Prescriptive SOA Architecture

Session 2290

Gary Gershon, VP, Enterprise Application Architecture
Chubb Group of Insurance Companies
gggershon@chubb.com
May 1, 2012 4:45-6:00
Palazzo P- Venetian
Abstract – Prescriptive SOA Architecture

This session provides insight on Chubb’s initiative to extend the value of Service Oriented Architecture (SOA) further into enterprise application architecture using components at the enterprise, service and programming levels. Chubb’s efforts meld two established and successful approaches: a prescriptive architecture where the number of decisions and tradeoffs to be made by architects is reduced, and an SOA to speed development and reduce costs via asset reuse. Chubb is extending the reach and scope of its core WebSphere and IBM BPM-based SOA application integration to encompass system interfaces and components that were previously out-of-scope for web services and MQ, building upon industry standards such as SCA and OSGi.
Agenda

- Chubb, SOA, and Architecture
- Focusing on IT Outcomes
- Architectural Visions
- Assembly of Solutions
- Design Principles
- Notations, Platforms, Engines, and Languages
Chubb, SOA, and Architecture
The Chubb Group of Insurance Companies

• Chubb Corporation is a holding company for a family of property and casualty insurance and affiliated companies known as The Chubb Group of Insurance Companies or Chubb.

• Chubb is the eleventh-largest P&C U.S. insurer with a worldwide network of 120 offices and 10,100 worldwide employees in 27 countries.

• In 2010, the Chubb Corporation reported $50 billion in assets and $13 billion in revenues.

• Chubb serves commercial and personal customers through approximately 8,500 independent agents and brokers worldwide.

• Fortune magazine in 2010 ranks Chubb as the 185th largest US-based corporation. Forbes lists Chubb as one of America’s 100 Most Trustworthy Companies.

• Three of Chubb’s member companies are among the select insurers that have achieved A.M. Best Company’s highest rating for more than 50 years. Chubb also earns high ratings from Standard & Poor’s and Moody’s for financial strength.

See About Us at http://www.chubb.com
Chubb, ICC, and SOA

- **The ICC — Integration Competency Center**
  - Assist with design, best practices, implementation, and thought leadership for application integration

- **Very successful at Chubb**
  - Savings from reuse
  - Service app building blocks

- **SOA concepts becoming increasingly core to building application solutions**
From Integration (ICC) to Application Architecture (EA AA)

A significant factor in the success of the ICC has been hands-on management and prescriptive standards and policies

- Patterns and Standards
- Documentation and Process
- Platforms and Tools

Chubb is extending these building blocks of effective outcomes to enterprise applications

- Services and Components
- Assembly and Wiring
- Prescriptive Solution Frameworks
New Expectations are Accelerating Demands on SOA

- Big Data
- Mobile
- Cloud
- BI/Machine Learning
Focusing on IT Outcomes
Focusing on Better IT Outcomes

Service-oriented Architecture (SOA) can facilitate the goals of application and enterprise architecture practice:
Better IT Outcomes by Leveraging SOA

Achieving

Costs → Quality

Time

Better IT Outcomes

Via

SOA

Reuse
Flexibility
Durability
Objectives, Outputs, and Outcomes

A core project management concept is the rationalization of:

Objectives, Outputs, and Outcomes

A prescriptive SOA architecture avoids redundant efforts and reduces the time and costs of outputs that don’t directly contribute to the target outcomes.
Traditional Project Metrics

Many IT projects execute against traditional metrics

- Functional Requirements
- Non-functional Requirements
- Usability Expectations
- Schedule and Budget
Adding Architectural and Quality Metrics

Adding architectural and quality metrics provides long term considerations.

Functional Requirements | Non-functional Requirements
Usability Expectations | Schedule and Budget

Durability of core investment | Leveraging of existing assets
Consistency of enterprise implementations | Extensibility and reusability of new assets

Interface Complete | Transactionality Complete
Business Process Complete | Monitoring and Reporting Complete

Architectural Metrics
IT Solution Quality Metrics
### Interface Complete

<table>
<thead>
<tr>
<th>Compelling primary Interface (e.g. mobile or desktop)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highly usable and fully functional user experience on alternative devices</td>
</tr>
<tr>
<td>Standards-based services interfaces for BPM automation</td>
</tr>
<tr>
<td>Interfaces that are stateless, multithreaded, and performant for batch processing</td>
</tr>
</tbody>
</table>
Targeting “Complete” Solution architectures – Transaction

<table>
<thead>
<tr>
<th>Transaction Complete</th>
<th>Traditional database ACID – atomic, consistent, isolated, durable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For SOA, automated transaction compensation</td>
</tr>
<tr>
<td></td>
<td>Transaction identification, tracking, and event emitting for reporting, and monitoring</td>
</tr>
<tr>
<td></td>
<td>Transaction queuing and retry automation or assistance after failures</td>
</tr>
<tr>
<td></td>
<td>Transaction batch management surrounding atomic transactions</td>
</tr>
</tbody>
</table>
### Targeting “Complete” Solution Architectures – Monitoring

