Key Challenges in SOA

Jason Bloomberg
Senior Analyst
ZapThink, LLC

Take Credit Code: CHALSOA

zapthink

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

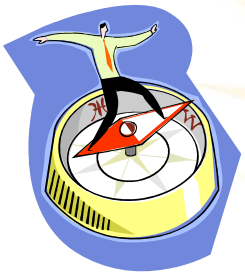
The Benefits of SOA...

- ✓ Reduced cost of integration
- ✓ Improved value from legacy applications
- ✓ Reuse leading to reduced redundancy
- ✓ Greater visibility for governance & compliance
- ✓ Increased reuse of software assets
- ✓ Business agility…

Business Agility

- Companies require *Business Agility…*

»Responding quickly to change,
and
»Leveraging change for competitive advantage



The Challenge of Managing Change

- **SOA is all about continuous and sometimes unpredictable change**
- Development issues
  - How to build the right Services?
  - How to handle metadata management?
  - How to develop changing policies?
- Runtime issues
  - Service availability
  - Policy enforcement
  - Guarantee Service-level agreement
  - Maintain low TCO

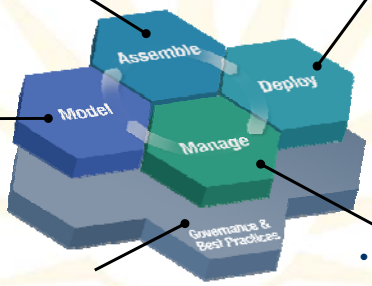
Building for ongoing change: The Death of the SDLC

- Traditional Distributed Software Development
  - "Waterfall" – Gather requirements, design, develop, test, deploy as separate steps
  - Works great when things don't change
  - Typically fails
- Improvement: Iterative Approaches
  - Same order, with overlapping cycles
  - Better, but still assumes project completion
- **SOA – applications are never complete, Services are always in flux**
  - *Traditional SDLC wholly inadequate*

The SOA Lifecycle: Beyond the SDLC

- Discover
- Construct & Test
- Compose

- Integrate:
  - People
  - Processes
  - Information



- Gather requirements
- Model & Simulate
- Design

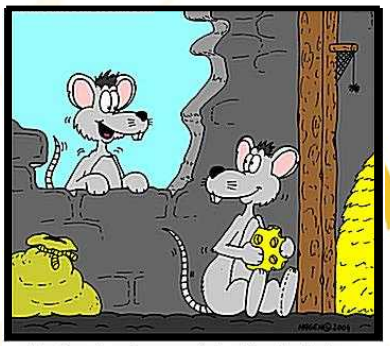
- Financial transparency
- Business/IT alignment
- Process control

- Manage applications & services
- Manage identity & compliance
- Monitor business metrics

Source: IBM

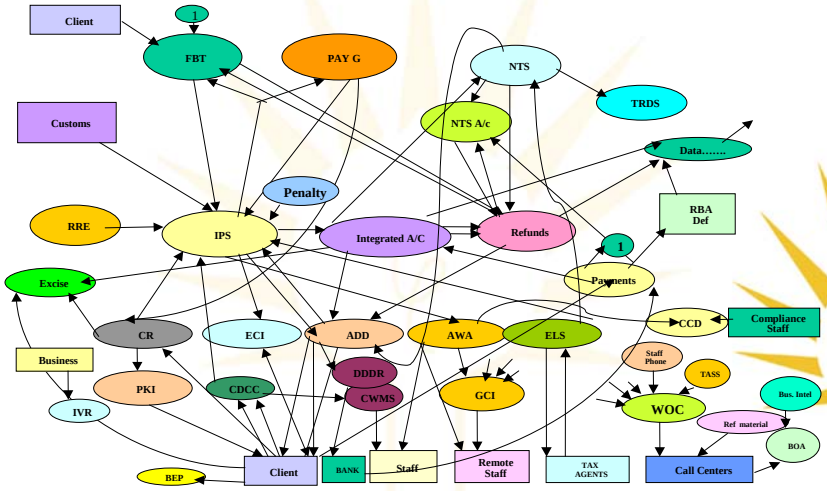
Planning for Change

- Take an *iterative*, *agile* approach
- Consider organizational as well as technical issues
- Architecture team should drive SOA: top-down + bottom-up



Hey Gus, there's a guy playing the pipe in town: Let's go and follow him...

