Using SOA to Improve Operational Efficiency
A Management Overview

Introducing MIKE2.0
An Open Source Methodology for Information Development
http://www.openmethodology.org
Agenda

**Service-Oriented Architecture (SOA) introduction**

Addressing 5 common questions related to:

- How can a SOA improve the quality of the applications you deliver?
- How can a SOA increase ROI, improve efficiency and reduce recurring operational spend?
- How can a SOA change the way you:
  - Build business solutions?
  - Should be organized?
  - Should be measured and compensated?
- How can a SOA be implemented for Enterprise Data Management?
  - What are the key components of a SOA for Enterprise Data Management?
  - What services can this architecture be used to build?
- What are the potential pitfalls around SOA and can they be avoided?

Other considerations when implementing an SOA
Scope within BearingPoint's IM Suite

Information Management Solution Suite

Delivered through a Collaborative Approach with the IM Profession and our Alliance Vendors

Enterprise Information Management

Supported by Solution Capabilities that provide a foundation for Suite Delivery

BI and EPM

Information Asset Management

Access, Search and Content Delivery

Enterprise Data Management

Enterprise Content Management

Information Strategy, Architecture and Governance

Sets the new standard for Information Development through an Open Source Offering
Service-Oriented Architecture and the MIKE2.0 Methodology

This presentation can be used for running an initial workshop around building a Services Oriented Architecture for Enterprise Data Management and is part of the MIKE2.0 Methodology www.openmethodology.org, an open source methodology for Information Development. It is used during Activity 1.2 of the MIKE2.0 Methodology as a means to bring awareness to new architectural concepts.
SAFE (Strategic Architecture for the Federated Enterprise) is the architecture framework for the MIKE2.0 Methodology. SAFE goes across applications, data, and infrastructure and was designed to accommodate the inherent complexities of a highly federated organization. SAFE covers a number of capabilities, varying from those that are fundamental for the majority of project implementations to advanced capabilities that are only emerging in the area of Enterprise Information Management such as Services Oriented Architectures.
Service Oriented Architecture can be defined as a software design & implementation methodology ("Architecture") of loosely coupled, reusable artifacts ("Services"), which can be integrated with each other, through a wide variety of platform independent service interfaces. Traditionally used more for application integration, SOAs are becoming more widely used for Information Integration.
SOA – What it is and What it is not

**SOA is a style of design, deployment and management of software in which**

- Software functions are built in a way that can be easily integrated with distributed systems
- Service interfaces are exposed through a mechanism that can be accessed in a common and open fashion
- Quality of service characteristics, such as response time, security and transaction recovery, and explicitly identified in the design of the architecture
- A registry is often used for cataloging and dynamic discovery of a given scope of available services

**SOA Clarifications**

- SOA is not simply Web Services. Web Services are a means to achieve a SOA
- SOA is a means to improve technology implementation techniques related to reuse, flexibility and quality
- Different types of services can be built, classifications include: Interfaces Services, Business Services, Data Management Services

**The Myths**

- Any use of services is SOA
- Use of J2EE or .NET automatically results in an SOA
- SOA requires SOAP or, conversely the, use of SOAP results in SOA
- SOA must be built from scratch
How can SOA Improve the Quality of Applications We Deliver?

Reduces overlapping functionality across areas that should be common
  - Resulting in less to build and manage

Removes contradictory functionality and reconciliation requirements
  - By going to a single service for the same functionality

Improves composite behavior of integrated systems
  - To bring together functionality in a highly federated environment

Provides for continuous improvement
  - As opposed to continually re-building new functionality

Uses contemporary technologies
  - That are GUI-driven, standards-based, configurable and far more flexible to change

Provides existing library of capabilities
  - To solve complex technology problems

Removes tight dependencies between system interfaces and business process
  - So it is easier to make ongoing changes
How can SOA Improve the Quality of Applications We Deliver?

**Breakout Questions**

- What are the key issues you face in the quality of your systems?
- When you have an issue between integrated systems where does the problem typically originate?
- What are 5 examples of issues you have had in the last 2 years in integrating federated systems? How they impact business operations?
- Of the issues you had, how were these problems resolved?
How can SOA Improve ROI, Efficiency, and Reduce Recurring Operational Spend?

Reduces the amount of software to be built and managed
- Through better reuse

Provides an inventory of pre-built capabilities
- That can be procured like infrastructure

Shifts more development costs to ongoing maintenance
- As opposed to new software development

Gains economies of scale
- By making it easier to move to a shared development model

Enables external development models
- Through use of standards and discrete pieces of functionality

Allows for evolutionary development
- As opposed to having all capabilities built at once
How can SOA Improve ROI, Efficiency, and Reduce Recurring Operational Spend?

