NATIONAL INSTITUTES OF HEALTH

enterpriseARCHITECTURE

EATS

NIH Integration Technology Architecture

June 13, 2006

Today's Objectives

- Describe NIH's Integration Technology Architecture and key outcomes from the recent Integration Technology Domain Team effort.
- Discuss how integration technology patterns can be applied to software development efforts.
- Provide an overview of ongoing integration efforts and how patterns are currently being applied at NIH.

Integration Technology Overview

The integration technology domain is composed of those technologies that enable communication between systems. This includes communications middleware, data integration tools, web services and other software that facilitate system-to-system integration.

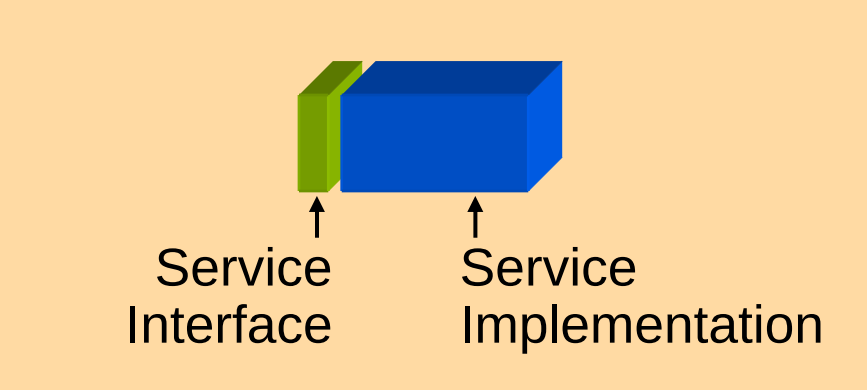
Benefits

- Increases the productivity of software development efforts reducing cost.
- Enables reuse of existing solutions leading to better leverage investments.
- Increases the reliability of integration points.
- Increases flexibility of systems.

Key Concepts: Service-Oriented Architecture

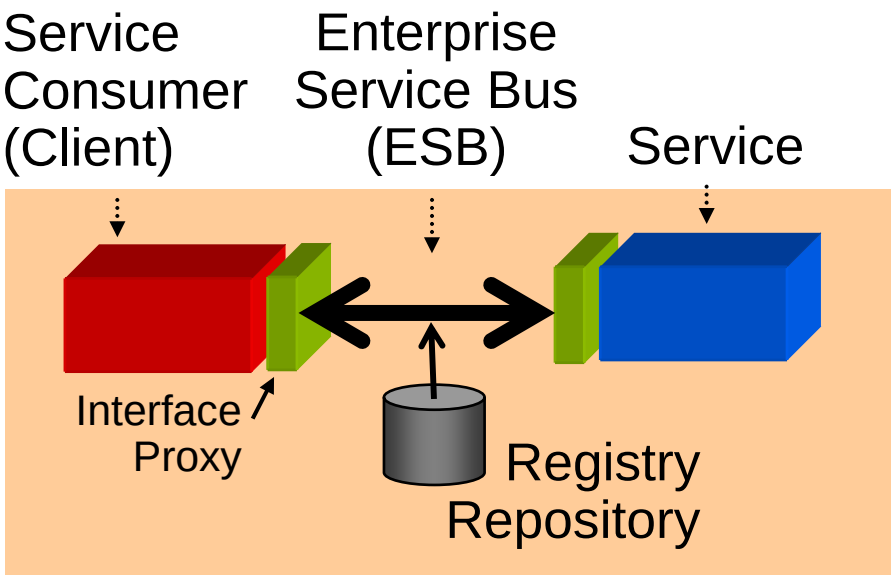
Service in a SOA

Business software component, designed to be invokable by name from external contexts via a well-defined, programmatic interface.



SOA

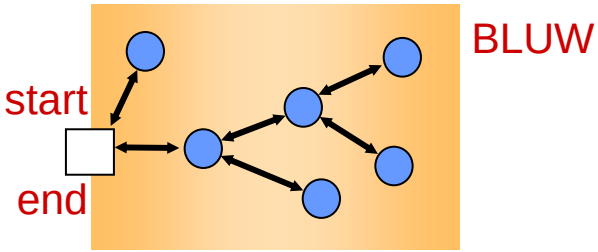
Software design pattern consisting of any number of registered services and service consumers (clients) in loosely coupled relationships.



Source: Gartner

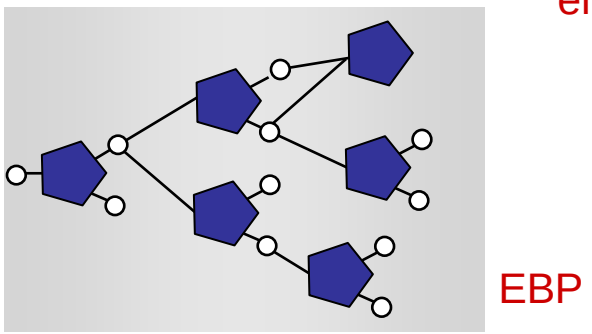
Key Concepts: Event-Driven Architecture

SOA Interaction



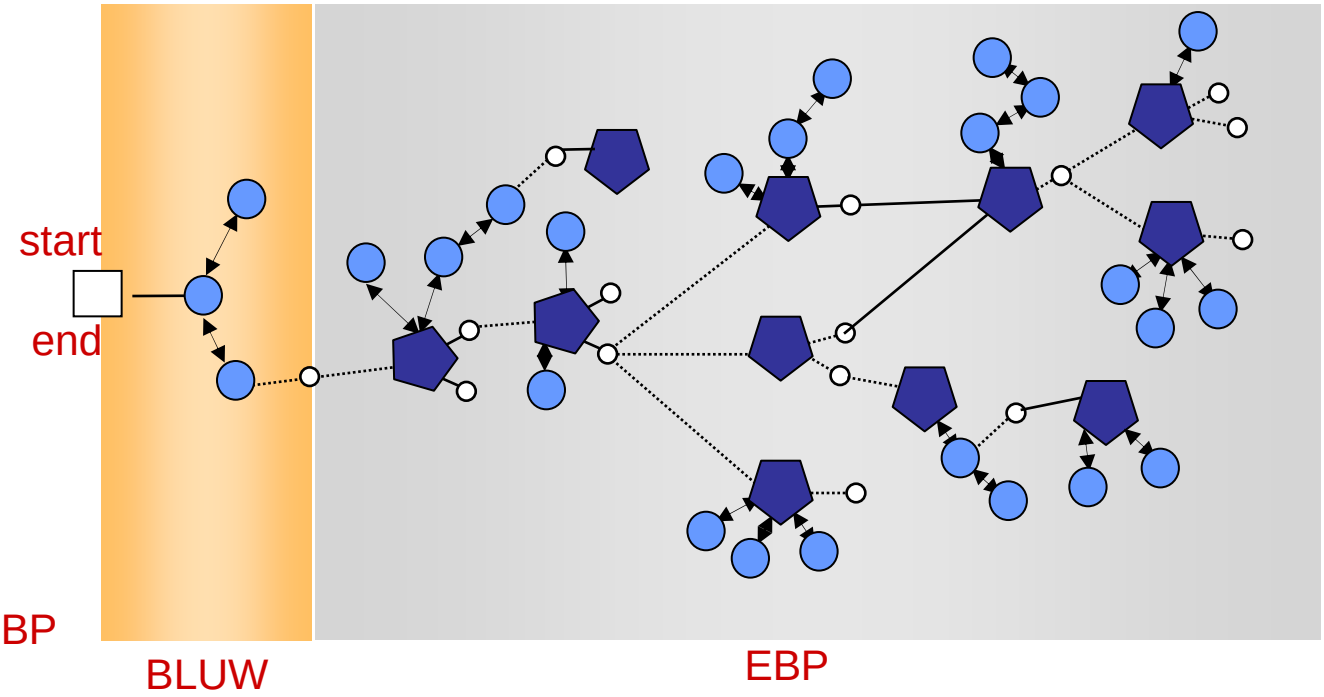
- Coupled
- Synchronous
- Subordinate
- Closed-ended

