Enterprise 2.0 Solutions with SOA

Mike Rosen
Chief Scientist
Wilton Consulting Group
Mike.Rosen@WiltonConsultingGroup.com

Enterprise 2.0
Solutions with SOA

Mike Rosen
Director, Enterprise Architecture
Cutter Consortium,
mrosen@cutter.com
Agenda

- What is Collaboration?
- Extending a Business Transaction
- Long Running Transactions
- Example Collaborative Business Transaction
- Enterprise Application Architecture
- SOA in Enterprise 2.0
- Enterprise 2.0 Framework
- Conclusion

What is Collaboration?

“Collaboration is a recursive process where two or more people or businesses work together toward an intersection of common goals by sharing knowledge, learning, and building consensus. Structured methods of collaboration encourage introspection of behavior and communication. These methods specifically aim to increase the success of teams as they engage in collaborative problem solving.

- Wikipedia

Types of Interaction
- Conversational
- Transactional
- Collaborative
**Collaboration Goals**

- Communications
- Publishing / feedback / information exchange
- Building community
- Managing knowledge
- History and tracking
- Joint contribution to a work product

**Collaboration’s Time-Space Matrix**

<table>
<thead>
<tr>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face to Face</td>
<td>Synch Distributed</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
</tr>
<tr>
<td>- F2F Meeting</td>
<td></td>
</tr>
<tr>
<td>- Presentation</td>
<td></td>
</tr>
<tr>
<td>- Whiteboard</td>
<td></td>
</tr>
<tr>
<td>- Water Cooler</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynch Distributed</td>
</tr>
<tr>
<td></td>
<td>Email</td>
</tr>
<tr>
<td></td>
<td>Voice mail</td>
</tr>
<tr>
<td></td>
<td>Discussion forums</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice the "unstructured" nature of these items?

Do these enable your business needs?

Or do they ‘merely’ facilitate them?
Business Facilitator Versus Enabler

Facilitator

- Processes
  - Research product feature
  - Review interview results
  - Compare flight alternatives
  - Lookup medicine side-effects

- Modes
  - Meetings, water cooler
  - White board, presentation
  - Email and Vmail
  - IM & social networks

- Characteristics
  - Highly collaborative
  - Data integrity (synchronization) low priority

Enablers

- Processes
  - Place order
  - Hire employee
  - Submit business expense
  - Authorize medication

- Modes
  - Transactions
  - Databases
  - Business processes
  - Biz Apps: ERP, CRM

- Characteristics
  - Minimally collaborative
  - High data integrity

Is it possible to integrate these? Do we even want to?

Collaborative Insurance Claim Example

- Insured
  - Route to M.D.
  - Route to Lab
  - Add’l information needed
  - Requested information provided

- Insurance Co. Server

- Insurance Company
  - Route to Insurance Company
  - Insured

- Adjuster
  - Submit for payment
Business Versus Technical Transaction

- For traditional transactional system
  - Business transaction is long
  - Technical transaction (‘Submit for payment’) is short

...instead of

Is this atypical or the norm?

Collaboration

- Work is collaborative; computer apps are not
The Pitfalls

- Lack of standard procedure
- Only spotty help from automation
- Out-of-channel conclusions have to be manually transferred to transactional application
- Communication off the record
  - No audit trail
  - No accountability

The Ideal

- Collaboration takes place within application context
The Advantages

- What you’re collaborating about is right in front of you
- Interim conclusions kept for future reference
- Record of who did what when
- System becomes a valuable “collaborator”
  - Validation errors
  - Soft and hard errors
  - Work routed intelligently

State-of-the-Practice

- Today collaborative transactions are available from:
  - Hand-crafted applications such as CRM, Call Center, ERP
  - Expensive to build
  - Hard-wired to the solution

- In today’s technology, for other enterprise applications
  - Not product-supplied solutions
  - But architecture-based solutions
    - Best-practices
    - Special process assembly tier in architectural profile
    - Combination of BPM concepts and transaction principles
Workshop – Where are You Today?