**Breakout Questions**

- Does a large portion of your integration spend involved building new interfaces where a common interface should have provided the functionality?

- How do architects and developers discover if there are existing artifacts that they can re-use for integration?

- How risk-exposed are you to the following scenarios and how does it slow down productivity?
  - Loss of a key developer
  - Business rule knowledge
  - Poorly documented systems

- Do you measure ROI for individual components or on a project basis?
How can SOA Change the Way We build Solutions for the Business?

Implementing a SOA will result in the following changes in how Business Solutions are built:

- Requirements and design should be using a starting point inventory of pre-built assets from across the enterprise.
- Your requirements initiatives may need to look beyond a strict project basis and take more of an enterprise view.
- Adherence to open and common standards will become more important than before.
- Application development will take on a new model to include Composite Applications/Services Oriented Business Applications.
How can SOA Change the Way We build Solutions for the Business?

We should treat architecture as a process to go from a Strategic Conceptual Architecture to an implementable Solution Architecture. Key steps within MIKE2.0 include:

- Revise overall architecture models if required
- Initial assessments of current-state and vision
- Definition of Guiding Principles
- Create Strategic Conceptual Architecture
- Define High Level Solution Architecture Options
- Gathering of Strategic Requirements for Integration and Information
- Definition of the Logical Architecture to understand what capabilities are needed from products
- Map Logical Architecture to Physical Architecture to pick vendors

- Gather Detailed Business Requirements
- Solution Architecture Definition/Revision
- Technical and Implementation Architecture

Strategic Business and Technology Architecture activities are done once, more detailed activities are done for each delivery increment.
How can SOA Change the Way We build Solutions for the Business?

Requirements and Design

- Should use starting point inventory of pre-built assets from across the enterprise
- Should look beyond a strict project basis and take more of an enterprise view

Adherence to open and common standards

- Will become more important than before

Development will take on a new model

- to include Composite Applications/Services Oriented Business Applications

"By 2007, composite applications will be based on the SOA principles of dynamic, extensible, federated interoperability and enabled by XML-based technologies such as Web services." – META

$340 Billion Shift over a 3 year period

Gartner
How can SOA Change the Way We build Solutions for the Business?

Breakout Questions

- Do you explicitly plan for re-factoring as part of an project?

- Do you have an implementation strategy that facilitates continuous implementation for large-scale projects:
  - What happens when business requirements change?
  - Does your organization use a method employing a "blueprint", "roadmap", and "framework" and have consistent definitions?
  - How have you ensured that your incremental progress is aligned with you strategic vision and tactical project goals?
  - Have you had issues aligning tactical projects with strategic initiatives?
  - Have you had an experience that you strategy was either too high-level, too detailed, out of touch or too serial?

- Do you have a policy towards where business rules are to be located; are there complex business rules within your integration environment?
How should SOA Change the Way We are Organized?

Consider an organizational model along the lines of:

- Application Development teams focused on Function and Business Processes along business verticals
- Centres of Excellence for Infrastructure and Integration Data Management

A physically central organization need not be required – A common set of governance standards is the key. Most organizations should modernize their Governance processes significantly.
How should SOA Change the Way We are Measured and Compensated?

SOA measurements should be across all level of the organization, from the Executive Level to the Architect to the Business Analyst

Measure the SOA artifacts that you produce. Closely track metrics for each service such as:

- Reusability across the enterprise
- Reliance on open standards
- Degree of business or technical functionality provided
- Reliability and performance
- Usability for designers and developers
- Manageability for operations
- Time-to-market for changes
- Data Management capabilities

Heavily market well-built services internally and promote their reuse

Offer major incentives for services that are built that act as design patterns or building blocks
How should SOA Change the Way We are Measured and Compensated?

**Breakout Questions**

- Is software re-use measured?
- Do you conduct impact analysis on the cost of changes to reusable software?
- Are Senior Leaders compensated for delivery that benefits the enterprise?
- Are practitioners at all levels measured/compensated to focus on best practices in information management?
How can a SOA be Implemented for Enterprise Data Management?

Services Oriented Architectures are becoming more widely used for Enterprise Data Management.

These architectures apply the same principles of reusability, loose coupling and open and common standards.

Vendor technologies typically used for Data Integration have expanded their capability set to enabled services-oriented implementation techniques.