EDA Notification



- Decoupled
- Notification/subscription
- Autonomous
- Open-ended
- Asynchronous

Business Component Architecture
(Service-Oriented and Event-Driven)



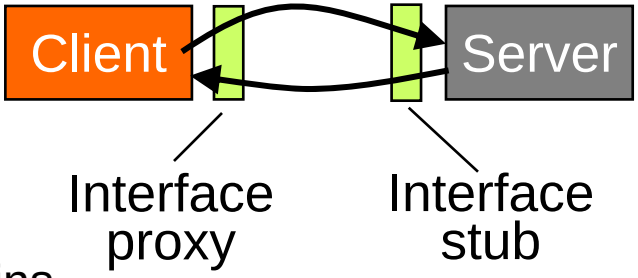
BLUW business logical unit of work
EBP enterprise business process

Source: Gartner

Two Ways for Business Components to Relate: Service-Oriented or Event-Driven

Service-Oriented Architecture *Interaction*

- Uses interface metadata
- One-to-one connections
- Client directs flow
- Data flows are predictable and linear
- Closed to unforeseen input once process begins



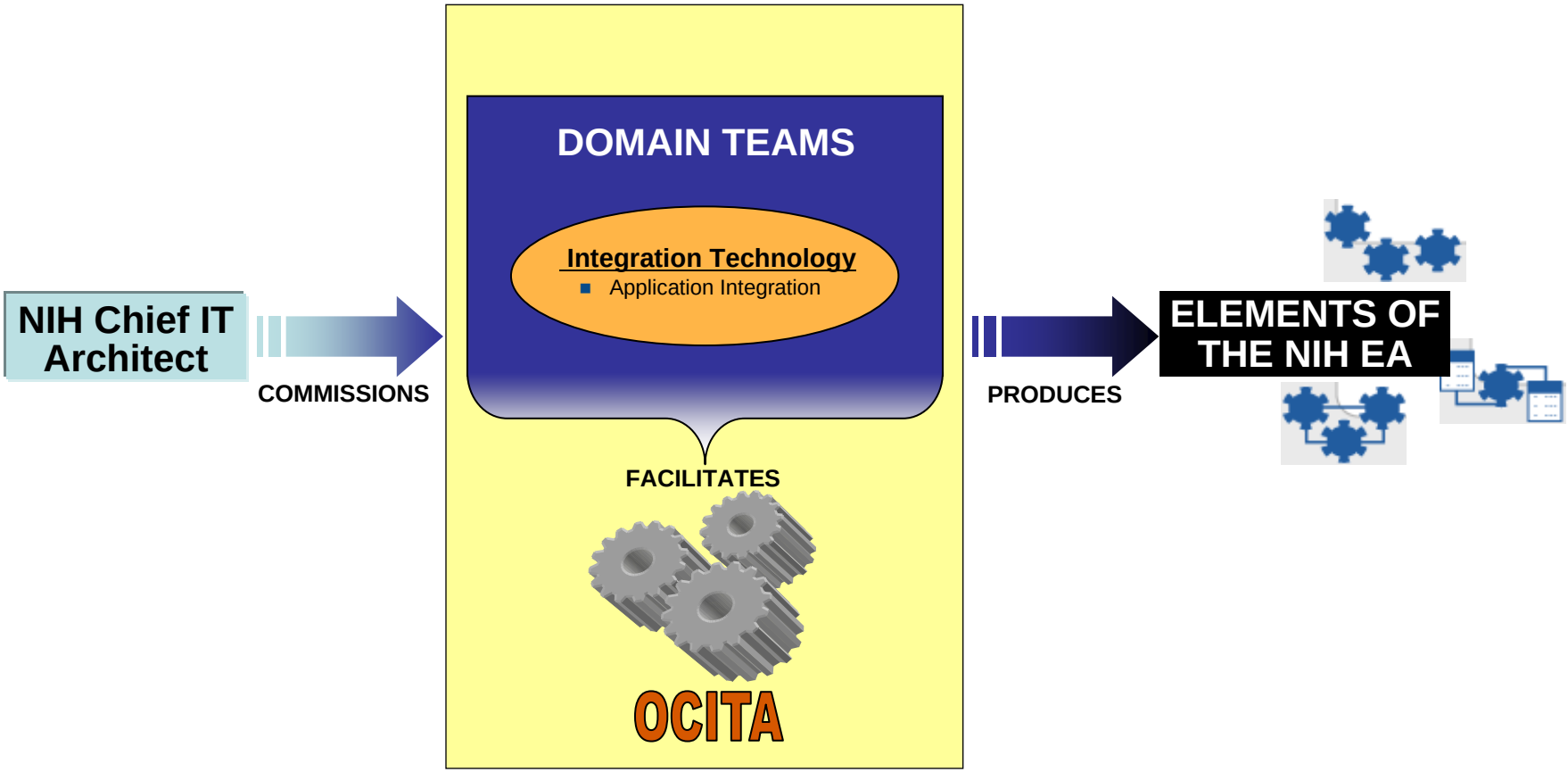
Event-Driven *Notification*

- Uses event descriptor metadata
- Many-to-many connections
- Sink (recipient) determines flow of logic
- Dynamic, parallel, asynchronous flows
- Can react to new external input while process is in flight

Source: Gartner

Why Do We Assemble Technical "Domain Teams"?

To participate in designing the NIH Enterprise Architecture (EA)



Domain Team Management Approach

The DT management approach was designed to ensure:

1) Consensus is developed across a broad base of ICs

2) The best use of your time is leveraged

NIH Domain Team Members

- The DT structure allows a community of NIH specialists to focus on their areas of expertise.
- Leverages their understanding of what makes sense for NIH to revise coordinator-developed straw-models.
- All materials, presentations, meeting minutes, and meeting attendance records will be posted on the EA community of the NIH portal (Project = *Integration Technology Domain Team*)

Coordinators

- Perform "heavy lifting"
- Drive towards closure
- Document DT final recommendations

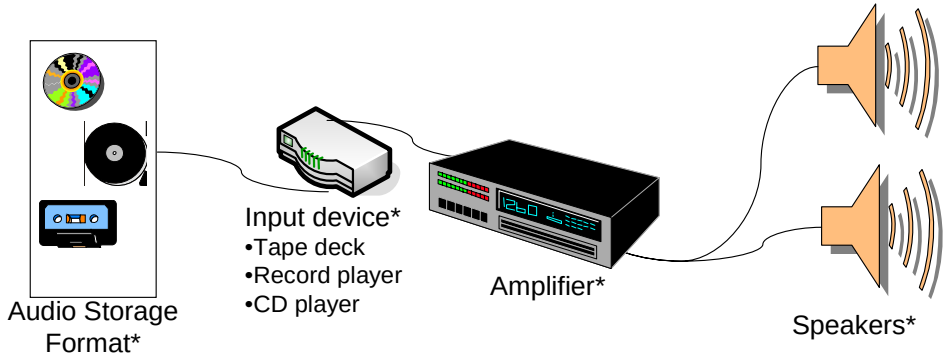
Technology Architecture Artifacts

- Patterns
  - Logical models of technology — design ideas that can be reused and leveraged across the enterprise.
  - Blueprints that identify components at a design or logical level (for example, a data server or an application server), and show the roles, interactions, and relationships of components at that level.
- Bricks
  - Specify NIH adopted technical standards and protocols or technologies and products.
  - Define current and future standards.
  - Define products or standards in the current environment that are to be retired or contained.

