Art of Integration

Service Oriented Architectures (SOA) and the Enterprise Services Bus (ESB)

Origination: Chris Huff, TWC
Current Philosophy – The Theme Continues

Quality Appliances
Donations to our nonprofit training business are tax deductible.

OPEN
Wednesday – Saturday
10 a.m. to 6 p.m.
Common Theme?

What’s missing:
- Ubiquitous Standards
- Open standards
- Open community

Goal → Flexibility and Reuse

What is still missing:
- Consistency
- Discipline
Service Oriented Architectures

How do I deliver business flexibility?

Can Information Become a Service?
Current Integration Philosophy

- EAI, ESB, SOA, ESA, EIEIO...
- BPM, Workflow

ESB is very similar to that of "Enterprise Application Integration" (EAI) tools, apart from three significant terms:

- Web services, ubiquitous and lightweight. ¹

“The emergence of the ESB Concept is closely linked with the lasting trends that have been slowly transforming the EAI market for the last few years: standardization of infrastructures with

- J2EE
- Microsoft .NET
- Web services
- the redistribution of roles of integration software component vendors.” ¹

¹ - http://searchwebservices.techtarget.com/tip/1289483,sid26_gci913058,00.html
Recurring Theme – Abstraction layers

- BI Apps
- Portals
- B2B Networks
- Composite Apps

**EAI** (application integration)

**BPM** (process management)

**ESB** (services integration)

DATA SERVICES
(data abstraction layer)

- Files
- Message Queue
- XML
- EDI
- Unstructured Data
- Application Databases
- Mainframe
- Data Warehouse
Recurring Theme – Abstraction layers
Why make the change?

- To deliver applications faster!

“Shifts focus to application assembly rather than implementation details”

Benefits to SOA
(ROB HIGH, IBM CHIEF ARCH SOA)
1. Customer Satisfaction
2. Business Operations Efficiency
3. Process Efficiency
4. Reuse
What’s the Return?

Significant cost savings will not happen **without** discipline

Big Payoffs at the End
- Dynamic application development (composites/mashups)
- Dynamic Business Processes
- Organizational Speed/responsiveness
- Customer Service

Incremental ROI along the way
- Granular Configurable Services
- Graphical development
- Extensibility (regression testing)
- Loose Coupling (upgrades, flexibility)
- Consistency, Standards
Document the Architecture
Build a Conceptual View

EAI / ESB

SAP Hub

Stores

MQ Hub
Channel Workload Mgmt.

EAI / ESB

SOSI/MM Hub

SAP CTI
SAP WFM
SAP Core Retail

Inv
PO
Sales Audit (FIN1), Credit (FIN2)

Legacy AP

Labor Mgmt

Tool Rental

PC (FastTax, TMS, CorpTax)

SVR1

PC (Call Center, FC)

All Other Partners and Vendors

Citi
Big Hammer
Trane

Affiliate/International

HD Supply

Mexico

Stores

Store nnn

Other

Call Box

ISPN

PSP

Base Int

WPC

WMS

UCM

TMS

WCS

MQ Gateway

B2B Gateway

Firewall

Important

Build a Conceptual View
Rally Projects Around Your Vision

- Rollout Enterprise Service Repository
- Service Governance

Today

- Loosely Coupled (LC) Applications
  - Some LC Services
  - Initial services/reuse
  - Messaging Std
  - Some XML Stds
  - Interface Visibility

Mar 2009

- Capability for Service Oriented Processes
  - MSR Go-live
    - new services
  - July 2009

- ESB

- BPM Go-live
  - 4 new services

May 2009

- Prepare Governance Standards
- Select ESR
- Establish SOA Governance Panel

- Service enabled Processes reused
  - New apps built from service composition & Mashups

- New Integration
  - Capability live in all stores
  - Sept 2009

- SC Visibility
  - new services

- Begin dev on ES API – define THD standard
- Begin dev on highly reused service list

