Fundamental SOA Concepts

Jason Bloomberg and Ronald Schmelzer
Senior Analysts
ZapThink, LLC

Take Credit Code: FSCIPR

We’ve had IT challenges for years ...
... but even after yesterday’s promises...

... we still have the same IT mess, only worse.
The Challenge of Complexity...

Yet the Best Technology...

- Is complex on the inside yet simple on the outside

- The secret is the abstraction layer
Level Set – What is SOA?

- SOA is architecture – a set of best practices for the organization and use of IT
- Abstracts software functionality as loosely-coupled, business-oriented Services
- Services can be composed into business processes (which are also Services) in a declarative manner

How to Think Service-Oriented

- Service Orientation is about change
- IT must respond to change and enable innovation
SOA Abstracts the Plumbing

- The goal is reusable, composable business Services
- Many different approaches to implementation

SOA shifts the way we think

<table>
<thead>
<tr>
<th>Traditional Distributed Computing</th>
<th>Service Oriented Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designed to last</td>
<td>Designed to change</td>
</tr>
<tr>
<td>Tightly Coupled</td>
<td>Loosely Coupled, Agile and Adaptive</td>
</tr>
<tr>
<td>Integrate Silos</td>
<td>Compose Services</td>
</tr>
<tr>
<td>Code Oriented</td>
<td>Metadata Oriented</td>
</tr>
<tr>
<td>Long development cycle</td>
<td>Interactive and iterative development</td>
</tr>
<tr>
<td>Middleware makes it work</td>
<td>Architecture makes it work</td>
</tr>
<tr>
<td>Favors Homogeneous Technology</td>
<td>Favors Heterogeneous Technology</td>
</tr>
</tbody>
</table>
Increasing Business Benefits of SOA

- Reduce Cost of Application Maintenance
- Reduce Cost of Point-to-Point Integration
- Increase Efficiency thru Service Reuse
- Increase Visibility & Control
- Improve Business Agility

The Economics of Integration

The Relative Costs of Different Integration Approaches:
- Custom Integration
- Traditional EAI, B2Bi
- Web Services "Adapters"
- Service-Oriented Integration

Copyright © 2006, ZapThink, LLC
Increasing Reuse over Time

Service Catalog

Existing Applications

Service Model

Service-Oriented Business Applications

Presentation Services

Coarse-Grained Business Services

Data Access Services

Legacy API Services

Business Agility

• Companies require Business Agility...
  » Responding quickly to change, and
  » Leveraging change for competitive advantage

Agility is the key to innovation
SOA Hotspots

- Four key areas of SOA investment
  - Reduction in integration expense
  - Increase in Service / asset reuse
  - Increase in business agility
  - Enablement of governance & compliance

- Key Problem areas
  - EAI replacement
  - Legacy enablement/migration
  - Shared Service development
  - Governance
  - Embedding processes in the extended enterprise

Don’t Get Lost in the Terminology

- There are many styles for SOA implementation
- Focus on your goals: Reuse? Governance? Reduced integration cost? Agility?
The Secret Sauce: Loose Coupling

- Power of Abstraction
- Power of Open Standards
- The Challenge of Loose Coupling
- Loose Coupling enables Change

The Challenges with doing SOA right

- **Architectural issues**
  - Enterprise Architecture capability maturity
  - Building SOA the right way
  - Enabling effective governance

- **Tool & infrastructure requirements**
  - XML performance issues
  - Security & management challenges
  - Contract & policy development and implementation
  - Composite application development
  - Metadata management

- **Organizational issues**
  - Pulling together the SOA team
  - Who’s in charge of Services?
  - How will you pay for Services?
  - How will SOA affect your organization moving forward?
Characteristics of SOA Implementations

- Services are *loosely coupled* – making a change to a Service provider does not mandate changing any Service consumers.

- Business processes are composed of Services, and are in turn exposed as Services.

- Services are *policy-driven* – business users can change how a Service behaves.

- Systems are *inherently integrated* by virtue of composable Services – not through layers of middleware.

Characteristics of SOA Implementations (Cont.)

- Services *leverage* legacy systems – SOA does not mandate replacement of runtime infrastructure.

- In SOA, *metadata* control how the system behaves instead of code – business logic trumps application logic.

- In SOA, it’s the *contracted interface* that matters, more so than the underlying runtime environment.
What is "legacy"?
- Host-based systems...
- SCM, CRM, and other business apps...
- Anything that’s in use...