<table>
<thead>
<tr>
<th>Monitoring and Reporting Complete (BAM)</th>
<th>Real-time monitoring, status, and workload dashboards</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Message-based event emitters for separation of monitoring and reporting from business application</td>
</tr>
<tr>
<td></td>
<td>Active mobile device notification of out-of-range KPIs or SLAs</td>
</tr>
<tr>
<td></td>
<td>Historical and trend reporting of business and productivity metrics</td>
</tr>
<tr>
<td></td>
<td>Data extraction, archiving, and cube analytics</td>
</tr>
<tr>
<td></td>
<td>Technical as well as business monitoring and reporting</td>
</tr>
</tbody>
</table>
# Targeting “Complete” Solution Architectures – BPM

<table>
<thead>
<tr>
<th>Business Process Complete (BPM)</th>
<th>Process management from moment of inception to completion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Able to locate and report on the status of transactions at any point during processing</td>
</tr>
<tr>
<td></td>
<td>Span automated and human activities across business units, systems, and departments</td>
</tr>
<tr>
<td></td>
<td>Case management for individual and time-separated transactions</td>
</tr>
<tr>
<td></td>
<td>Complex and non-standard transactions (referrals in insurance)</td>
</tr>
<tr>
<td></td>
<td>Distinct transactions and process flows over time for same customer workitem (e.g. Date-of-Service linking of transactions in health claims)</td>
</tr>
<tr>
<td></td>
<td>Workload balancing, assignment, affinity, reporting, and absence management</td>
</tr>
</tbody>
</table>
Architectural Visions
Architectural Visions

• Christopher Alexander – Patterns
• John Zachman – Enterprise Architecture Framework
• Geoffrey Moore – Systems of Engagement
• Gartner – Pace Layering
• Corporate Executive Board – Reference Architectures
• Open Group - TOGAF
For row houses, place houses along pedestrian paths that run at right angles to local roads and parking lots, and give each house a long frontage and a shallow depth.

long thin houses

cars

common green

CONSTRUCTION

that they give acoustic isolation and make a comfortable “thunk” when they are closed.

solid and with glass

“thunk”
Christopher Alexander – Patterns

• Christopher is an award-winning academic thought leader on architectural patterns
  – Applied to cities, villages, houses, rooms, and furniture
  – The originator of the concepts applied to programming patterns
  – Author of several books including
    • A Timeless Way of Building (1979)
    • A Pattern Language: Towns, Buildings, Construction (1977)

• Key Concepts
  – Architecture follows patterns at many levels of abstraction
  – These patterns enable effective communications on best solutions
  – Patterns result in higher quality designs

• Application
  – We can apply these pattern concepts to IT systems as well as programming patterns
  – Assembly of solutions from design components at varying levels of abstraction
John Zachman – Architecture Framework

The Zachman Framework for Enterprise Architecture

The Enterprise Ontology

Version 2.0

© 1997-2011 John A. Zachman, all rights reserved. Zachman® and Zachman International® are registered trademarks of John A. Zachman. To request Permission Use of Copyright, please contact Zachman.com
John Zachman – Architecture Framework

• John designed the Zachman framework which serves as the fundamental ontology for IT architecture
  – Drawings to communicate to stakeholders at varying levels
  – Communicate the fundamental message for complex ideas at appropriate levels of detail and abstraction

• Key Concepts
  – Reification (making concepts real and concrete): Identification, Definition, Representation, Specification, Configuration, and Instantiation
  – Hierarchical layers of diagrams from “bubble drawings” to “shop drawings”
  – Result is a very flexible framework for purpose-driven design artifacts

• Application
  – IT architecture drawings and other artifacts can communicate most effectively when they address the interrogatives and reification
  – These designs map very nicely into concepts of service components and services to effectively build and assemble complex solutions
Enterprise IT 1.0: Systems of Record

The Next Stage of Enterprise IT: Systems of Engagement

“...systems designed to dramatically improve the productivity of middle tier knowledge workers. ... (they) enhance the ability of knowledge workers to quickly cooperate with each other in order to improve operating flexibility and customer engagement.”
Geoffrey Moore – Systems of Engagement

• Geoffrey is a well-known author and thought leader in information systems and business
  – Author of “Crossing the Chasm”
  – More recently developed the concepts of “Systems of Engagement”

• Key Concepts
  – IT is built upon interactions with core “Systems of Record”
  – Over time, how we engage (interface or interact) with these Systems of Record is via innovation with Systems of Engagement
  – These evolve – content, batch, online, client-server, browser, mobile, SMS, Twitter, automated BPM, etc. -- and are conceptually separate architectural components
  – Conceptually like the programming pattern of Model-View-Controller (MVC) at the enterprise architecture level

• Application
  – Core concept of separating persistence and business logic in Systems of Record from how they are engaged
  – Maps directly in SOA-based access to data, events, and business logic
Gartner – Pace Layering

How Do Other "Systems" in Industries Deal With Varying Rates of Change? Pace Layering?