- Form teams
- Think of an app in your organization that is non-collaborative:
  - personal
  - two-party
  - short duration
- Is this app non-collaborative
  - because of design
  - or because of implementation difficulties?
- What you are doing in your enterprises today with collaboration?
  - What are the impediments to collaboration?
  - What could you be doing? What benefits would that convey?
- Choose one or two scenarios from the group.
- Sketch out some new collaborative possibilities.

Transaction Principles and Long-runningness

- Transactions (aka short-running transactions) are at the heart of
  - Database systems
  - TP monitors
  - CORBA OTS and EJB JTS and others...
  - Architected via X/OPEN DTP standards
- Short-running transaction are computing’s great success story:
  - Work year-after-year without fuss
  - “Transparent” technology == success
- But,…applying to Time-Space matrix, Time rears its ugly head…
Time-Space Matrix Applied to Transactions

The problem dimension is…

Time

To Analyze LRT, Investigate ACID Properties

- Short-running transactions
  - Are successful because of well-understood ACID properties

- Long-running transactional integrity should match that of short-running
  - Consider impact of removing A, I, D properties (C less important)
  - How can these missing properties be compensated for?

- Since mid-80s ... ACID test of Transactions:
  - Atomicity
  - Consistency
  - Isolation
  - Durability
**Atomicity: Integrity via Two-Phase Commit**

- Two-phase commit achieves distributed consensus
  - Phase 1: Prepare
  - Phase 2: Commit
- One participant designated the coordinator
- Coordinator signals participants to prepare, then commit
- Like the wedding ceremony
- Window of vulnerability

---

**Isolation: The Key is Locking**

- Isolation (serializability) via locking shared resources
- Maintained until Commit or Abort
- Technologies
  - Read / shared
  - Pessimistic: Write / exclusive (state-of-the-practice)
  - Optimistic: Write / shared -- then test for conflict (state-of-the-art)
  - Deadlock
    - Detection
    - Avoidance (canonical lock ordering)
    - Resolution (timeout, selective abort)
**Durability: Hardening via Logging**

- Records information to reconstruct persistent data
- Guarantees durability with failures from:
  - Transaction
  - Server
  - Media
- Technologies
  - Write-ahead logging
    - Buffering of logging (avoid re-write)
    - Updates written sequentially to disk
    - Rapid restart
  - Common log

**Consistency: The Goal of Transaction Processing**

- Consistency =
  - Atomicity, plus
  - Isolation, plus
  - Durability, plus
  - Correctness of application
    - not an infrastructure responsibility
Long-running Processes with no ACID

- For a long-running process to achieve the integrity of short-running process it must match the results of ACID using a variety of alternative measures.

<table>
<thead>
<tr>
<th>SRT Property</th>
<th>Provides</th>
<th>Is Replaced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomicity</td>
<td>Coherent results</td>
<td>Compensation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If a process step fails, do what’s necessary to correct it.</td>
</tr>
<tr>
<td>Isolation</td>
<td>Non-interfered results</td>
<td>Optimistic checking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Periodically checks to ensure assumptions are unchanged.</td>
</tr>
<tr>
<td>Durability</td>
<td>Permanent results</td>
<td>Full process state persistence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reliably preserve all in-flight process state (plus context)</td>
</tr>
</tbody>
</table>

Approaches for organizing complex processes

- **Orchestration**
  - **Top-down:** Single master controls all aspects of a complex process.

- **Choreography**
  - **Bottom-up:** Each autonomous element controls its own agenda.
Orchestration overview

- **Concept**
  - The process is defined by a single master process definition which acts as a master-controller of ‘dumb’ data cargo

- **Design-time**
  - Process definition is defined by script, table or graph
  - Process definition is kept separate from application definition
  - Governing definition standard: BPMN swimlane diagram

- **Execution-time**
  - Governing execution standard: BPEL
  - Process instantiated when listener hears predefined start event(s)
  - As process proceeds, execution engine invokes specified actions
  - In-flight process state is persisted in a process database when it’s not sleeping (99% of the time)