Sample Pattern & Brick

Simple Home Sound System Pattern and Audio Storage Brick

Pattern



* Technical bricks would exist for each of these technical elements to ensure system operability when modules are developed and integrated independently.

Brick

(Audio storage format)

Current	Two Years	Five Years
Baseline	**Tactical Deployment**	**Strategic Direction**
■ Cassette ■ CD ■ MP3 ■ Radio ■ Vinyl records (LPs, 45s) ■ 8-track ■ Reel-to-reel	■ CD ■ MP3 ■ Radio ■ Cassette (for books on tape)	■ MP3
Retirement Targets	**Containment Targets**	**Emerging Technologies**
■ Reel-to-reel	■ Vinyl records (LPs, 45s) ■ Cassette (for music) ■ Reel-to-reel (for recording production) ■ 8-track	■ Digital rights management ■ Digital asset management

Comments

- Note: eight-track tapes are containment for recording satellite data
- Industry moving from analog to digital media

Accomplishments

- Revised existing patterns
  - Service-Oriented Architecture (SOA)
  - Data Consistency
  - Composite Application
  - ***Two Patterns Removed –***
    - ***Multi-step Process***
    - ***Basic Communications Pattern***
- Developed several new patterns
  - Enterprise Service Bus (ESB)
  - Service Implementation
  - Event-Driven Architecture
  - SOA Security
- Updated bricks
  - Updated baseline technology
  - Issued Requests for Information (RFIs) for data management middleware and communications middleware
- Developed guidance to address strategic integration technology issues
  - Emergence of Service Component Architecture (SCA)
  - Convergence of workflow and business process management (BPM) technologies

Key Integration Pattern Decisions

- In general, a logical “hub-and-spoke” topology for integrations is preferred because it provides for more consistent integration approaches, facilitates reuse, and increases reliability.
- Service-Oriented Architecture (SOA) within NIH will be based on open web-services standards.
- An enterprise service bus (ESB) will be used to provide basic communications.
- Event-Driven Architecture (EDA) will be used to support asynchronous, decoupled interactions between systems.

Key Integration Brick Decisions

- TIBCO has been selected as NIH's strategic integration broker suite (IBS) acting as a hub for system-to-system integrations across NIH.
- Informatica has been identified as NIH's strategic extract, transform, and load (ETL) tool for data integration.
- Core Web Services standards such as UDDI, SOAP, and WSDL have been adopted for service implementation across NIH.

Categorizing Integration Technology Patterns

Basic Building Blocks

- Service-Oriented Architecture
- Web Services
- Service Implementation
- Event Driven Architecture
- Data Consistency
- Service-Oriented Architecture Security

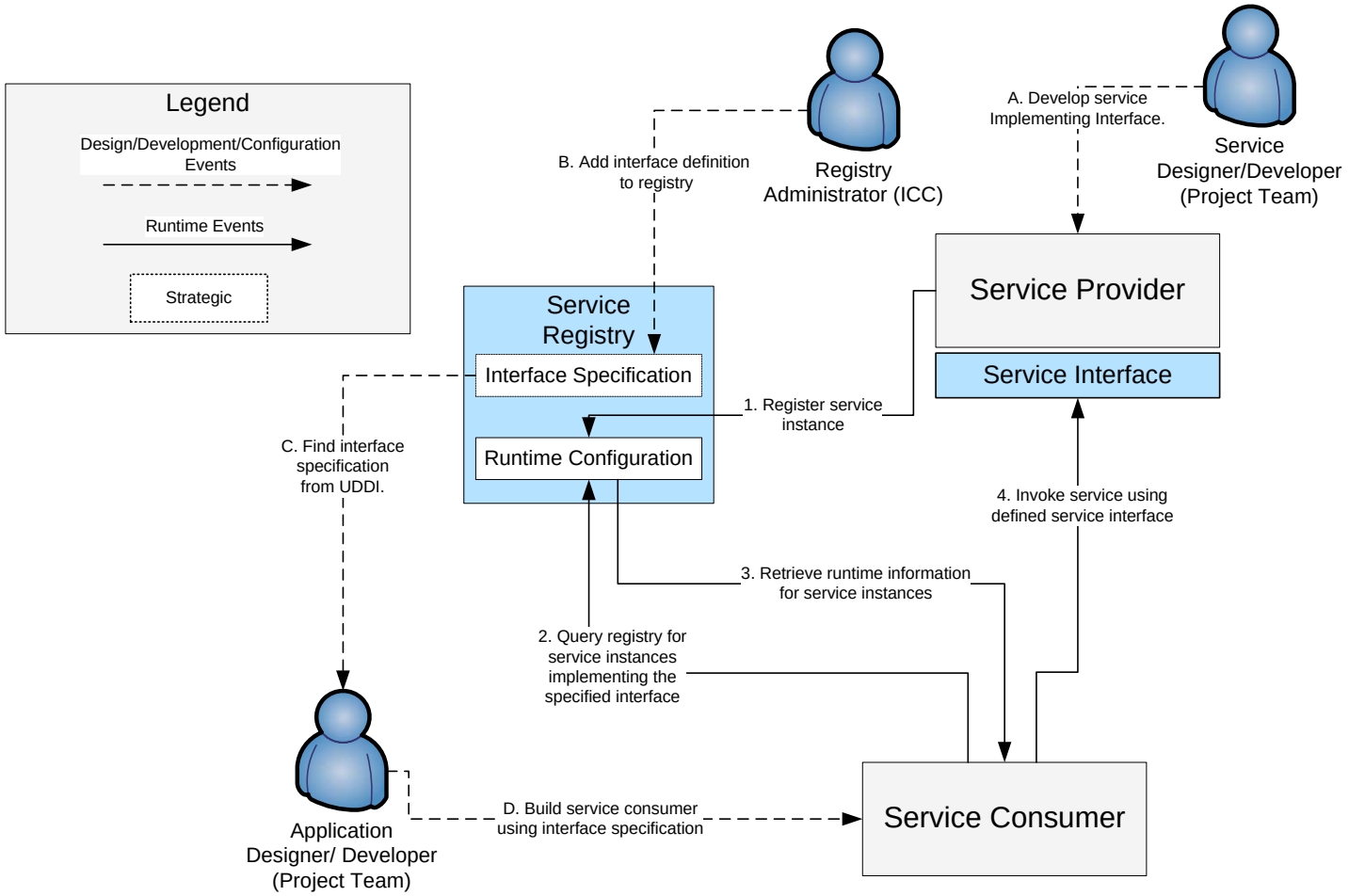
Form the basis for interactions between systems at a granular level and can be composed into more complex patterns.

Complex Patterns

- Composite Application Pattern
- Enterprise Service Bus
- Large-Scale Integration Pattern
- Broker/Operational Data Store/Warehouse Pattern

Provide comprehensive views of overall architectures for integrations addressing different solution requirements.

