

Mashups for Composite Enterprise Applications

SYSTEMATIC THOUGHT LEADERSHIP FOR INNOVATIVE BUSINESS



Shel Finkelstein and Ümit Yalcinalp
SAP Research, Palo Alto
shel.finkelstein@sap.com
umit.yalcinalp@sap.com

- Who are We?
- Client is Becoming a Full-Fledged Tier
- Examples of Composition
 - Mashups, Composite Apps, and Service Platforms
- Industry Trends
- Enterprise Service Composition Platform
- Recent SAP Research Work
- Taxonomy for Composite Apps/Mashups
- Conclusions

- Researchers from SAP Palo Alto Research Center
 - Colleagues include Rainer Brendle, Tilman Giese, Ralf Güldemeister, Rama Gurram, Anne Hardy, Jan Schulz-Hofen, Brian Mo
- Our Mission: To research and experiment with emerging trends and new technologies that simplify creation, delivery and execution of enterprise applications
 - Web technologies, dynamic languages and programming environments
- This talk is about trends & experiments, not product directions

This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material.

- Client memory and processing power is impressive
 - For example, Smartphones may have 128MB RAM and 620MHz CPU
- Client can handle more than presentation
 - Browser is becoming an app platform with additional services and plug-ins
 - Plug-in as a Platform (PaaS)
 - Dynamic language libraries enable logic to be executed on the client (e.g., Microsoft Silverlight)
 - Controls can be deployed in standalone runtimes or in browsers (e.g., ADOBE AIR)
 - Rich controls can utilize client capabilities (video, sounds, etc.) in applications
 - Local data cache can be managed by client apps (e.g., Google Gears)
 - Capable of extending and managing data and display characteristics (via JavaScript/AJAX and proprietary APIs)

■ Client/Server in new clothes?

- Backend servers provide services that are primarily data-oriented
- Middle tier handles integration, transformation, dispatch and connectivity
- Client handles presentation
- Cross-Tier Aspects
 - Composition
 - Execution of Logic
 - Events/Messages

■ What are the implications ...

- ... for development, assembly and execution of Enterprise Applications?
- ...particularly for composite applications?

- Components and Composition
 - Must support flexible reuse
 - Preferably described by metadata
 - Fast development, composition and change
 - Supply contracts for development, composition, and deployment
 - Different skills/roles for different tasks
 - Provide simple syntax to compose multiple components into a composite
 - Enable understanding composite apps based on:
 - Understanding semantics of individual components
 - Understanding composition operations
- Examples:
 - Unix pipes, Yahoo pipes
 - Relational queries
 - MTS, EJB and SCA frameworks
 - Technologies for service composition and orchestration

- Mashups Today
- Composite Applications
- Service Platforms and Software as a Service (SaaS)

- **Achieves: Data aggregation**
 - Client side controls the aggregation (Google Maps API)
 - Client side relies on data-centric services from the server (get, update, delete, insert, replicate....)
 - Client devices perform asynchronous interactions (AJAX technologies)
 - Client devices side may utilize application specific data caches (SQLite, Google Gears)
- **Utilizes: Lightweight rapid development style**
 - Scripting and dynamic languages (JavaScript, Ruby)
 - Component metadata
 - Popular environments such as Yahoo Pipes
- **Succeeds:** Delivers the “Goals for Components and Compositional Frameworks”, described on Slide 6
- Compare [*Jonathan Marsh*]

- **Established:** Controlled by the client, with 2 distinct styles
 - Client side components and APIs (Google Maps)
 - Development Model: Use APIs to supply client side components with server-side data
 - Execution Model: Run components on client; fetch (async pre-fetch) data from server
 - Client side components and events (OpenAjax Hub, SAP Research Enterprise Web Widget Framework, IBM QEDWiki)
 - Development Model: Compose using events and metadata
 - Execution Model: Run components connected by pub/sub events
- **Emerging:** Server-centric composition
 - Use public APIs and interfaces to link services (WS/data-centric interfaces) to create a new service (WSO2/JackBe) in server
 - Server-centric approach makes mashups resemble traditional composite applications