---

BPMN Used for Non-personal Roles

![BPMN Diagram](image-url)
Long-running Behavior in Typical Orchestration

- Atomicity replacement
  - Transaction boundaries and compensation actions are both modeled as first-class elements in BPMN
  - Notice that this isn’t automatic and transparent as in SRT (nothing is)

- Isolation replacement
  - None…except exhaustive explicit checking of assumption data

- Durability replacement
  - Plus: All process data and application data is persisted while inflight
  - Minus: Application data is completely fragmented and can’t be separately reassembled
  - Minus: No collaboration data is persisted without special procedures

Orchestration Summary

- Top-down
  - Single master controls of all aspects of a complex process.
  - Supports holistic possibly graphical view of the activity.
  - Simpler to start; harder to scale.
  - Function follows form.
  - State-of-the-practice.
Choreography Overview

- Concept
  - Process state is built into the application object itself (the “case” or “work-unit”)

- Design-time
  - Process definition is defined by script, table or graph
  - Finer-grained than orchestration because only this object’s workflow is defined

- Execution-time
  - Process instantiated when application object is defined
  - As workflow needs arise, various roles are summoned to attend to application object
  - Full application object state (plus process state) is persisted when the object is passivated

Choreography in Software

- Two principal approaches:
  - Message-examination
    - Defines behaviors by exhaustively capturing the message contracts between collaborating parties (WS-CDL)
    - Used for B2B applications: only need to specify the interchange definitions
  - Work-component configuration
    - Define behavior of individual work components and let the process behavior emerge as it evolves
    - Not “workflow” but “work that flows”
Long-running Behavior in ‘Typical’ Choreography

- Atomicity replacement
  - Compensation actions are a part of the process scripts
  - Notice that this isn’t automatic and transparent as in SRT (nothing is)

- Isolation replacement
  - Assumption failure detection is automatic
  - Assumption recovery can be incorporated into the need processing

- Durability replacement
  - Full durability of process, application, and collaboration data

Choreography Summary

**Bottom up**

- Each autonomous element controls its own agenda
- Total process behavior “emerges” from the working of its parts
- More complex to start; easier to scale
- Form follows function
- State-of-the-art.
When to Use

- **Orchestration**
  1. Given product sets, this is your default choice
  2. For composed services
  3. Where transaction semantics can handled by ‘compensation’ alone
  4. For relatively static process definitions
  5. Where a graphical process definition is desirable

- **Choreography**
  1. Where processes may scale to a very high number of steps
  2. Where opacity of process details is desired among partners (B2B)
  3. Where different process partners may require their own process tweaks (also B2B)
  4. Where process is highly dynamic or goal-seeking

Summary

- New way of thinking about system’s role relative to human role.

- System helps the business process, doesn’t just record the results.

- Enables true collaboration.
Example Business Transaction

'Out of Band' Interactions

Applicant

Phone

Email

Documents

Business Transaction

HR

ERP

New Employee

Complete, collaborative transaction

Collaborative Business Transaction

Job Posting

Create Job Posting

Manager

Video Conference

Schedule Interviews

Interviewer / SME

Scheduling

Addressing

Document Management System

Resume

Iterative communications

Email

Voice

Applicant

Manage Resume

HR

Initiate 'New Employee'
Enterprise Goals of Application Architecture

- Support single view
  - Customer Centric
  - Integrated
- Rich collaborative environment
- Common, consistent look and feel
- Flexibility and alignment
- Lower cost and faster time to market
- Reduced maintenance costs
- Auditing and compliance
- Simplified operations and management
- Standardized platforms
- QoS - Scalability, performance, reliability, security …

Sample Enterprise Application

- Marketing
- Product Catalogue
- Inventory
- Business Rules
- Billing
- Shipping
- CRM
**“Conceptual Architecture”**

- Customers
- Employees
- Suppliers
- Other Partners

- Personalization
- Security
- Workflow

- Web Application and Integration platform
  - Content Management
  - Knowledge Management
  - Rules

- Applications and resources
  - ERP / Billing
  - CRM
  - Product Catalog
  - Inventory / Shipping
  - Legacy Systems