The Challenge of Complexity…

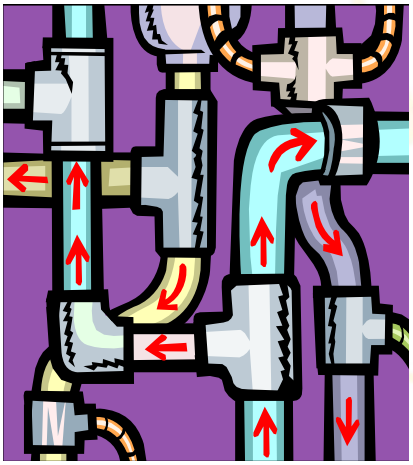


The Best Technology…

- Is complex on the inside yet simple on the outside

- The secret is the *abstraction layer*

SOA Abstracts the Plumbing



- The goal is reusable, composable business Services
- What kind of infrastructure enables loosely coupled, composable, asynchronous Services?
- Many different approaches to implementation

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?

Challenge: 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

SOA as Enterprise Architecture

	DATA	FUNCTION	NETWORK
Planner	Entity = Class of Business Thing	Process = Class of Business Process	Node = Major Business Location
ENTERPRISE MODEL (CONCEPTUAL)	e.g. Semantic Model	e.g. Business Process Model	e.g. Logistics Network
Owner	Ent = Business Entity Reln = Business Relationship	Proc. = Business Process I/O = Business Resources	Node = Business Location Link = Business Linkage
SYSTEM MODEL (LOGICAL)	e.g. Logical Data Model	e.g. "Application Architecture"	e.g. "Distributed System Architecture"
Designer	Ent = Data Entity Reln = Data Relationship	Proc = Application Function I/O = User Views	Node = I/S Function (Processor, Storage, etc) Link = Line Characteristics
TECHNOLOGY MODEL (PHYSICAL)	e.g. Physical Data Model	e.g. "System Design"	e.g. "System Architecture"
Builder	Ent = Segment/Table/etc. Reln = Pointer/Key/etc.	Proc = Computer Function I/O = Screen/Device Formats	Node = Hardware/System Software Link = Line Specifications
DETAILED	e.g. Data Definition	e.g. "Program"	e.g. "Network Architecture"

SOA

Zachman Framework source: The Zachman Institute for Framework Advancement

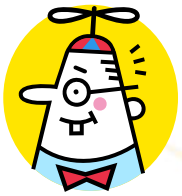
The ZapThink SOA Roadmap

- A roadmap to SOA success that lowers risk
- Intended to be adopted iteratively

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

Bring Together Different Mindsets



- Developer Mindset: "Bottom-Up"
  - Everything is a Service or an Interface
  - Goal: connect Services
  - Method: Use objects and App Servers
  - Problem: Too many things to connect!
- Business Mindset: "Top-Down"
  - Everything is a Process
  - Goal: Run business efficiently: manage processes
  - Method: Use diagrams and flowcharts
  - Problem: How can you turn "shelf-ware" into software?