Legacy systems enable a significant amount of mission-critical functionality

Rip-and-Replace vs. Maintain-and-Extend

The first step to extending functionality: abstracting the implementation – aka “Service Wrappers”

Secure Service Interfaces

- XML is text-based, human readable – give hackers the keys and the instructions
- Existing security inadequate to address content security issues
- Authorization, threat prevention, confidentiality key issues
The Security Context Challenge

Build a Governance Framework

- How will you communicate corporate policies to your team?
- How will you enforce those policies?
- What is the role of SOA in your governance plan?
The Role of Governance...

- Establishing and communicating the policies that employees must follow
- Giving employees the tools they need to be compliant with those policies
- Providing visibility into the levels of compliance in the organization
- Mitigating any deviations from established policy

SOA Governance

- Policy management
  - SOA configured & controlled via metadata, including policy
- Visibility
  - Services abstract heterogeneous data sources, providing necessary business intelligence
- Flexibility
  - Ability to build Services that address compliance issues and adjust them as regulations or business needs change

*Not just governance of SOA... governance in the context of SOA*
Manage Services

- Loose coupling is a *management* problem: require Services to behave as expected
- Must handle infrastructure management issues “behind the scenes”
- If a business user knows how Services work, then something is wrong

Contract-First Development

- Service contracts specify required functionality to IT and provide functionality to the business
- Service contract thus acts as a requirements document for all Service implementation activities
Build the SOA Metamodel

- Architecture is *design* – requires a formal approach
- A metamodel is a model of models
- Models provide ongoing architectural guidance
- The core model of the SOA Metamodel is the *Service Model*, which represents Services in production and ongoing requirements for Services

Service-Oriented Process

- Compose Services into process that are themselves Services
- Put responsibility for such agile composite applications into the hands of the business
- Business logic increasingly represented by configuration of composite applications
Business Process the Old Way...

- People plugged into rigid processes
- Inflexible & brittle

Business Process the Service-Oriented Way...

- IT resources (among other resources) available to the business as needed
- Business users create composite applications by composing Services & configuring processes dynamically
Semantic Integration

- Balance loose coupling with semantic consistency
- Leverage industry standard vocabularies
- Tools still immature – still requires substantial manual work

Dynamic Discovery

- Service consumers identify and select appropriate Services dynamically at runtime
- Most applicable in business-to-business scenarios
- Advanced capability – many years away from a reality
Phases of SOA Maturity

Enterprise SOA Buildout

Cross-Departmental SOA

“Grass Roots” Web Services Implementations

SOA Pilots

Mission-Critical SOA

Building SOA the Right Way: Take an Iterative Approach

- Top-down only: have the plan, may not be able to execute
- Bottom-up only: build Services, may not be reusable
- SOA planning *must* be both
  - Develop the vision (but not the details) ahead of time
  - Service development should be iterative
SOA Pilots

- A few high ROI Services
- Build acceptance for SOA
- Get team up to speed
- Work out the kinks
- Pilot the governance model
- Part of an iterative approach to SOA

DANGER! Avoid the SOA Pilot Pitfall
- Piloting only the Services instead of the architecture
- Remember, the pilot is one step on the roadmap

SOA is Architecture!

- Remember...SOA is architecture – in particular, Enterprise Architecture, including:
  - An aggregated architecture of all the individual IT systems within an organization
  - The human element within the enterprise
  - Systems, people, and organizational constructs at other companies that have relationships with the enterprise
  - Individual consumers who are that enterprise’s customers
  - Corporate governance
Building Agility with Agility

Traditional Distributed Software Development
- "Waterfall" – Gather requirements, design, develop, test, deploy as separate steps
- Works great when things don’t change
- Typically fails!

SOA – applications are never complete, Services are always in flux
- Traditional SDLC wholly inadequate

• Reuse: The Holy Grail of IT

Mission-Critical SOA

• Services incorporated into core business processes
• Issues of security, management, granularity resolved
• Provides ongoing agility – built to change
The “Ivory Tower” Problem

- Architects create design and other artifacts, but don’t have the authority or mandate to require their use
- Development team considers them optional
- Business likes idea of architecture in principle, but short-term needs trump best practices
- When architects are external consultants, the “not invented here” syndrome makes the Ivory Tower worse

Cross-Departmental SOA

- Organizational issues of governance and control become paramount
- Long-term architectural plan critical
- Increased focus on semantic issues
Service Domains

- A Service Domain is a logical grouping of shared Services with a common business context
- Manage Services by managing the Domains
- Move away from traditional IT silos

Enterprise SOA Buildout

- The corporate enterprise architecture becomes Service-Oriented
- Service lifecycle becomes dominant framework for IT change
- Service abstraction frees company to retire legacy as appropriate
The Service-Oriented Enterprise

- IT resources are available on demand to businesses as Services
- The Service-oriented abstraction layer enables companies to run their operations and conduct business with each other in a dynamic and automated fashion
- Business drives IT, and agile IT enables agile businesses