Creating a Systems Architecture for an SOA-based IT System as Part of a Systems Engineering Process

Robert S. Ellinger, Ph.D.
Enterprise Architect

Gabriel Hoffman
Senior System Engineer

October 2008
Agenda

• Service Oriented Architecture?
• Service Oriented Architecture and Investment-Driven SOA (ID-SOA™)
• Creating a System Architecture for an SOA-based IT System
Services-Oriented Architecture?
The Problem

• IT organizations often focus on implementing technology yet not enough on helping a customer organization accomplish its **mission**

• Today's IT solutions tend to be based on COTS architectures that often enforce specific business processes and lack open technology upgrade paths
  – **Flexibility**: Large-scale COTS applications have limits on their configurability, so the application does not readily support the organization’s continuously changing processes
  – **Technology Upgrade**: Upgrading or replacing technology is often very difficult, that hinders the organization in achieving its mission with the current IT technology

• Therefore, too often the organization conforms to the IT needs, instead of IT conforming to what the organization requires
SOA as a Solution

- Service Oriented Architecture (SOA) assists the organization with focusing its IT on solving business problems
  - Measurably links the applications to the organization’s processes to enable the organization to understand the contribution of each application within the process. This enables the organization to determine whether its investment is worthwhile
  - Provides “line-of-sight”
    - From the Organization’s Mission through its Strategies to its Processes to determine the optimal place for the next investment
    - To support the intent of the Federal Enterprise Architecture (FEA), a concept which is being incorporated into the DoD Architecture Framework version 2
SOA as a Solution

(Continued)

• The SOA enables the organization’s IT to support continuous process change
  – Existing “composite” applications can be reassembled to support a change in the organization’s processes in a *Continuous Operational and Development Environment (CODE)*
  – The “Composite” applications are the Services and are assemble from Service Components, each of which is a separate application (which may be a Web Service)
    • Each Service Component may be upgraded to new technology or replaced independently
    • These Service Components are assembled using a process flow which may be redesigned independently of the components being used
A good SOA process will help management to:

- Identify processes and IT systems that are producing minimal or no value for the organization, that is, IT systems that are good candidates for investment
- Recommend deletion or investment (by updates/upgrades or replacement)
- Execute projects that the customer has approved as IT investments
- Evolve toward a service-based business vision/model with the agility to successfully respond to unexpected challenges and opportunities

In other words, a good SOA process allows the organization to use an IT investment process to optimize its processes and the supporting IT systems and applications in a CODE

Northrop Grumman’s Investment-Driven Service-Oriented Architecture (ID-SOA™) is such a process
The ID-SOA™ Process
Enterprise SOA

Composite Application Lifecycle

- Business Process Modeling and Engineering
- Composite Application Modeling and Assembly
- Composite Application Operation
- Composite Application Deployment

Execution Context (Services Infrastructure)

Service Development: Service Development provides activities and functions that support the creation and testing of Service Components
ID-SOA™ Simplified High-level Flow

Service Oriented Architecture Enablement Process

Enterprise Architecture Repository (Supporting FEA & DoDAF)

Mission/Charter & Strategy Driven Investment Process

New Projects

Enterprise SOA Created Here
Creating a System Architecture for an SOA-based IT System
Business Process Modeling and Engineering

Composite Application Deployment

Composite Application Management

Creating Systems Architecture—This activity:
- Decomposes the Business Process Model
- Derives the IT Functions required to enable and support the process—creating the “functional requirements”
- Structures the IT Functions to optimize communications among the functions and minimize the number of redundant functions
- Allocates the Functions to Components—creating the “Component Requirements”
The Overall IT System Architecture Process

Identify the Customer’s Requirements

- Decomposition includes:
  - Process Modeling
  - Functional Modeling
- Derivation Includes:
  - Determining the IT Functions Required
- Structuring/Allocation Includes:
  - Structuring and grouping the Functions to support the Process
  - Assigning the Grouped Functions to Components

The System Architecture Process

Decomposition

Derivation

Structuring/Allocation
The Overall System Architecture Process (More Detail)

- **High-level What**
  - SRD (FRD)
    - Business Processes
      - Traces To
        - Use Cases
          - Design Constrains
  - QFD and Graph Analysis
    - UML Diagram as output of Activity
    - Non-UML Diagram as output of Activity
    - Process Flow Feedback Loop

- **Decomposition** (Detailed What)
  - Communications Diagrams (of Processes)
    - Possible Feedback when Refactoring or Combining Service Components
      - Actor -> SwimLane
      - Event Flow -> Activities
      - Recursive Decomposition
      - Must Meet (Constraining the How)
  - Level 1 Activity Diagrams
    - Level 2+ Activity Diagrams

- **Derivation** (High-Level How)
  - Sequence Diagram
    - Activity => Message
      - Method
  - Class Diagram
    - Message => Method

- **Structuring/Allocation** (Service Functional Design)
  - Communications Diagrams (of Services)
    - Logical (Service) Functional Components
    - Minimum set of Logical (Functional) Components
  - QFD and Graph Analysys
  - Dynamic Functional Modeling
  - Allocate Design Constraints To Service Comp.
  - SAD (SDD)
    - Functional Design
    - Service Component Design
The System Requirements Document (SRD) Used as Inputs (Table of Contents)

1 Introduction
1.1 Purpose
1.2 Scope
1.3 Background
1.4 Identification
1.5 Rapid Implementation Approach
1.6 Document Change Management
1.7 Referenced Documents

2 System Context
2.1 Organizational Mission, and Strategies
2.2 System Scope
2.2.1 In Scope
2.2.2 Out of Scope
2.3 Business Processes
2.3.1 BP1
2.3.2 BP2
2.3.3 BP3
2.3.4 BPI
2.3.5 BPn