- Focus in today's mashups is data-centric integration
- Data-centric integration doesn't require REST, but it aligns well with REST
 - Backend resources identified by URLs provide RESTful data services
 - RESTful approach enables efficient development and management of compositional apps
 - Development: Easy and intuitive programming model (avoids side-effects)
 - Management: Great for scalability, load-balancing, availability

- Service Integration is at the server [*SAP NetWeaver CE*]
 - Data/data feed aggregation is still important
 - Can compose new services using interfaces and metadata from existing services
 - Lightweight development still valuable but not critical
 - Enterprise Service Bus as one integration approach
 - Integrated services may be within same org or at a remote company
 - Runtime protocol bindings may be managed by frameworks
- SOA-ready services aren't enough for composing enterprise suites
 - Data integration, data management, transaction management, process integration, service level agreements, etc.
 - Replication, distribution and eventual convergence [*Pat Helland*]

- Service Platforms
 - Hosted services enable composition based on:
 - Generic services provided by the host; “Infrastructure as a Service”
 - Custom services supplied by individual tenants (hosted or non-hosted)
 - Multi-tenancy issues for SaaS include:
 - Virtualization: Sharing securely, load balancing, extensibility
 - Coupling: Coarse-granular services, service levels, context management
 - Encapsulated implementation offers migration advantages
 - ... but doesn't completely eliminate software versioning issues
- Which applications and services are suitable for hosted environments?
 - Amazon, Google, salesforce.com, SAP Business ByDesign, ...

■ **Metadata** enables composition, management and flexibility

■ **Component Descriptions**

- Interfaces
- Events published/subscribed to
- Backend data sources
- Cross-tier deployment capabilities
- Requirements for composition

■ **Policies**

- Authorization
- Protocol bindings
- Composition policies
- Cross-tier deployment requirements
- Quality of Service

■ **Composite application metadata**

- Description of composite services
- Description of composite data
- Events/message flows
- Derived policies and imposed policies for composites

■ **Examples: Component, Assembly and Deployment Descriptors for SCA and JEE; Widget Frameworks**

■ “Stateless” middle tiers

- Middle tier mediates service access, protocols, data representation, identity and other security
- Can also handle composition and intermediation (broker, mediator, gateway, aggregator, etc.) [*Alistair Barros*]
- Caching mainly as a performance optimization

■ End-to-end cross-tier deployment and optimization

- Leverages declarative metadata describing logic, data, components and compositions
- Enables different execution model for different client capabilities
- Manual optimization → (semi-)automatic optimization?



- Emerging Next Generation research platform addressing multiple aspects of cross-tier compositional apps
 - User experience and presentation
 - Data and service management
 - Business logic
 - Events and messages
 - Backend integration
- Utilizes dynamic languages and metadata, enabling flexible optimization
 - Deployment of compositional app can depend on configuration
 - Specification based on loosely-coupled components with constraints
 - Optimization determines “good” deployments and execution plans based on metadata
 - Entire lifecycle needs to be considered
 - Development, composition, execution, management, change

Learning from the past, guiding the present, inventing the future

- Prototype from Client Perspective

- User Interface
- Architecture

- Lessons Learned

- Taxonomy for Composite Apps/Mashups
- Conclusions



Repository & Metadata based Management

Composite App

Event Hub for Composites

Widget Repository

- Widgets
 - Uncategorized
 - BijouX
 - Event Hub Debugger
 - Widget Editor
 - Widget Repository
 - Theme Switcher
 - ContactManagement
 - Contact Browser
 - Contact Search Quick
 - Games
 - PacMan v2.6

Contact Search Quick

query:

Topic:

Publish:

Yalc

Event Hub Debugger

Topic	Data
bijoux.edit_saplet	"CMContactBrowser"
bijoux.contact_selected	"Jensen"
bijoux.contact_selected	"Yalc"
bijoux.edit_saplet	"CMSearchQuick"

Contact Browser

First Name	Last Name	Email	Telephone
Serhat	Yalcin		
Umit	Yalcinalp	umit.yalcinalp@	

Sort Ascending / Sort Descending / Columns

Widget Editor

```

New Open Save Import Versions

<widget xmlns="http://openajax.org/widget"
name="CMSearchQuick"
scope="CMSearchQuick"
category="ContactManagement"
height="40"
scalable="false"
title="Contact Search Quick">
<properties>
  <property name="query"
topic="bijoux.contact_selected" publish="true"
/>
</properties>
<content mode="view" type="frame">
  <![CDATA[
  <script><!--
    CMSearchQuick = function() {
      this.searchField = new
Ext.form.TextField({
  </script>
  </![CDATA[
  </content>
  </widget>
  
```

Third Party Apps

1. Contact Dev Team

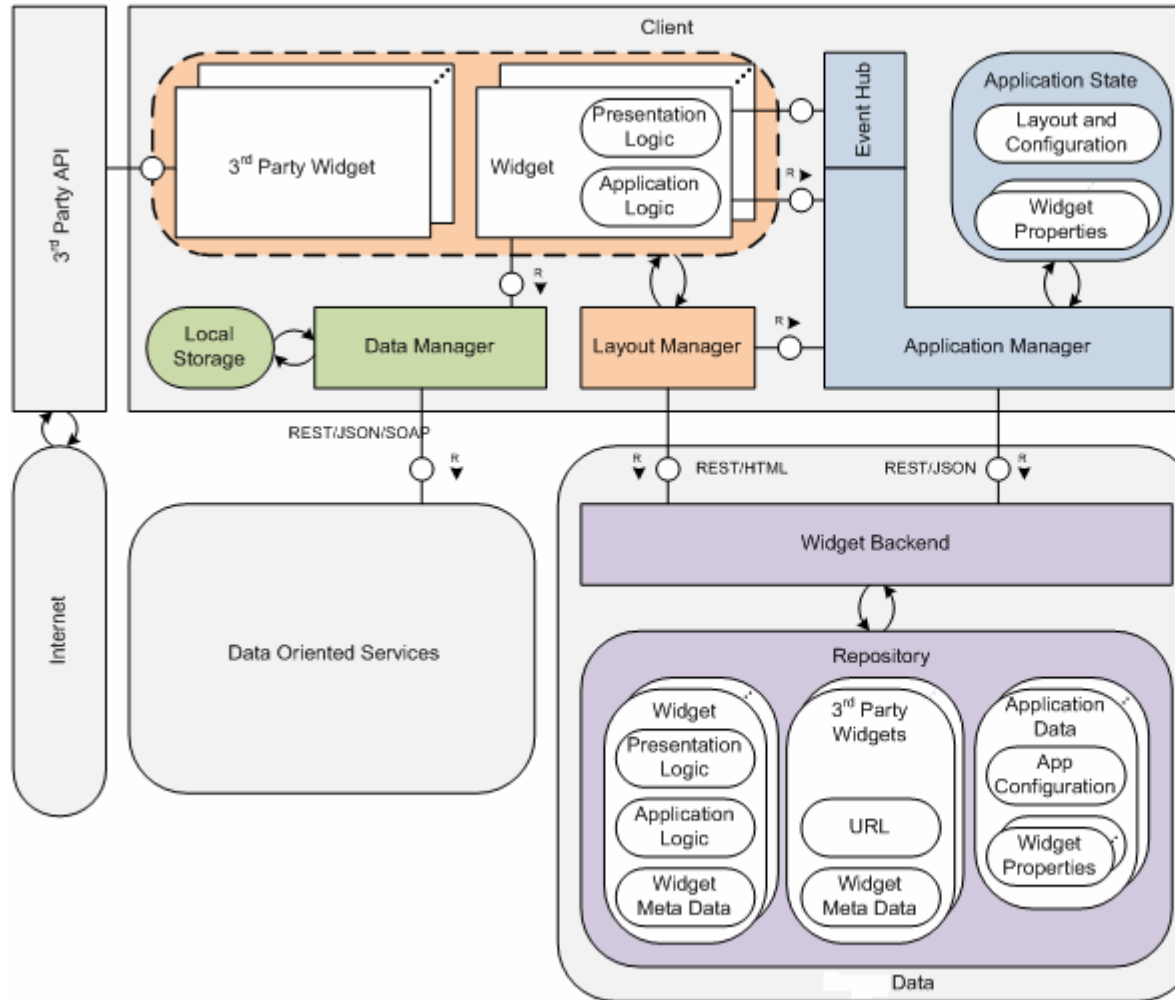
2. Prepare Talk

3. Type new task here

WeatherRadar

Click on map for local weather forecast

Map Sat Radar



Compare [Charlton Barreto]