Challenge: Building the *Right* Services

- The trick of building composable Services is *building at the right level of granularity*
- Challenges:
  - Engraining business logic into code
  - Decomposing legacy services that are not fine-grained enough
- Method
  - Top-down process decomposition, vs. bottom-up Service development
  - Must be iterative

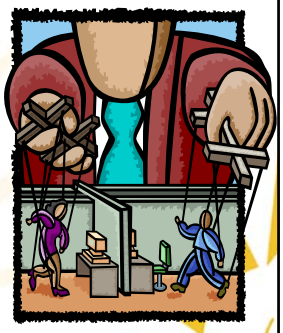
Challenge: Too Many Services

- Pros:
  - Services reduce cost of integration
  - Improve business agility
- Cons:
  - May not be reusable
  - May be redundant
  - Hard to manage
  - May proliferate in an ungoverned manner

Rise of Rogue Services

Challenge: 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



From project to program to sustainable process

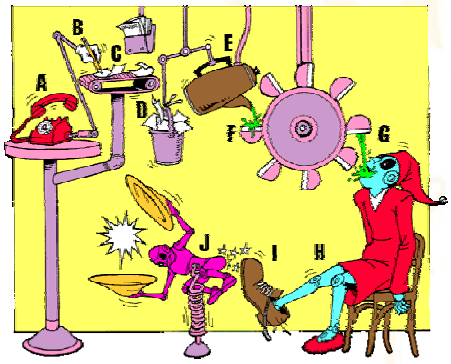
SOA Governance

- 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

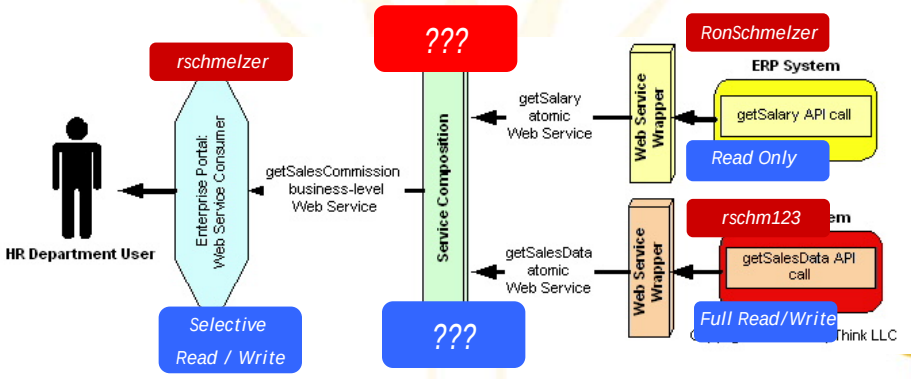
The XML Processing Problem

- Steps required to process XML for each message:



- Receive
- Authenticate
- Decrypt
- Validate
- Canonicalize
- Transform
- Parse
- Aggregate
- Sign
- Encrypt
- Transmit

The Security Context Challenge



The SOA Reliability Conundrum

- Three aspects of reliability
  - Reliability of runtime
    - Will the system work as planned?
  - Reliability of design
    - How can we make sure we have consistency?
  - Reliability of people
    - What if people don't behave as they should?
- Traditional view of reliability now insufficient
  - Reliable components do not guarantee reliable Services
  - Loosely coupled composite apps require new approaches to reliability
- ***SOA ROI hangs in the balance***

Challenge: Contracts & Policies

- **Service Contract Challenges**
  - What does the Service provide?
  - What are its quality of service constraints?
  - What are the constraints on consumers?
  - What are the message/semantic formats for requests?
  - How can consumers communicate with the Service?
- **Service Policy Challenges**
  - How do you translate business policies into executable metadata?
  - What is the format for policies?
  - How do you handle policy lifecycle?

People, Change and Fear

- People are inherently resistant to change
- People consider job security, authority and responsibility when asked to share
- Fear is the strongest emotion of all!

Ownership of Services

- Shared Services cross organizational boundaries
- Siloed IT management styles are becoming *obsolete*
- A new role for enterprise architects

How do you Budget?

- Who pays for Service development?
- Who pays for Service changes?
- Should architects have their own budget?

Revenge of the chargebacks!

Interaction Challenges

Services blur the Application / Network Boundary!

Developer/Architect

Network Operations

- ***Cultural Issues***
  - Network Ops and Developers don't talk to each other
- ***Budget issues***
  - Who pays for Service Infrastructure / intermediaries?
- ***Responsibility issues***
  - Who is in control of policy?

