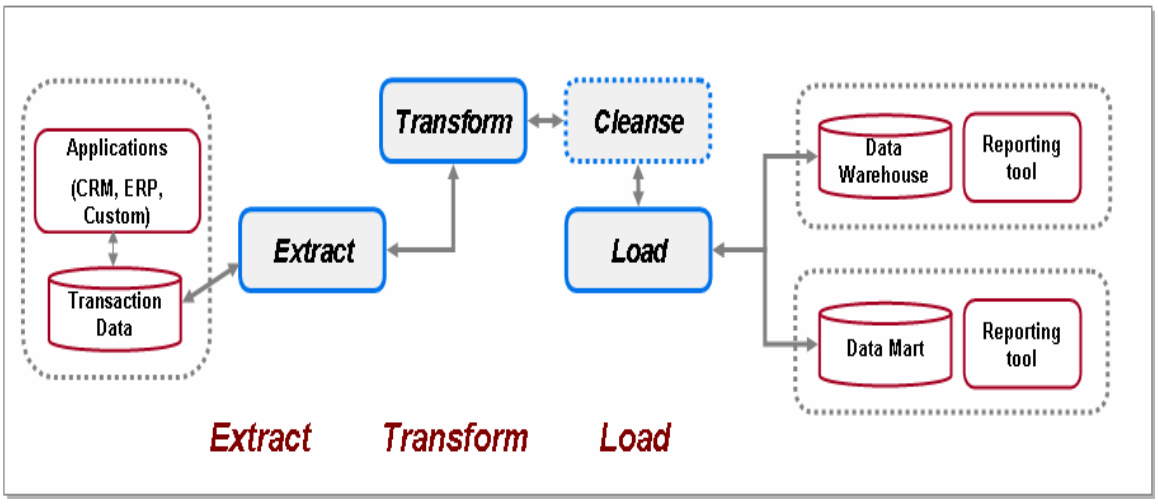


Figure 12: Extract, Transform and Load (ETL)

Even though most ETL scripts are batched, the majority of ETL vendors now provide the capability to invoke these scripts as Web services.

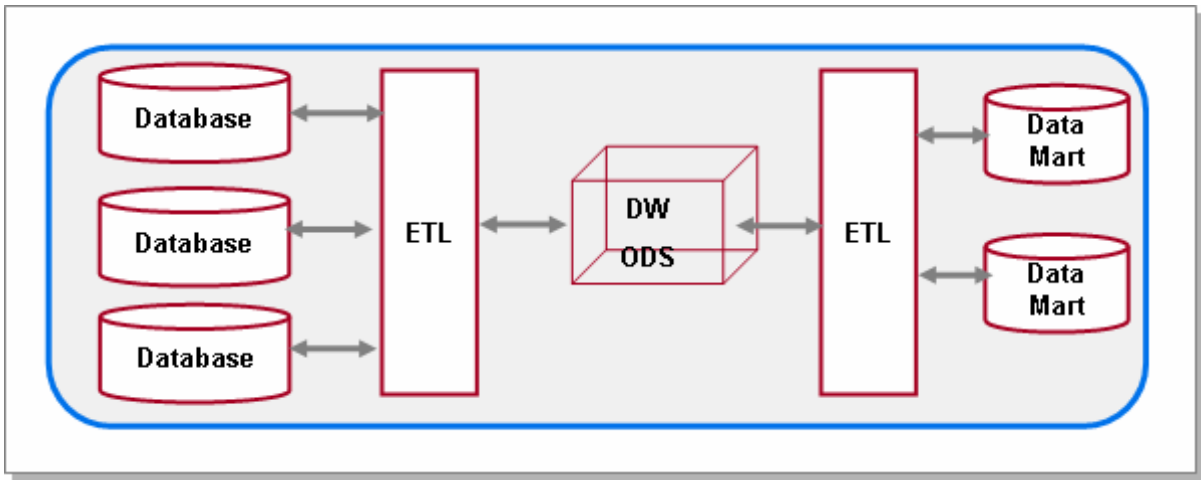


Figure 13: Traditional use of ETL for data warehouse and data marts

The above diagram illustrates the typical use of an ETL system. Data is extracted from multiple sources to create the operations data store (ODS) or data warehouse. Downstream, a datamart might leverage the ODS or data warehouse. This is the recommended approach, but sometimes organizations save time and money by building a datamart without first creating an ODS or data warehouse.

The traditional approach has been to leverage the EAI infrastructure to populate the ODS or data warehouse, but increasingly companies have been using EDA to populate the ODS/ data warehouse. They use ESB and EII tools instead of the traditional EAI and ETL tools. This is driving convergence between ETL and EII tools, pushing EII tools to provide data manipulation and movement capabilities and pushing ETL tools to provide real-time transaction capability.

4.8 Recommended Reading

Following are some links to additional sources of information related to SOA best practices.

4.8.1 Architecture Frameworks

Federal Enterprise Architecture (FEA)
http://www.whitehouse.gov/omb/egov/a-1-fea.html

The Open-Group Architecture Framework (TOGAF)
http://www.opengroup.org/togaf/

The Zackman Institute for Framework Advancement
http://www.zifa.com/

4.8.2 Associated Standards

Services Component Architecture (SCA)
http://www.osoa.org

Web Services Interoperability Organization
http://www.ws-i.org

Organization for the Advancement of Structured Information Standards (OASIS)
http://www.oasis-open.org

Object Management Group (OMG)
http://www.omg.org

4.8.3 Industry Forums

SOA Institute
http://www.soainstitute.org/index.php

Financial Services Technical Conference
http://www.fstc.org

Compliance Oriented Architecture (COA) for SOA
http://www.s-ox.com/Feature/detail.cfm?articleID=1202

4.8.4 Analysts' Web Sites with focus on SOA

ZapThink
http://www.zapthink.com/

CBDI Forum
http://www.cbdiforum.com/

4.8.5 Templates

Capturing Business Scenarios (Use Cases)

http://www.ws-i.org/Requirements/SubmittingScenarios-1.0-2004-09-27.html