Implementation examples where there is a strong case for SOA are:

- Master Data Integration Hubs (e.g. Customer Data Hub)
- Real-Time Warehouses
- Application Co-Existence between multiple systems for de-commissioning
- Data Quality Management Services

Metadata management through a Services Oriented Architecture is becoming a more widely used technique to share this information with distributed systems.
What are the Key Components of a SOA for Enterprise Data Management?

### Enterprise Applications
- Product Systems
- Sales Systems
- Support Systems
- Tightly Integrated Applications

### 'Integration Apps' Developed Over Time
- Integration Infrastructure
- Orchestration of Integration Processes

### Composites Applications
- Data Quality Management

### Common Data and Metadata Services

#### Technical DM Services
- Shared SCD job
- Shared Functions
- CDC Capabilities
- Process Automation
- Data Standardisation
- Technical Functions

#### Metadata Services

#### Operational Metadata
- Business Metadata

#### Interface Services
- Service Requestors
- Service Providers
- Mediator Services
What are the Key Components of a SOA for Enterprise Data Management?

**Interfaces Services** encapsulate discrete application functions and expose them via the Common Messaging Model. Although logically seen as one entity, an Interface Service often contains multiple physical components. Interface Service and implemented as either **Service Requesters** or **Service Providers**. Multiple services can be brought together into a **Composite Application**.

**Data Management Services** are specialized Business Services that facilitate data synchronization. In the past, the functionality provided by Data Management services has been associated with batch data integration and offline data quality improvements. The need for real-time synchronization of data to distributed systems mandates that these capabilities be available for invocation in an event-based fashion. Examples include standardisation services, matching services and de-duplication services.

Across the Enterprise, redundant data exists in a number of applications for multiple entities. The **Data Mastering Model** governs the exchange of information between applications by defining the rules for data ownership of a particular business event.

The **Common Messaging Model (CMM)** is the framework for modelling "data in motion" and enables standardised information exchange across multiple applications, departments or organizations. CMM Messages are built based on standards (e.g. industry models, corporate standards) and evolve over time based on usage and new requirements.

**Services Orchestration** provides discovery and scripting capabilities that allow is to find services across the enterprise, link them together with orchestration scripts and run the execution of this process with an orchestration engine. Services Orchestration is supported by open and common standards for the development, integration and operations of an enterprise services environment.

A centralised **Service Container** provides the repository of existing services; different technologies use different types of service containers (e.g. UDDI for Web Services). In addition m

**Metadata Services** provide fine and coarse grained services to build reusable platform independent metadata capabilities to drive a Model Driven Architecture. Metadata Services are enabled by the Foundation Capabilities and Enabling Technologies for metadata that have emerged from standards bodies such as the Object Management Group (OMG), the Java Community, Vendors and other standards groups. There is a metadata management overlay across each architectural component.
What Services can this Architecture be Used to be Build?

<table>
<thead>
<tr>
<th>CDI Services</th>
<th>Common Services</th>
<th>PDI Services</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Candidate Customer Domain</strong></td>
<td><strong>Candidate Product Domain</strong></td>
<td>All the above are examples of Business Services. However there are a number of candidate services which do not necessarily involve master data.</td>
</tr>
<tr>
<td>- Create/Modify Whole of Customer</td>
<td>- Design a new Product</td>
<td></td>
</tr>
<tr>
<td>- View Whole of Customer</td>
<td>- Modify a current Product</td>
<td></td>
</tr>
<tr>
<td>- View Customer Membership in Groups or Hierarchies</td>
<td>- Migrate new/modified product to selected production environments</td>
<td></td>
</tr>
<tr>
<td>- Check to see if a customer exists</td>
<td>- Review product options in general</td>
<td></td>
</tr>
<tr>
<td>- Query customer contracts or Service Level Agreements (SLAs)</td>
<td>- View related products</td>
<td></td>
</tr>
<tr>
<td>- View all accounts associated with a customer</td>
<td>- View pre-requisite products</td>
<td></td>
</tr>
<tr>
<td>- View customer reporting requirements</td>
<td>- Provide Product Price Quote(s)</td>
<td></td>
</tr>
<tr>
<td>- View all registered IDs for a given household</td>
<td>- Validate Product Availability</td>
<td></td>
</tr>
<tr>
<td>- View churn likelihood for a given customer</td>
<td>- Validate a proposed Sales Order</td>
<td></td>
</tr>
<tr>
<td>- Audit customer for missing or invalid information</td>
<td>- Query product functionality</td>
<td></td>
</tr>
<tr>
<td>- View customer status summary</td>
<td>- Query product configuration rules</td>
<td></td>
</tr>
<tr>
<td>- View customer profiles for marketing, service assurance and billing (finance)</td>
<td>- View Marketing Product Catalog</td>
<td></td>
</tr>
<tr>
<td>- View Lifetime Value of a Customer</td>
<td>- Request prospective products for a customer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Request specific products for a particular customer segment</td>
<td></td>
</tr>
</tbody>
</table>
How should SOA Change the Way We are Measured and Compensated?