More Interaction Challenges

Architecture is difficult to mandate

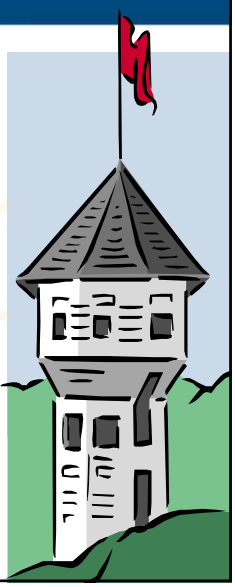
Architect

Development/Testing

- ***Management issues***
  - People tend to avoid risk, stay within "comfort zone" - may appear stubborn
- ***Technical issues***
  - Architecture is a difficult subject
- ***Cultural issues***
  - The "Ivory Tower" problem…

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



Convincing Technical Specialists

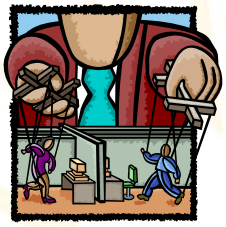
- Among the most risk-averse are *technical specialists* – mid to late-career experts in a (typically legacy) technology (e.g., "COBOL Jockeys")
- Architectural change threatens their careers
- Solution:
  - Work with younger developers to build acceptance for SOA (eventually the TS's will come around)
  - Take a "leave and abstract" approach over "rip and replace"

One Approach: 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

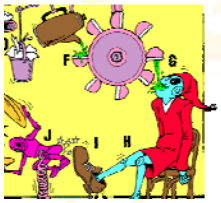
Resolving SOA Challenges…

- People…



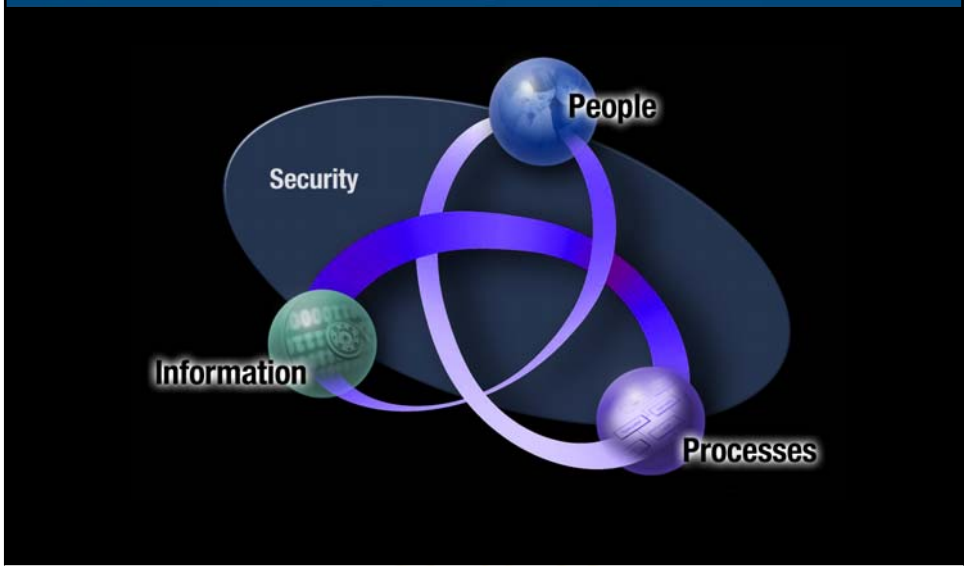
- Process…

- Information…



zapthink

SOA and IBM



Thank You!

zapthink

ZapThink is an industry analysis firm focused exclusively on XML, Web Services, and Service-Oriented Architectures.

Read our new book, *Service Orient or Be Doomed! How Service Orientation Will Change Your Business.*

Available wherever books are sold.

Jason Bloomberg

jbloomberg@zapthink.com

zapthink