SOA Pattern


Legend
Design/Development/Configuration Events
Runtime Events
Strategic
Registry Administrator (ICC)
B. Add interface definition to registry
A. Develop service Implementing Interface.
Service Designer/Developer (Project Team)
Service Registry
Interface Specification
Runtime Configuration
Service Provider
Service Interface
1. Register service instance
C. Find interface specification from UDDI.
4. Invoke service using defined service interface
3. Retrieve runtime information for service instances
2. Query registry for service instances implementing the specified interface
Application Designer/ Developer (Project Team)
D. Build service consumer using interface specification
Service Consumer

SOA Pattern Strawmodel

Description

- Pattern consists of three core elements – service provider, service consumer and service registry
- The service registry describes the services available in its domain; the service provider will register it service in the registry
- The service provider is the function that performs a service in response to a request from a consumer
- The service consumer is the function that consumes the result of a service supplied by a provider
- A service interface exist which defines the programmatic access "contract" of the service; it also establishes the identity of the service and the rules of the service invocation
- Relationship between a service provider and consumer is dynamic; established at runtime by a binding mechanism
- This pattern works by the service consumer invoking services stored in the registry and leveraging the interface to use the business logic of the service provider

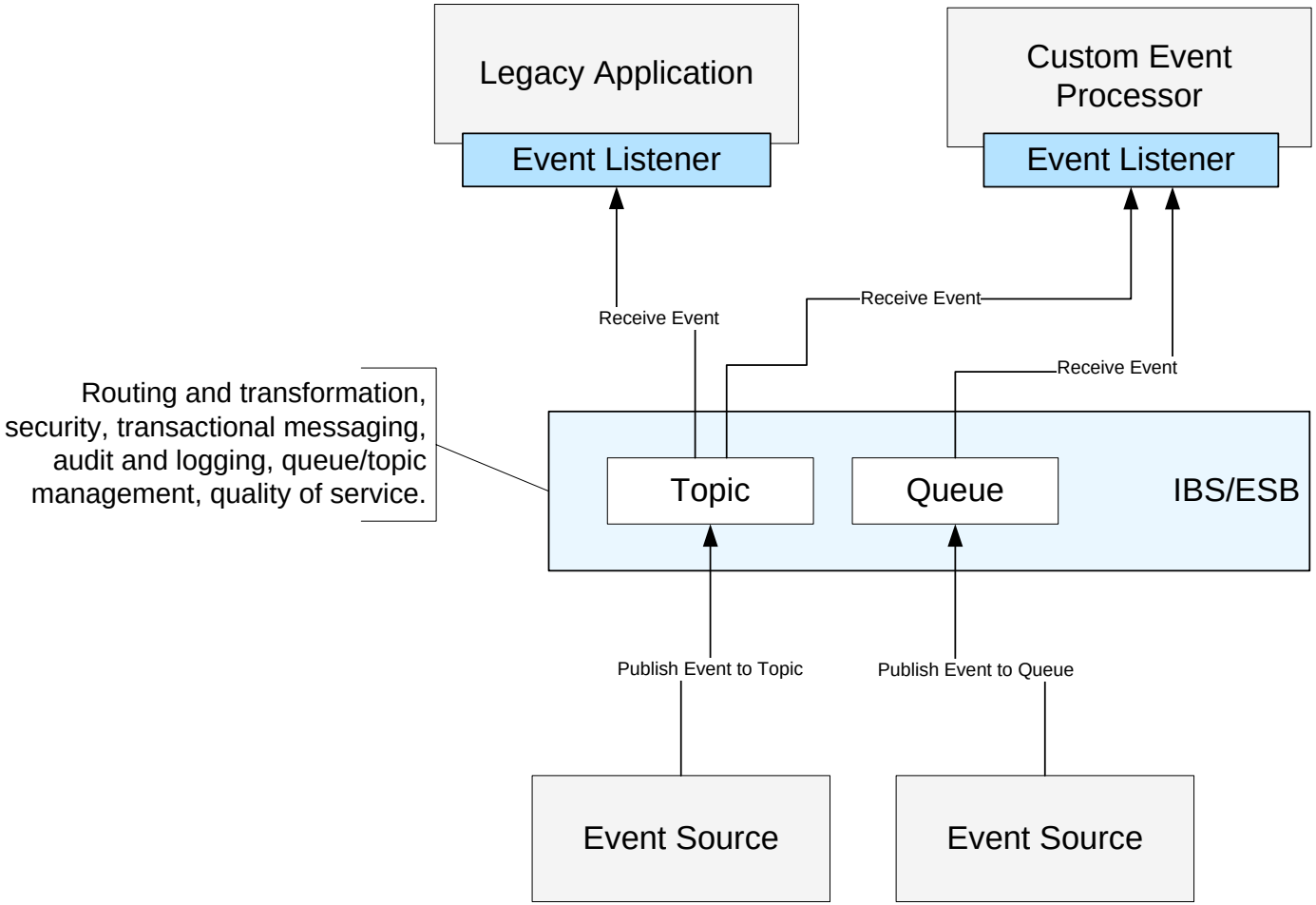
Benefits

- Reuse of services
- Incremental deployment and maintenance
- Architectural partitioning
- Flexibility and agility

SOA Pattern Recommended Usage

- Strongly preferred in cases where there is a need for request-reply, real time integration between systems.
- Strongly preferred when more than two systems are involved in the integration.
- Service being provided is a likely candidate for reuse.
- Service implementation requires no advanced knowledge of the service client.

Event-Driven Architecture Pattern



Event-Driven Architecture Pattern

Description

- EDA uses unidirectional messaging to communicate among two or more, largely independent peer procedures.
- The communication is initiated by an "event".
  - This trigger typically corresponds to some business occurrence.
  - Any subscribers to that event are then notified and thus activated.
- An event is the automated record of what happened.
- Pattern works as follows: An event source publishes an event to a topic or queue. This in turn causes an event listener to be activated by the messaging infrastructure.