- **Structure**: 30 to 300 Years
- **Skin**: 20 Years
- **Space Plan**: 3 to 30 Years
- **Services**: 7 to 15 Years
- **Stuff**: 1 Day to 1 Month
- **Site**: Eternal

*Note: For more on pace layers and shearing layers in building architecture, see "How Buildings Learn" (1994) by Stewart Brand*
A Pace-Layered View of Applications

- New Ideas
- Competitive Threats
- Systems of Innovation
- Unique Processes
- Systems of Differentiation
- Greater Efficiency
- Common Ideas
- Systems of Record

Gartner – Pace Layering
Gartner – Pace Layering

• Gartner has extensively developed and refined the architectural concept of “Pace Layering”

• Key Concepts
  – Buildings have a durable core of steel, brick, cement, and framing that last for decades
  – Over this core are layers that change much more frequently at varying paces.
  – For example, paint/wallpaper can change very frequently, wall materials change or are rearranged less frequently, floors and foundations infrequently.
  – Pace Layering has concepts in common with Moore’s Systems of Engagement versus Systems of Record

• Application
  – Architect systems to enable rapid evolution and change of layers that are expected and known to change rapidly
  – SOA concepts of loose coupling and standard interfaces work well to enable this separation of concerns
IBM’s SOMA spans the spectrum of SOA concerns
SOMA

- The IBM Service-Oriented Modeling and Architecture Method (SOMA) provides an effective architectural and design methodology for SOA

- Key Concepts
  - Service Identification
  - Service Specification
  - Service Realization
  - Service Implementation
  - Service Deployment

- Application
  - Focuses on the required activities and roles to create a SOA
  - Identify and describe services for design
  - Detail the services for implementation
  - Works well in top-down approaches using high-level tooling like ESBs
Open Group – TOGAF and SOA Reference Architecture
Open Group – TOGAF Models and ESB
Open Group – TOGAF and SOA Reference Architecture

- The Open Group maintains TOGAF (The Open Group Application Framework) as well as SOA Reference Architecture

- Key Concepts
  - Framework for creating and evaluating enterprise and SOA architecture
  - Comprehensive – TOGAF Version 9.1 is 700 pages!
  - TOGAF covers business, information, application, and technology architectures while SOA RA focuses on service solutions

- Application
  - Defines Architecture Building Blocks (ABBs) to specify and shape capabilities
  - Defines Solution Building Blocks (SBBs) representing components that are used to implement capabilities
  - Ties the various architectures together with vision, planning, and governance models
  - Identifies architectural decisions and roles in these architectures
Programming Patterns and Methodologies

• Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides – Gang of Four – Patterns
• Rebecca Wirfs-Brock – Responsibility-Driven Design
• Grady Booch, Ivar Jacobson, James Rumbaugh – UML
• Martin Fowler – Patterns of Enterprise Application Architecture

• Key Concepts
  – Core OO Concepts map directly into SOA and Service Component Design
  – Polymorphic
  – Encapsulation
  – Design methodologies

• Application
  – Abstraction of these concepts at higher levels of component design
  – These are generally patterns for programming, not for enterprise architecture
Assembly of Solutions
Wiring (Visual) versus Programming (Text)

Assembly Diagram

MAP
Moving from Programming to Assembly of Solutions

- Assembly of Solutions is by “Wiring” and not “Programming”
- Sub-assemblies are referred to as components
- Each component has a well-understood responsibility
- Components are at varying hierarchical levels of abstraction
- All successful physical ecosystems are via assembly via components – cars, houses, computers
- As are human systems, solution assembly is via services – realtors, stock brokers, lawyers, doctors, decorators.
- Many solutions merge a combination of physical components and human service components – restaurant, travel, hotel,
- Components have interfaces and are wired using integration tools
- These are essential to rapid exploitation of SOA
Design Principles
Design Principles

• Core Architecture Design Principles
• Prescriptive Architecture
• Component Layers
Core Architectural Design Principles

• Architecture and Design
  – Layers of abstraction with purposeful layering
    • E.g. TOGAF Process Flow layer over services
  – Loose Coupling and Separation of Concerns
    • E.g. BAM Message Emitters to separate monitoring
  – Hierarchical Decomposition
    • E.g. Component Assembly (Zachman)
  – Encapsulation (SOA)
    • E.g. Systems of Record/Systems of Engagement (Moore)
  – Evolution (Gartner)
    • E.g. Systems of Differentiation/Systems of Innovation
Prescriptive Architecture