Breakout Questions

- Have you implemented any services for "Business Integration"?
- Is your current SOA strategy inclusive of projects that are more traditionally thought of as "data projects" such as data warehouses or data migrations?
- Where in your Data Management environment would you benefit from re-use?
- Does your current technology set for data integration allow tightly-coupled steps in the integration process to be exposed as services?
- Do you expose your metadata artifacts out to other systems in a form that can be effectively re-used?
What are the Potential Pitfalls and How do We Avoid them?

Services Oriented Architecture pitfalls and how they can be addressed:

Thinking an SOA architecture will help an unstable application environment
- If an operational system is unstable and unreliable, it's not going to be fixed by SOA – make sure to address critical issues with these core systems
- Quantitatively understand data quality issues within operational systems and realize the impact of these issues for integration
- Recognize that automating integration into an unstable application can make it more difficult to manage

Ignoring the complexity of managing Enterprise Data
- A SOA architecture should not just be for application integration, a SOA strategy should also be incorporated for data management projects
- Information Development should be just as much of a priority as developing applications and infrastructure
- Implement a Data Governance programme as a means of issues prevention as opposed to reaction to issues

Assuming that SOA technologies will automatically provide flexible systems that are easy to manage
- Set firm guidelines for the implementation of SOA standards
- Base development on existing design patterns and leading code artifacts
- Use of open standards – don’t get locked into a specific vendor technology

Ignoring technology risks of off-the-shelf software
- Conduct proper diligence during the selection process
- Use an architectural model that goes from strategic conceptual all the way to solution implementation in an iterative fashion
- Use contingency planning for new technologies

Maintaining poor Software Development Lifecycle practices
- Take the opportunity to add sophistication to your configuration management, defect management and testing processes
- Automate the testing and deployment lifecycle as much as possible
- Focus appropriate amount of review times based on risk

Failing to put the proper skills set or organizational structures in place for the SOA implementation team
- Have the strongest developers play leadership roles around framework and common services development
- Realize that Composite Applications and SOBAs are more like operational applications in terms of functionality they contain – i.e. may hold complex business rules
- Define a governance model where architecture, delivery and management are closely aligned with joint responsibilities
- Realize that enterprise-wide initiatives do not happen on their own, they must be explicitly planned. That doesn’t mean, however, you have to tackle all issues at once

Ignoring the security challenges posed by a federated, integrated environment
- Understand security challenges from webifying, XML and building external interfaces
- Balance emerging standards with traditional practices
- Make security a key focus of your Data Governance programme

Failing to Improve Governance Practices
- Modernize your Governance process related to software deployment, testing timeframes and delivery.
- Remove antiquated and ineffective SDLC processes that dramatically slow down the SOA lifecycle
Other Considerations

Other Important Points to Consider:

- Most organizations are increasingly federated, with more systems and more data than ever before. Therefore, they have no choice but to dramatically improve their techniques related to integration and data management.
- Creating reusable components that will be shared across the enterprise means they will be shared with a wider user community. This means that standards around definition and testing are even more important.
- The goal should be to make an SOA integration environment (especially system interfaces) like other forms of infrastructure, by making it:
  - Standards based
  - Well-defined, inventoried and something we can procure on demand
  - Reusable and reliable
  - Modular
- Make sure your SOA business case is comprehensive and includes the cost of decommissioning existing infrastructure - remember that every technology has a half-life.
- Make sure there is a "balance of power" in the organization related to Architecture, Delivery and Leadership across each of these areas for delivering solutions.
- The perceived strict choice between build vs. buy is inconsistent with how software has evolved. All software requires some level of construction; build options are made much easier through frameworks.
- The options from the open source community will continue to get better and in some cases already offer excellent alternatives to commercial products. Every organization needs an open source strategy that should factor into its approach to SOA.
- Traditional forms of documentation do not facilitate an effective approach to building an SOA. Modern tools and techniques for defining and sharing services metadata must be part of your strategic approach.
- Systems and the means to integrate them is only as good as the underlying data. A comprehensive approach to Data Management must complement your SOA strategy.
- Most organizations do not focus on continuous improvement of their integration environment. Explicitly fund re-factoring of delivered software as part of your business case.