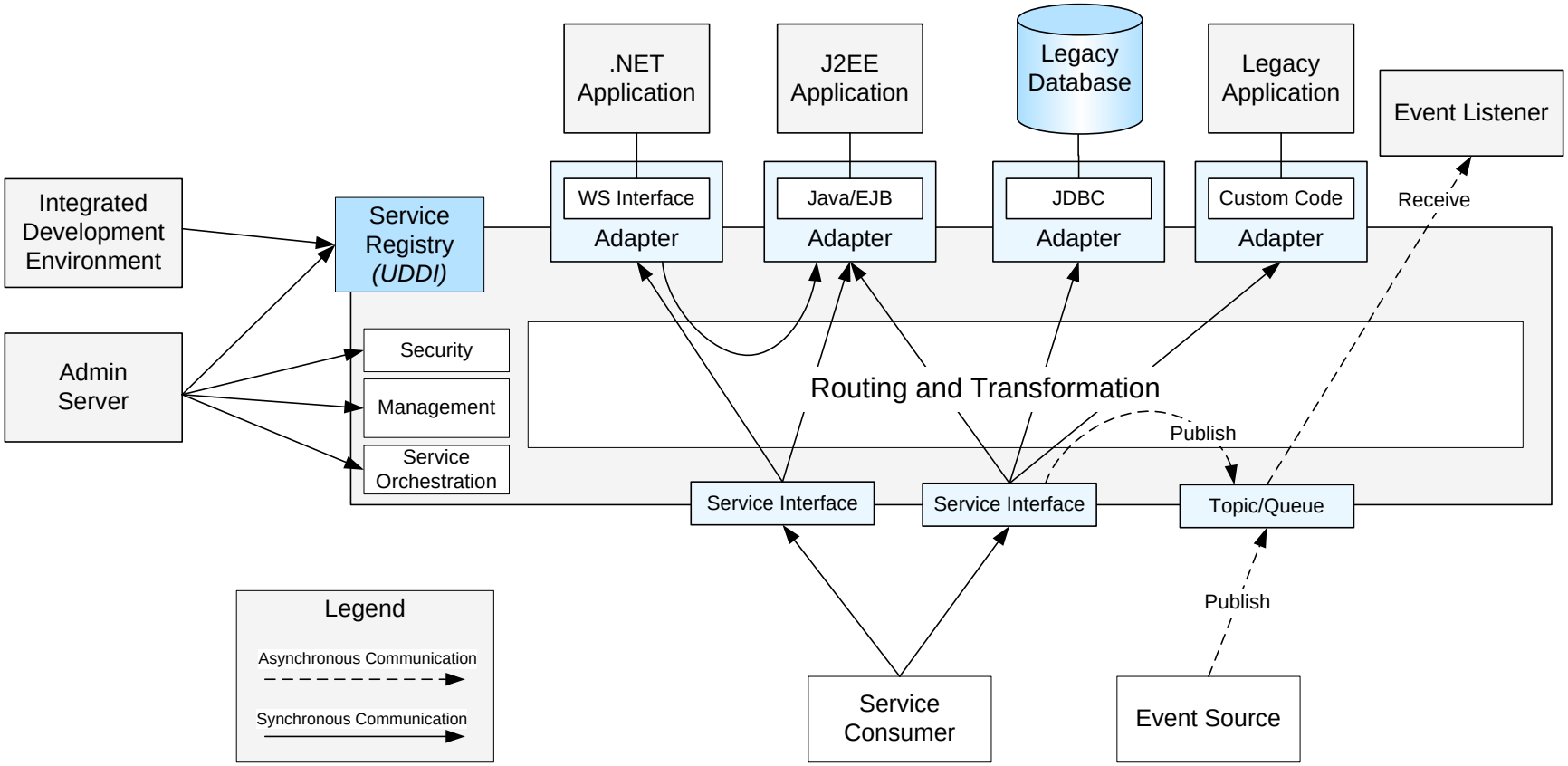
Benefits

- Reduces the elapsed time of a business process by working with individual events rather than batches and executing activities concurrently.
- Decoupled interactions - Event publishers are not aware of the existence of event subscribers. Subscribers do not need to be available for sending system to continue processing.
- Can be achieved less invasively than request-reply service interactions in many cases.
- Many-to-many communications - Publish/Subscribe messaging where one specific event can impact many subscribers
- Asynchronous - Supports asynchronous operations through event messaging.
- Supports asynchronous acknowledgement when required to complete a process step.

Event-Driven Architecture Recommended Usage

- The business process is inherently asynchronous.
- Events being published are of interest to multiple other applications.
- Real-time request/reply processing is not required.
  - No immediate acknowledgement is required by the event source.

Enterprise Service Bus Pattern



Enterprise Service Bus Pattern

Description

- The ESB at NIH is TIBCO BusinessWorks.
- The ESB provides a number of higher level services that facilitate service reuse and event-driven architecture.
  - Provides both message-based and request-reply communications
  - Rule-based routing of messages
  - Security
  - Management and monitoring
  - Process orchestration
  - Message transformation
- The ESB provides adapters for a variety of technology platforms including:
  - Java applications
  - Generic web services
  - Generic databases
  - Packaged applications
  - HTTP (Web “screen scraping”)
  - LDAP
  - Unix command capability (e.g., invoke a shell script).
- Allows for the real-time aggregation of data from multiple systems.

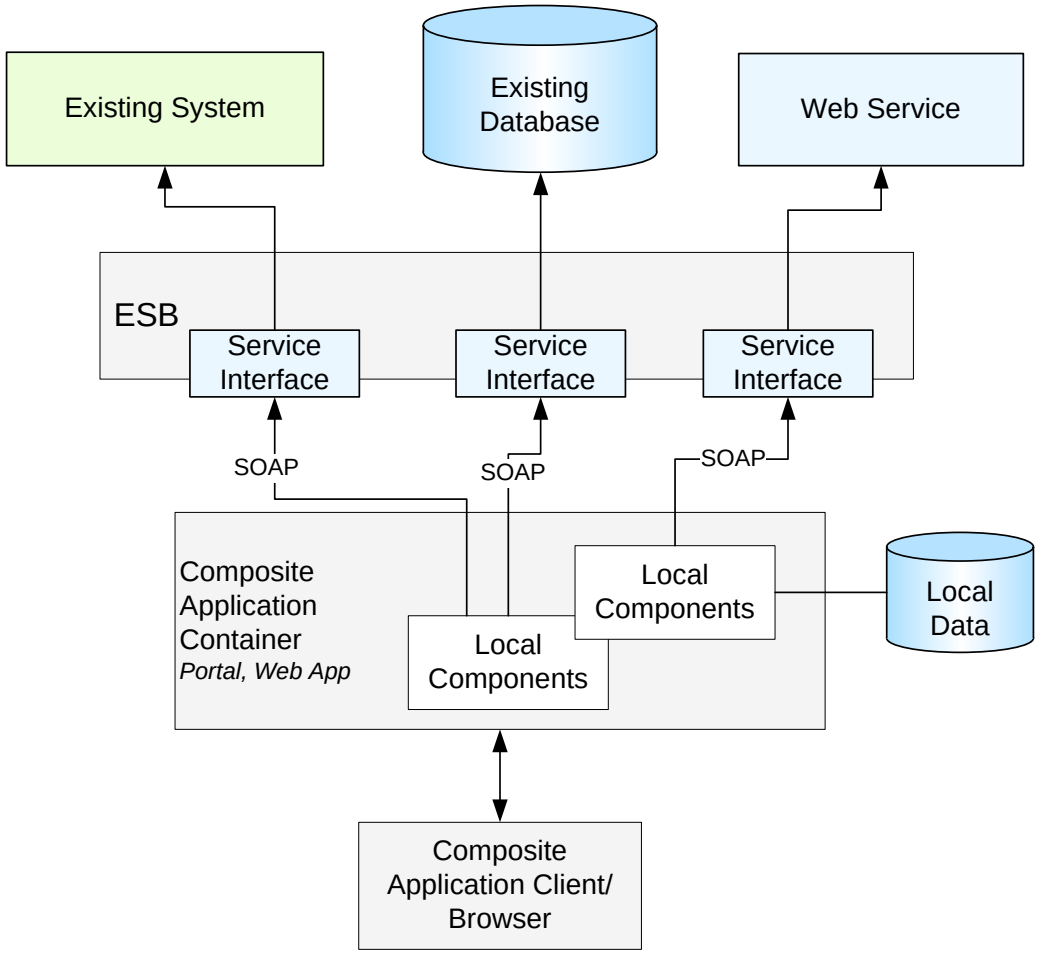
Benefits

- Enables a more manageable hub and spoke topology for services allowing for a consistent approaches to security, centralized configuration management, and management of quality of service.
- Accelerates implementation of services by providing interface mapping, adapters to existing systems, and the ability to compose existing services into more complex new services.
- Provides the framework for service reuse.

Enterprise Service Bus Recommended Usage

- Integration across heterogeneous systems and databases is required.
- Services are widely shared across the enterprise.
- Central management of security is required.
- Complex message routing or data transformation is required.
- Service orchestration is being used to define more complex composite services.
- Both SOA (request-reply) and EDA (message-oriented) interactions are needed to support a business process.

Composite Application Pattern



Updated Composite Application Pattern

Description

- A composite application seems to the user like a new application. It has some new functionality itself, plus functionality from multiple existing applications.
- A composite (or “virtual”) application needs to be near zero in latency (usually under a minute, often within a couple of seconds). It may be implemented with any of a wide range of middleware technologies.
- A variety of methods may be used to create composite applications. For NIH, the standard approach to composite applications will be through the use of web services and leveraging the ESB.
- The figure shows a direct interaction between the composite application and the services leveraged. The ESB’s service orchestration capabilities may also be leveraged to provide more complex flow between services.

Benefits

- NIH can use the composite application pattern to “glue together” some of its partially redundant applications and give users a much more convenient interface.
- Extension systems can use this pattern to construct an interface that uses and updates both enterprise and IC-specific data at the same time.