3 Use Cases
3.1 Actors
3.2 Use Cases Identified by Actor, Business Process, and Release
3.3 Use Cases

4 Design Constraints
4.1.1 Size and Location of User Community
4.1.2 Interfaces
4.1.3 Customer Furnished or Identified Components
4.1.4 Data Base
4.1.5 System Transition
4.1.6 Training
4.1.7 General Computing Controls & Security
4.1.8 Business Continuity (COOP)
4.1.9 Performance and Availability
4.1.10 Information Retention/Purging/Archiving
4.1.11 Other Constraints
4.1.12 Hardware and Software Standards
4.1.13 Communications and Network
4.2 Customer IT Standards Constraining the Design
4.2.1 Architectural Constraints
4.2.2 Scalability
4.2.3 Reliability/Availability
4.2.4 Service Infrastructure
4.3 Outside Standards and Specifications Constraining the Design

5 System Validation
Appendix A: Acronyms, and Abbreviations
Appendix B: Glossary
Objective: To define processes and activities in sufficient detail to allow the derivation of IT functions to support the process
Derivation

Objective: To determine what IT Functions are required to enable and support the processes and activities determined in decomposition

IT Classes (Services and Service Components) are derived from objects in the process
Objective: To **Structure and group IT functions into Services** to:
- **Minimize** the number of redundant functions
- **Optimize** the grouping of IT functions for allocation
- And to **Allocate** the grouped functions to actual service components

**Step 1: Structure Classes with Communications Diagrams**
Objective: Determine tightly and loosely coupled classes (and functions)
- **Step 1.1 Create Communications Diagram**
  - Initiate by Duplicating IT portions of Activity Diagrams as Communication Diagrams
  - Two Top-level Classes:
    - Service
    - Process Flow
Structuring and Allocation: Step 1

Level 2 – Seq. Dia.: Stage APO*

Parallel Operations Service

The POS Communications Diagram

POS Candidate

Inputs to Communications Diagram

*Asynchronous Parallel Operations
Structuring and Allocation: Step 2

Objective: To Structure and group IT functions into Services to:
- Minimize the number of redundant functions
- Optimize the grouping of IT functions for allocation
- And to Allocate the grouped functions to actual service components

Step 2: QFD and Graph Analysis
Objective: Determine the minimum number of Logical Service Components Required
- Step 2.1 Graph Analysis—Analysis of Communications Diagrams
  - Analysis Principles Include:
    - Noting where one class communicates only with one other class or a small group of classes
    - Noting where there is one or two links between groups of classes
- Step 2.2 Quality Functional Deployment Analysis
Structuring and Allocation: Graph Analysis

Service Candidates are identified by base grouping or clustering of functions and communications interfaces

Interface used here

Parallel Operations Service

- Pre-Process Queue
- Process Queue
- Send IAFIS Response
- Receive IAFIS Response
- Store Match Response
- Receive Match Response

Service Candidates are identified by base grouping or clustering of functions and communications interfaces.
Quality Functional Deployment analysis is common sense with a template

Note: QFD analysis is generally used to determine + or – correlations between two phenomena, but used here to determine:
• If all activities have associated services
• Where services are highly correlated leading to redundant services

QFD Performs the Service (Class) Normalization (combining) activities

| Service* | Rec. IAFIS Response | Rec. Match Response | Send IAFIS Submission | +
|----------|---------------------|--------------------|-----------------------|---
| ++       | ++                  | ++                 | ++                    | +

<table>
<thead>
<tr>
<th>Activity from Activity Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
</tr>
<tr>
<td>++</td>
</tr>
</tbody>
</table>

"The House of Quality"

"Roof" added to determine level of service (functional) redundancy
Structuring and Allocation: Step 3

Objective: To Structure and group IT functions into Services to:

- Minimize the number of redundant functions
- Optimize the grouping of IT functions for allocation
- And to Allocate the grouped functions to actual service components

Step 3: Dynamic Service/functional Modeling

Objective: Use simulation to verify, within a confidence interval, that the functional model will meet the customer’s system requirements

- Currently, the SOA and SA tools suppliers do not support this activity to the degree required by SOA
Structuring and Allocation: Step 4

Objective: To Structure and group IT functions into Services to:
• Minimize the number of redundant functions
• Optimize the grouping of IT functions for allocation
• And to Allocate the grouped functions to actual service components

Step 4: Allocation of Services/Functions to Actual Components
Objective: Allocate the services/functions to components using a make/buy/use tradeoff study procedure
• Step 4.1: Assign Design Constraints to Proposed Components
• Step 4.2: Perform Tradeoff Study
  − Make—the team develops the service component
  − Buy—the service component is purchased from a software supplier
  − Use—the team discovers and uses a service component in the SOA Ecosystem (across the Internet)
Results to Date

• The team has completed the Systems Requirements Document (SRD) to establish the requirements baseline with the customer. This includes:
  – Identifying the processes and activities (in the form of use cases)
  – Identifying the design constraints

• The team has *decomposed* the requirements in the SRD into two levels of Activity Diagrams to define the “detailed what” with the customer. This was delivered as the System Design Document

• The team *derived* the IT functions by translating the detailed Activity Diagrams into Sequence and Class diagrams

• The team then *allocated* these into the specific components (COTS, existing software, server scripts, hardware, and networks) to create the detailed design. This was integrated into the Detailed Design Document and presented to the customer for a very successful Critical Design Review
Questions?
Contact Information

- Robert S. Ellinger, Ph.D.
  - robert.ellinger@ngc.com

- Gabriel Hoffman
  - gabriel.hoffman@ngc.com