---

**N-tier Application Architecture**

- **user**
  - Presentation
  - RIA Controller

- **workspace**
  - Application Coordinator
  - Session
  - User Profile

- **enterprise**
  - Composite Business Services
  - Business Entity Service

- **resource**
  - Application Adapter
  - Resource Adapter

- Services
  - Security Service
  - Entitlement Service
  - Location Service
  - Identity Service
  - Xformation Service
  - Routing Service
  - Configuration Service
  - Logging Service

- Infrastructure
  - Application Services Platform

---

© 2009 Michael Rosen
Architectural Elements

- The functional and structural components (and their interfaces) from which the system is composed
- The architectural elements are defined to provide for:
  - Distribution
  - Scalability
  - Technology independence
  - Device independence
  - Application integration
  - Future enhancements and migrations
- Not all applications will have all elements
- Elements map to different implementation types depending on the technology
- Not all elements map to distributed components

AJAX Based User Interfaces
Web Services (B2B) Architecture

Combined Architecture
Application Architecture

- Describes how to build your applications to consistently meet enterprise goals

- Includes
  - How to use technical architecture
  - Fundamental product elements
  - Common patterns
  - User model
  - Application frameworks
  - How to configure and manage applications

Architectural Foundations

<table>
<thead>
<tr>
<th>Layers</th>
<th>Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>user</td>
</tr>
<tr>
<td></td>
<td>workspace</td>
</tr>
<tr>
<td></td>
<td>enterprise</td>
</tr>
<tr>
<td></td>
<td>resource</td>
</tr>
<tr>
<td>presentation and device independence</td>
<td>User session and application experience</td>
</tr>
<tr>
<td>Application level business logic</td>
<td>Business processes and entities</td>
</tr>
<tr>
<td>Common utility functions applied across tiers</td>
<td>Shared enterprise resources</td>
</tr>
<tr>
<td>Underlying technical and communication capabilities</td>
<td></td>
</tr>
</tbody>
</table>
Tiers and Layers

- Tiers – Logical distribution of functionality
  - Each tier has roles and responsibility
  - Physical distribution
  - Scalability
  - Reuse

- Layers – Logical separation of logic
  - Spans tiers
  - Separates business logic from services from infrastructure
  - Insulates from technology evolution

Responsibilities

- Each logical tier has distinct areas of responsibilities

- Within each tier, architectural elements have distinct roles (areas of responsibility)

- Domains separate individual users from shared resources

- Roles and responsibilities are delineated by well defined interfaces

- Division of responsibility is critical to “plug-and-play” capability of components and services
Typical Layered SOA Architecture

- Business Processes
- Interfaces defined by enterprise model
- Business Services
- Integration Services
- Operational Resources

Tiered Service Architectures

- Presentation
- Application Logic
- New E 2.0 Services
- Traditional SOA Services

3 Tiered

n - Tiered
Basic Collaboration Architecture

Collaboration
Scheduling, File Sharing, Document Management, Wiki, Social Networks

Communications
IM, Email, Voice, Video, Publishing, Blogs

Infrastructure
Networks, Mobility, Presence, Identity, Security, Application Delivery, Policy

Tiered, Collaboration Application Architecture

user
workspace
enterprise
resource

Presentation
Scheduling
Application Coordinator
Publication
Session
User Profile

Document Management
Community Management
Business Process
Business Entity

Legacy System
ERP

Application Platform

services
infrastructure

Security and Entitlement Services
Policy Service
Presence and Location Services
Voice and Video Services
Conferencing Service
Transformation Service
Logging Service
Evolution of Architectural Styles

- Monolith
- Structured
- Client / Server
- Objects
- 3-Tier
- Components
- n-Tier
- SOA
- Enterprise 2.0

Conclusion

- Collaborative transactions
  - Creates a new breed of business-process-driven systems
  - Requires re-visioned architectural insight and practice

- SOA
  - New type of service provides E2.0 application functions
  - Has a different role than traditional SOA business or utility services

- Three-tier architecture doesn’t get you there
  - A ‘workspace tier’ provides foothold for longer-running user sessions
  - A collaboration framework provides reusable support for multi-role interactions
Thank You!