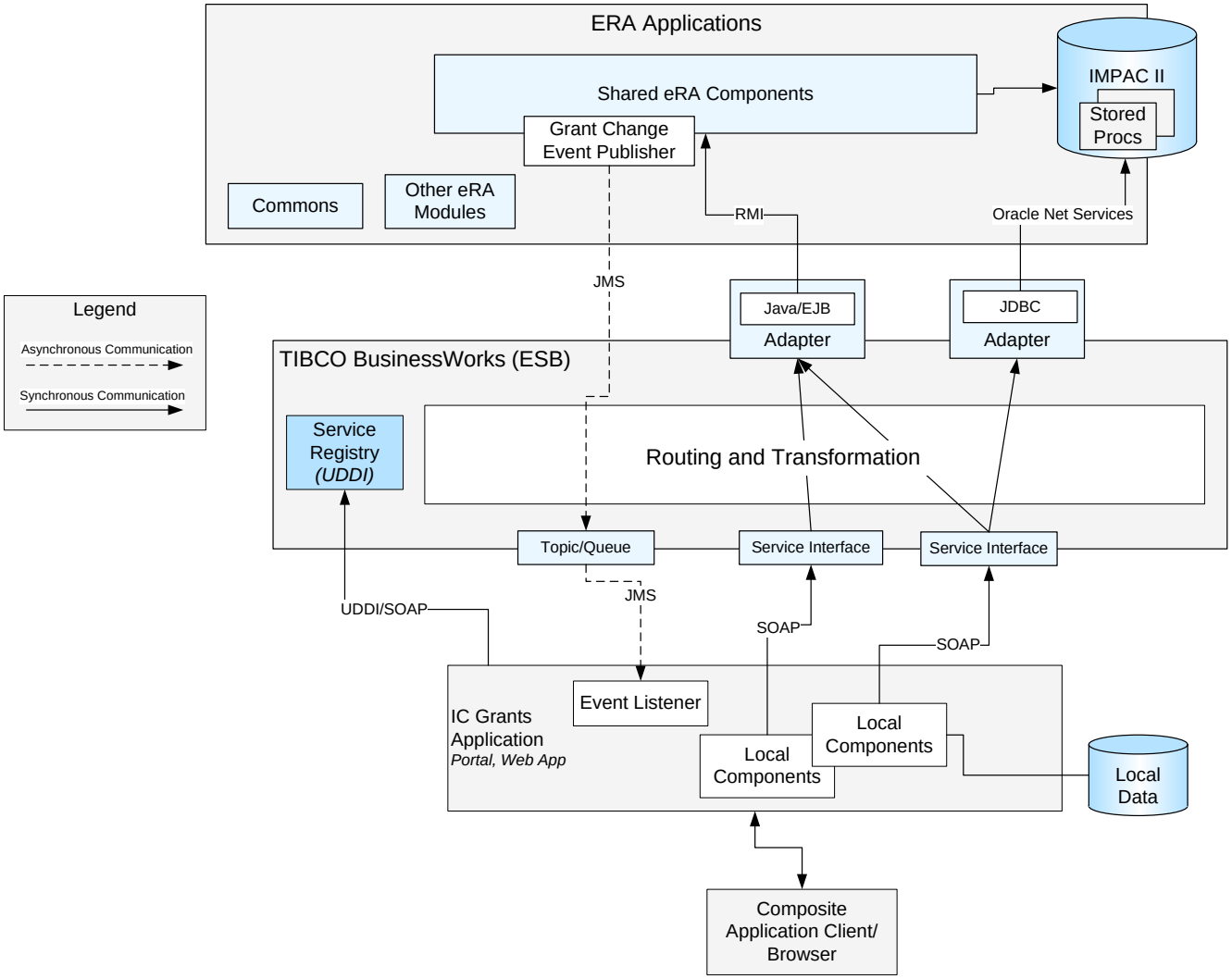
Composite Application Recommended Usage

- An end-user application is required to leverage the functionality of an existing system or service using a request-reply interaction.

Applying Patterns – Example Scenario

- Description
  - An extension system supporting grants management within an IC must exchange data and share processing of grants with the eRA system.
- Key Requirements
  - The application must read and update data managed locally within the IC and centrally within the eRA system.
  - Grants managers need to be notified when there are changes to funded grants.
  - Existing eRA integration points with defined interfaces must be leveraged.