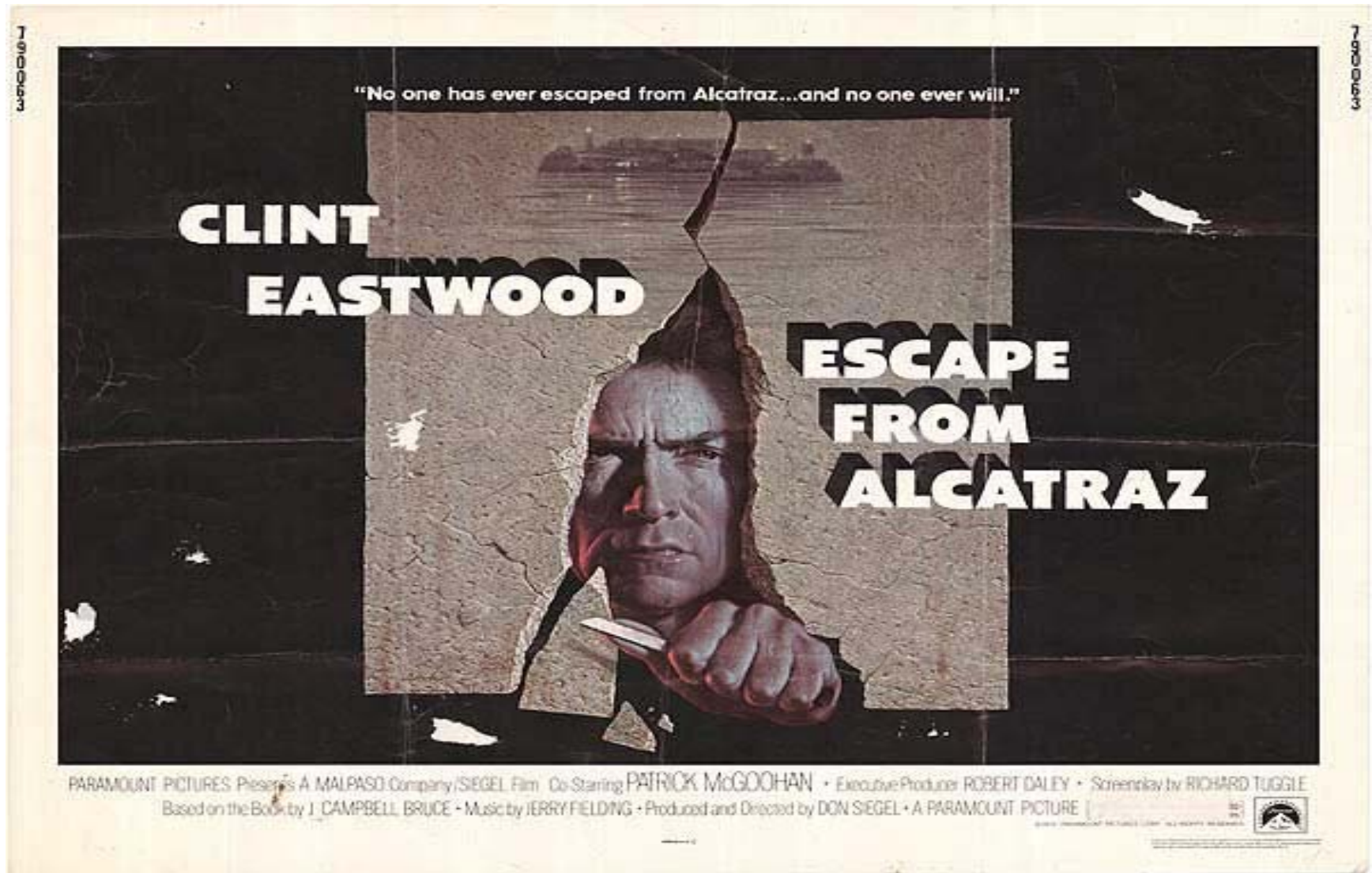
Taxonomy for Composite Apps/Mashups

A Spectrum with Multiple Good Answers



1. **Deployment:** Client; Servers
2. **Development:** Dynamic and lightweight; static and heavyweight
3. **Presentation:** Rich Internet Applications; rigid UIs
4. **Connectivity:** Must be connected; (occasionally) disconnected
5. **Integration Approach:** Data-oriented; Service-oriented
6. **Component Composition:** Events; APIs; compositional operations
7. **Programming Model:** Declarative metadata; imperative programming
8. **State Management:** RESTful; RESTless
9. **Data Management:** Loose consistency; tight consistency
 - Caching, async prefetch, versioning
10. **Service Coupling:** Loosely-coupled; tightly-coupled
11. **Process Integration:** Isolated; augmented; fully integrated
12. **Execution Plan:** Fixed; flexibly optimized based on system characteristics
13. **Service Delivery:** Packaged Software; Software as a Service

- For Composite apps, start from **What** you want to do, not **How** to do it
- Composite applications and mashups share a common taxonomy
 - Concepts are growing past adolescence
 - Clients such as browsers are a full-fledged tier
 - Dynamic languages enable lightweight composition
 - Component and composition frameworks should be metadata-driven