• Standardize
• Reduce architectural decisions

Goals

- CIO Issues
  - Facilitate solution evolution
  - Manage technology lifecycle
  - Enable engagement
- Reduce Costs
- Consistency
- Focus efforts on business solution design vs. technology selection
- Reduce variability, focus energy, reduce maintenance and training
- Speed development cycle, enlarge pool of qualified people with needed skills
Component Layers

The SOA concepts of well-defined interfaces and encapsulated implementations extend over the whole hierarchy of IT abstractions.

- **Business Sub-Systems (Systems of Record)**
  - Integrated with services, ETL, and shared repositories
- **Services (Web or Queued Message-based)**
  - Ubiquitously invoked from all platforms
  - Transformed and orchestrated with WESB and BPM
- **SCA (Network or memory connections)**
  - Independently deployed and managed components
  - Wired in memory, and exposed as Web or JMS services
- **OSGi Bundles (Open Services Gateway initiative)**
  - Java components with distinct interfaces and co-existing version management
  - Coexisting versions are resolved and invoked from Java methods
- **Java Classes (and Code Snippets)**
  - Classical OO service patterns (Martin Fowler and Gang of Four)
Notations, Servers, Engines, Frameworks, and Languages
Notations and Design Documentation

For architects to communicate with business users and analysts, as well as developers, requires documentation that is suited for the audience, purpose, and level of abstraction.

This necessitates a combination of graphical, narrative, and tabular documentation.

Taking the lead from Zachman, differing packages of documentation are prescribed at various phases of the development lifecycle – conceptual, high-level, and detailed – with various purpose-driven diagrams.

One of the most important modern tools is the use of BPMN – Business Process Model and Notation – which carries through from business concepts to detailed design.

- BPMN shows collaborations and interactions between people and systems
- It is hierarchical to describe lower levels of details on separate pages
- BPMN Activities can be readily mapped to architectural components
BPMN provides an industry standard notation to describe human and system interactions and flows.
BPMN

BPMN is well documented and richly expressive
Servers, Engines, and Frameworks

- Purpose-designed servers (engines) are key to assembling SOA solutions. Examples:
  - JEE, messaging, BPM, rules, events, monitoring, reporting, modern batch, etc.

- Prescribing standard frameworks provides the construction and assembly code to build components within a JEE application server environment. Examples:
  - JPA, JSF, Wink, Axis, JAX, etc.

- Prescribing standard human and system interfaces provides consistency and loose coupling to consumers
Languages – DSLs and Visual Tools

We continue to move towards domain-specific languages (DSLs), replacing usage of general-purpose languages, like Java.

- Effective visual tools implement process, rules, user interfaces, mapping, services, and reporting.
- These are self-documenting and closely aligned with the problem domain, and thus can be understood, validated, and often authored by analysts.
- Java (and the JVM) is becoming the underlying generated machine language rather than the language in which developers program.
- The lingua franca of cross-platform UX is converging on HTML5, CSS3, and JavaScript libraries – bundled together with visual wireframing and implementation tools.
- Better scripting languages, with OO underpinnings, such as Jython and Groovy are replacing early languages like Perl.
- SOA services can readily connect components developed in these varied languages.
Summary: Prescriptive SOA Architecture

- Start with a prescriptive vision that mirrors common ontological concepts
  - Abstraction and Layered Hierarchy
  - Components and Assembly
  - Services with well-articulated responsibilities
  - Separation of Concerns and Design for Evolution

- At each level, make prescriptive choices to optimize resources
- Prescribe notations, tools, and languages appropriate to that abstraction.
  Examples:
  - Business Process – Process Engine, BPMN, and BPEL
  - Business Activity Monitoring – KPI, Dashboard, and Notifications
  - Persistence – Domain Model backed by JPA
  - Views – Web Model backed by JSF Dojo
  - Interfaces – SOA – Standard XML naming, definitions, and protocols
  - Reporting – Reporting tools (rather than writing code)
  - Infrastructure – Appliances where available
We love your Feedback!

• Don’t forget to submit your Impact session and speaker feedback! Your feedback is very important to us, we use it to improve our conference for you next year.

• Go to impactsmartsite.com from your mobile device

• From the Impact 2012 Online Conference Guide;
  – Select Agenda
  – Navigate to the session you want to give feedback on
  – Select the session or speaker feedback links
  – Submit your feedback

• Reminder:
  – This is session 2290
  – Speaker is Gary Gershon
Thank You

Includes Proprietary Information of Chubb & Son, a Division of Federal Insurance Company. All Rights Reserved.