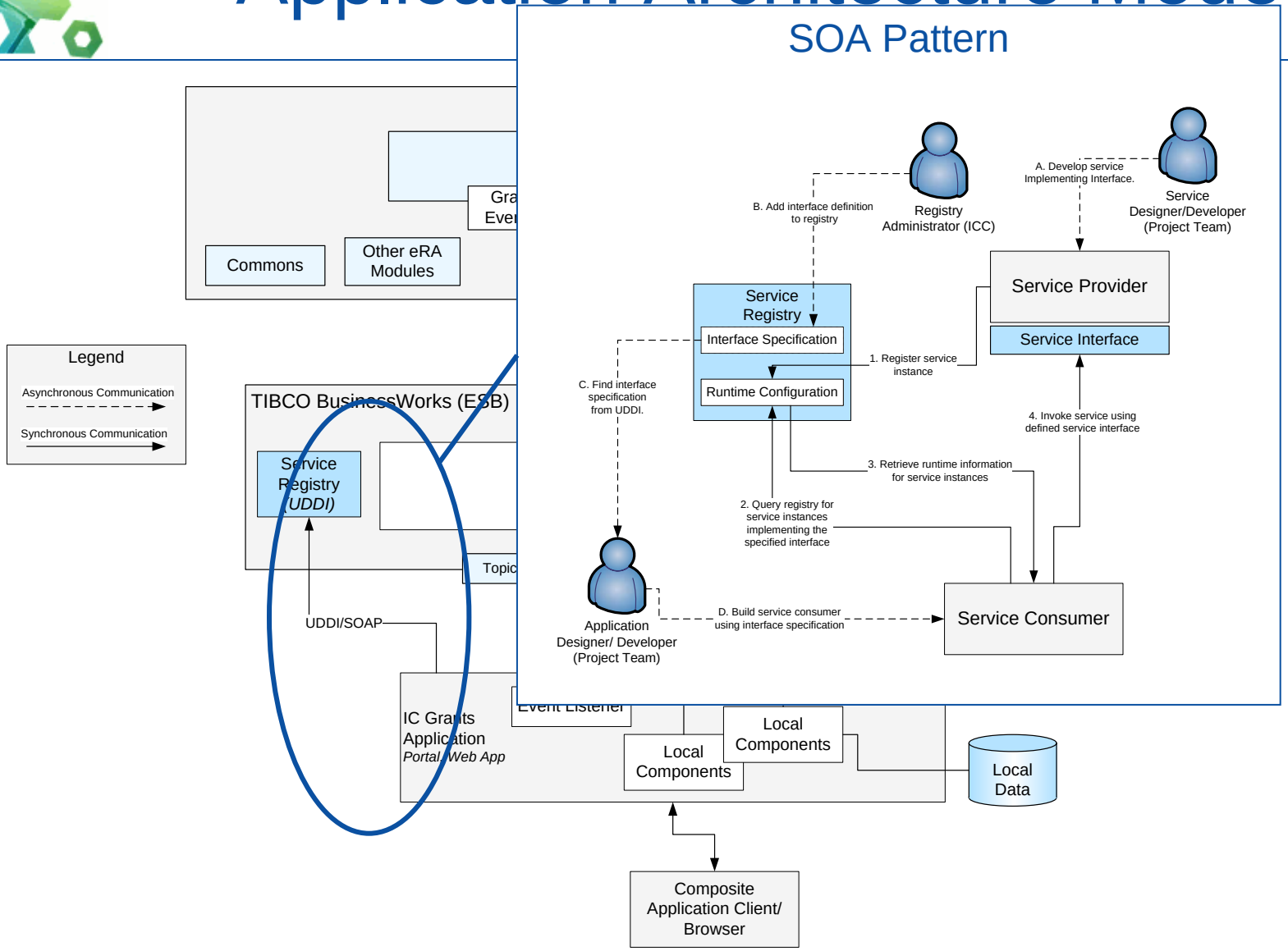
Solution Architecture Model



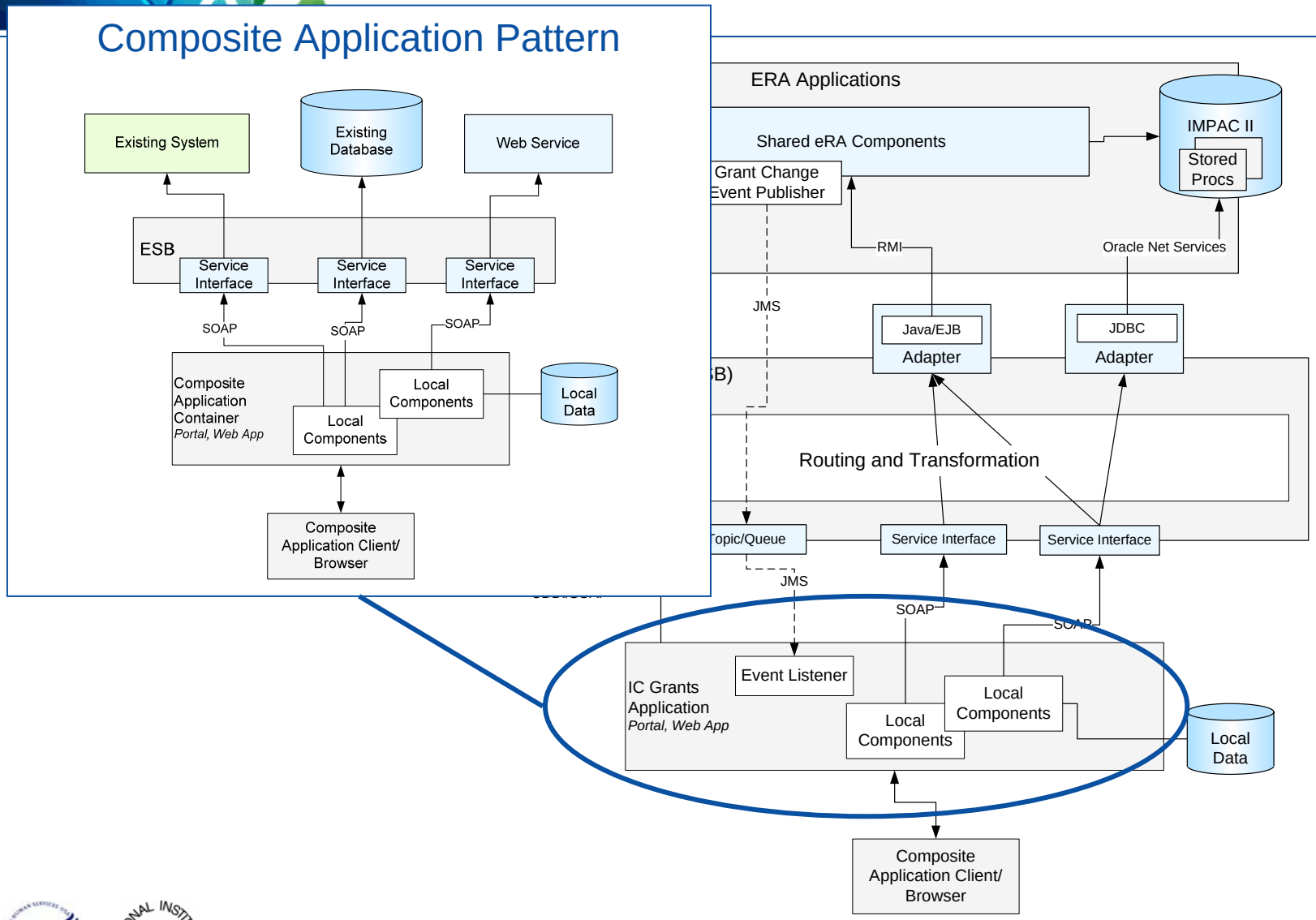
Solution Description

- The core services within the eRA system that need to be used have been built both as J2EE components and stored procedures. Both JDBC database adapters and J2EE adapters will be required.
- Additionally views within IMPAC II are made accessible via services to facilitate the development of composite services that consolidate information from multiple sources.
- The ESB is used as the central hub for integrations managing the services and providing for transformation, routing and service orchestration as needed.
- The need for notifications of changes to grant status handled through an event-driven approach with changes being published to a topic which can then be accessed by a subscribing event listener within the IC grants system.
- The IC grants application becomes a composite application delivering functionality based on local components as well as integrate remote services provided through the integration infrastructure.