- End-to-end composite apps, deployed for cross-tier execution, are the next wave
 - Principles in SAP Research's Enterprise Service Composition Platform can help drive this
 - Taxonomy clarifies functionalities and tradeoffs
 - Enterprise qualities remain critical
 - Security, integrity, scalability, performance, availability, manageability, ...
 - Cross-tier optimization is an exciting future direction





- SAP Netweaver CE (Composition Environment)
<https://www.sdn.sap.com/irj/sdn/nw-ce>
- Sanjay Patil: *SCA Programming for the Enterprise Service Bus*
Presentation at this symposium
- Jonathan Marsh: *Mashup: Noun or Verb?*
<http://wso2.org/repos/wso2/people/jonathan/Mashup%20Noun%20or%20Verb.pdf>
- Charlton Barreto : *Web 20-20 Architecture for the New Internet*
<http://charltonb.typepad.com/talks/120407-cbb-web2020/Web2020TheNewInternet.pdf>
- Pat Helland:: *The Irresistible Forces Meet the Moveable Objects*
<http://blogs.msdn.com/pathelland/attachment/7082107.aspx>
- Alistair Barros: *The Rise of Web Service Ecosystems*
<http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/it/&toc=comp/mags/it/2006/05/f5toc.xml&DOI=10.1109/MITP.2006.123>



Shel Finkelstein and Ümit Yalcinalp

SAP Research, Palo Alto

shel.finkelstein@sap.com

umit.yalcinalp@sap.com