- User Service Discovery Tool Live
- Nov 2009

- Next phase BPM Services Live

- Jan 2010

- Dynamic Services
- Business becomes SOA
  - Driver?
- Mar 2010

- Dynamic Service Discovery
- Nov 2010

- Store Integration Standards
- Managed Services
- Nov 2011

THD Standards Adoption

Industry Standards Adoption

Technology Standards Adoption
Implement Standards

- Know and plan your layers of reuse: code, modules/objects and services
- Certify that all service interfaces are extensible
- Spend time up front on API/Interface design
- Develop Middleware/SOA Cost Model
- Ask COTS package providers to supply “out-of-the-box” industry standard formats to exchange data
- Don’t get caught in the Web Service guise of loose coupling, broker synchronous calls unless they are true Web Services
Challenges In the Enterprise
Principles Should Chart the Course

Forget the acronyms and stick to the principles

- Low total cost of ownership
- Repeatability
- Ease of maintenance
- Object Reuse
- Reliable performance
- Rapid development

- Adherence to architectural principles
  - Design that is flexible and versatile (abstraction between applications)
  - Application Decoupling (Loose coupling reduces assumptions two parties make about each other, while tightly coupled architectures tend to result in brittle, hard-to-maintain, and poorly scalable solutions)
  - Ease in importing application metadata

- Highly Scalable, Secure and Reliable
  - High volume processing capability
  - Scalability

Always ask yourself, “why is this better than point-to-point?”

- If it is not cheaper, then don’t do it
- If it’s not providing a desired end-state, then don’t do it (be careful with this one)
- If it can’t support your volumes and growth, then don’t do it
# Graphical Mapping — Cost Savings

## Development Estimates

<table>
<thead>
<tr>
<th>Developer Skill Level</th>
<th>Map Complexity</th>
<th>56% less</th>
<th>33-43% less</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic – Small to medium size data structure with simplistic mapping logic</td>
<td>35 hrs</td>
<td>80 hrs</td>
</tr>
<tr>
<td></td>
<td>Average – medium to large size data structure with a moderate amount of logic and complexity</td>
<td>80 hrs</td>
<td>120 hrs</td>
</tr>
<tr>
<td></td>
<td>Difficult – large data structures, advanced logic, multiple inputs and outputs and possibly, disparate data cardinality</td>
<td>160 hrs</td>
<td>240 hrs</td>
</tr>
<tr>
<td></td>
<td>*Percent of hours to add if type trees cannot be imported</td>
<td>50%</td>
<td>50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Novice</th>
<th>Beginner (a few maps)</th>
<th>Proficient</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>35 hrs</td>
<td>8 hr s</td>
<td>8 hrs</td>
<td>4 hrs</td>
</tr>
<tr>
<td>80 hrs</td>
<td>56 hrs</td>
<td>24 hrs</td>
<td>16 hrs</td>
</tr>
<tr>
<td>160 hrs</td>
<td>120 hrs</td>
<td>80 hrs</td>
<td>40 hrs</td>
</tr>
</tbody>
</table>

**Typical Interface estimates (Days)**

<table>
<thead>
<tr>
<th>Low</th>
<th>Med</th>
<th>High</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>20</td>
<td>30</td>
</tr>
</tbody>
</table>

** Estimates provided by TCS**

**Interface Development (batch)**

<table>
<thead>
<tr>
<th>Low</th>
<th>Med</th>
<th>High</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>25</td>
<td>35</td>
</tr>
</tbody>
</table>

**Interface Development (real time)**

<table>
<thead>
<tr>
<th>Low</th>
<th>Med</th>
<th>High</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>25</td>
<td>35</td>
</tr>
</tbody>
</table>
Consider Performance

We needed 250-300/sec Invoice record to Invoice IDoc

Figure 12: Stress Test Results for Converting Numerous EDI Formats to IDocs
Rationalize the Cost Model

- Are your tools ready for XML (extended character sets, parsing, etc. These can become costly)?
- Are you paying a premium for middleware development?
- Do your middleware tools provide rapid (or graphical) development?
- Consider maintenance cost (e.g. scripts and complex logic)