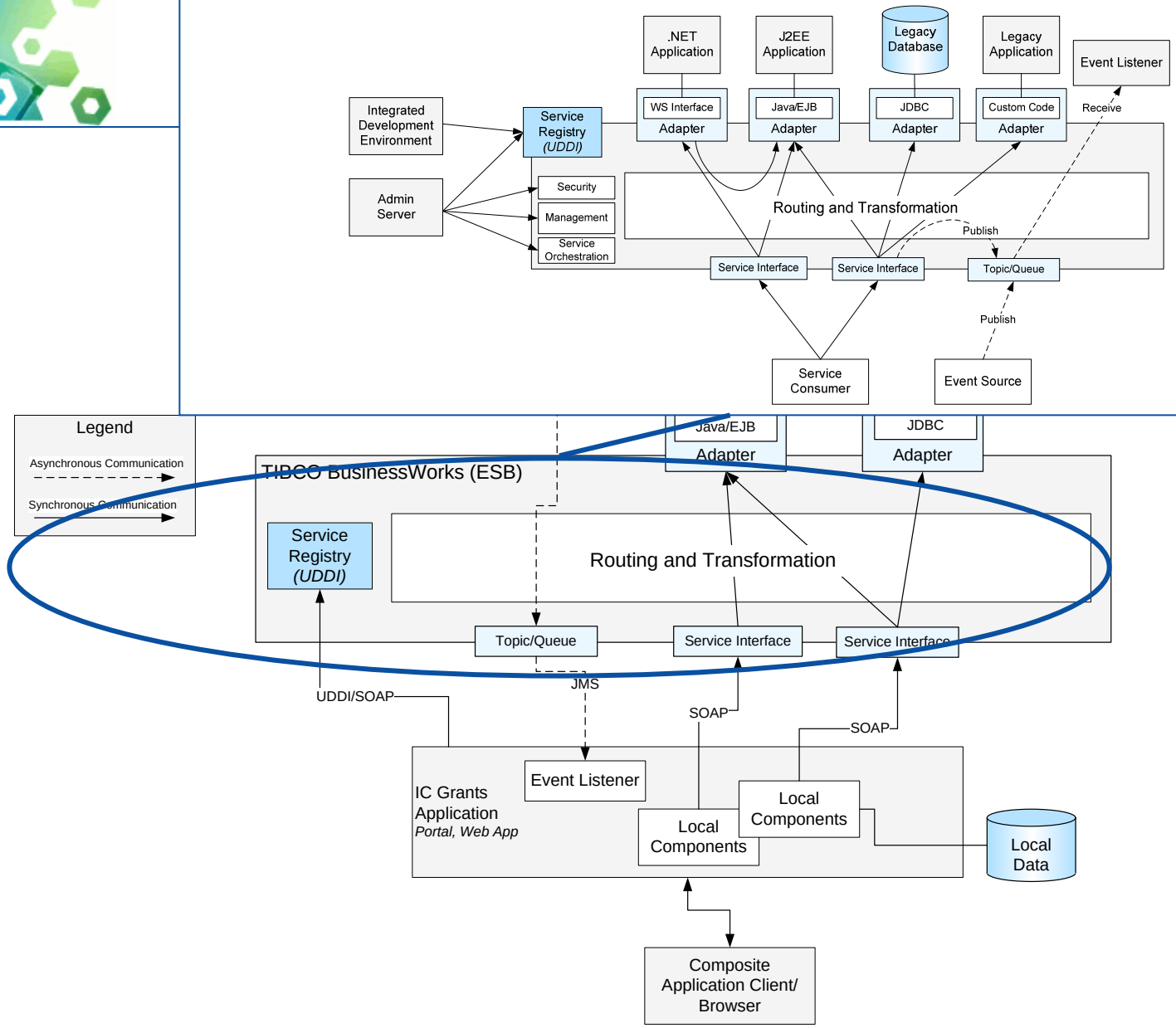
Application Architecture Model



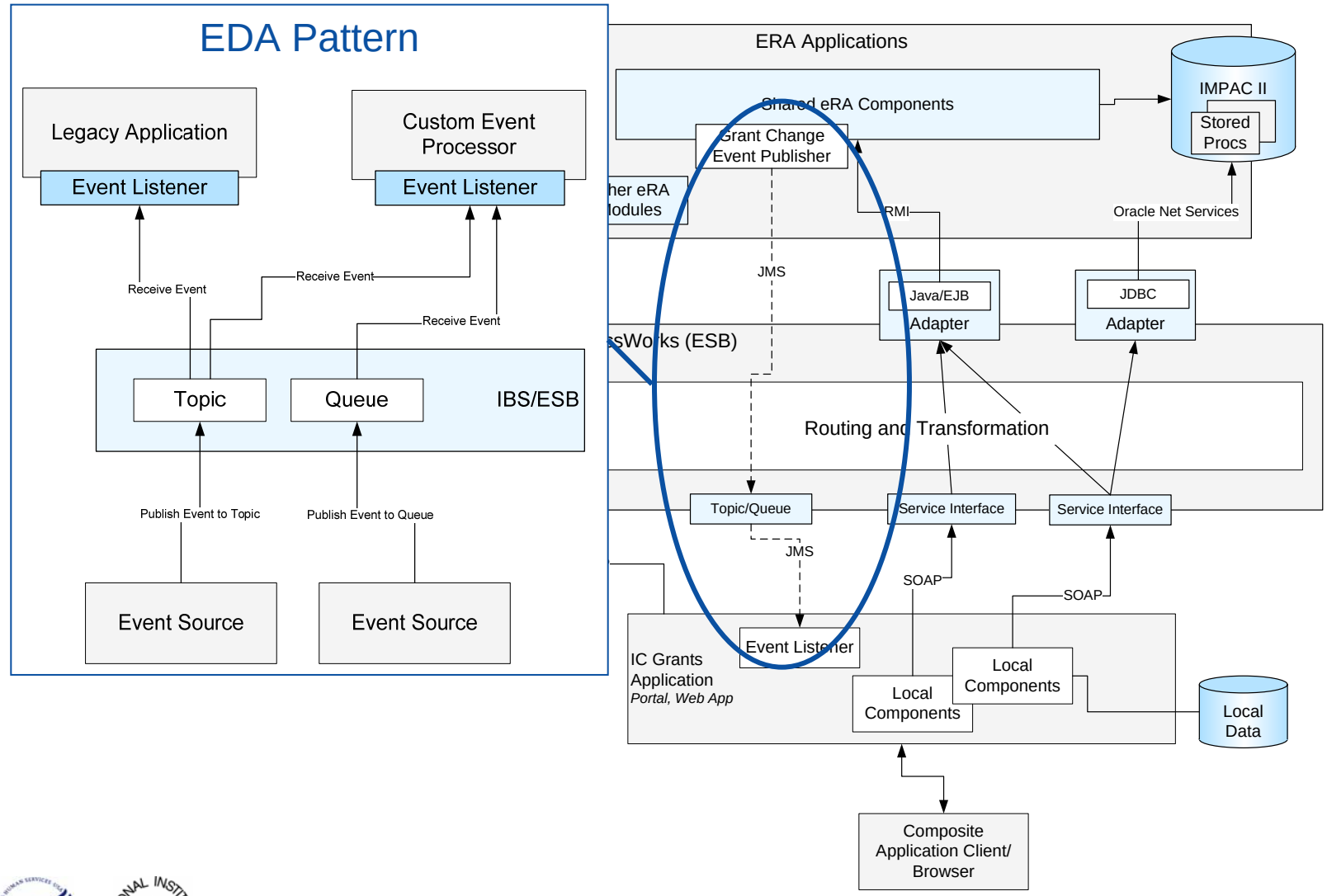
Application Architecture Model



ESB Pattern


.NET Application
J2EE Application
Legacy Database
Legacy Application
Event Listener
Integrated Development Environment
Service Registry (UDDI)
WS Interface
Adapter
Java/EJB
Adapter
JDBC
Adapter
Custom Code
Adapter
Receive
Admin Server
Security
Management
Service Orchestration
Routing and Transformation
Publish
Service Interface
Service Interface
Topic/Queue
Publish
Service Consumer
Event Source
Legend
Asynchronous Communication
Synchronous Communication
Java/EJB
Adapter
JDBC
Adapter
TIBCO BusinessWorks (ESB)
Service Registry (UDDI)
Routing and Transformation
Topic/Queue
Service Interface
Service Interface
UDDI/SOAP
JMS
SOAP
SOAP
IC Grants Application
Portal, Web App
Event Listener
Local Components
Local Components
Local Data
Composite Application Client/ Browser

Application Architecture Model



Examples of Patterns Being Used in Current Development Efforts

- *OFM/eRA Batch Process* - Data Consistency Pattern being implemented to support synchronization of data between systems.
- *NEES Data Access* – The Composite Application Pattern is being used to support integrated access to data from multiple databases in real-time.
- *NED Data Access* – Access to NED information is being provided via a web service.

All of these implementations leverage the ESB pattern!

Ongoing Efforts

- Implementation of central TIBCO infrastructure.
- Implementation NIH Integration Competency Center (ICC) as a central resource for integration knowledge sharing and enabling infrastructure.
- Selection and implementation of service registry/repository and lifecycle management.
- Development of additional NRFCs to provide more detailed guidance in the areas of service implementation and integration security.

Where to Find More Information

- Principles
  - Enterprise Principles
    - http://enterprisearchitecture.nih.gov/ArchLib/Guide/EnterprisePrinciples.htm
  - Application Principles
    - http://enterprisearchitecture.nih.gov/ArchLib/AT/IA/Applications/ApplicationsTechnologyPrinciples.htm
  - Integration Principles
    - http://enterprisearchitecture.nih.gov/ArchLib/AT/IA/Integration/IntegrationPrinciples.htm
- Bricks and Patterns
  - Application Technology Bricks
    - http://enterprisearchitecture.nih.gov/ArchLib/AT/TA/TAListing.htm?CatName=Applications%20Technology&SectionName=Technology
  - Integration Technology Bricks and Patterns
    - http://enterprisearchitecture.nih.gov/ArchLib/AT/TA/TAListing.htm?CatName=Integration%20Technology&SectionName=Technology
- NIH Standards Development Process
  - http://enterprisearchitecture.nih.gov/About/Approach/StandardsDevelopmentProcess.htm

Contact Information

- Web site

  http://EnterpriseArchitecture.nih.gov

  Subscribe to updates - http://enterprisearchitecture.nih.gov/Tools/EmailUpdates/

- The NIH Enterprise Architecture Community in the NIH Portal
- Email

  EnterpriseArchitecture@mail.nih.gov

- Subscribe to the EA LISTSERV:

  http://list.nih.gov/archives/enterprise_architecture.html