Leveraging the Social Web for Situational Application Development and Business Mashups

Stefan Tai
stefan.tai@kit.edu
About the Speaker: Stefan Tai

- Professor, KIT (Karlsruhe Institute of Technology)
- Karlsruhe Service Research Institute (KSRI)
- Institute for Applied Informatics (AIFB)
- Director, FZI Research Center for Information Technology (FZI)
- Background: IBM T.J. Watson Research Center, New York, USA (1999-2007)
The eOrganization Research Group

www.fzi.de

www.eOrganization.de

www.aifb.uni-karlsruhe.de

www.ksri.uni-karlsruhe.de

www.kit.edu
Agenda

- Introduction:
  The Long Tail and Situational Applications (Mashups)

- The Swashup Platform and DSL
Gartner’s Hype Cycle for Emerging Technologies, July 2008

Figure 1. Hype Cycle for Emerging Technologies, 2006

- Technology Trigger
- Peak of Inflated Expectations
- Trough of Disillusionment
- Slope of Enlightenment
- Plateau of Productivity

Years to mainstream adoption:
- less than 2 years
- 2 to 5 years
- 5 to 10 years
- more than 10 years
- obsolete
- before plateau

Source: Gartner (July 2008)
Why Social Computing?

- Explosion in Web usage and content, fueled by global use, mobile device access, and the participation Web: user-generated content, cooperation and innovation
- The Web as the desktop (your desktop!)

- New concepts, methods, and technology needed to leverage the Web as a combined compute platform, business platform, and collaboration platform
The Long Tail of Business: Mass Servicing of Micro-Markets

Commercial Off The Shelf Software
- Millions of Generic Users
- Long-term, Expensive Development
- Good Quality (Reliability, Scalability)
- Professional Software Engineering

Mashups
- Few Specialized Users
- Short Lifespan, Cheap Development
- Good Enough Quality
- End-User Development (à la Excel)
Mashup Example: housingmaps.com
flickrvision.com
Mashups

- are Web applications composing a variety of Web sources via Web APIs, services and feeds (RSS, Atom)
  - Content provider, mashup site, client Web browser
- For situational purpose, fast and good enough
  - Consumer mashups vs. enterprise mashups

- Creation is mostly end-user oriented (often graphical)
- Typically data-driven, aggregating data streams
  - Yahoo Pipes, Microsoft Popfly, IBM Mashup Center, SAP Research Rooftop
- Sometimes process-driven using lightweight workflows
  - IBM Research Bite, Serena Mashup Suite
Yahoo! Pipes

- ... provides a graphical user interface for building applications that aggregate web feeds, web pages, and other services, creating *data mashups* from various sources, and publishing them.

- The site works by letting users "pipe" information from different sources and then set up rules for how that content should be modified (e.g. filtering, cf. the New York Times through Flickr example).
NY Times Through Flickr! (Yahoo! Pipes)
The Importance of Communities: Yahoo! Pipes Developer Community

- Pipe repository with some 10,000 mashups
  - Advanced searching, querying, and ranking capabilities
  - Reuse: Pipe Clones or references
- Web 2.0 support for pipes developers: tagging, feeds, blogs, forums
Agenda

- Introduction: The Long Tail and Situational Applications (Mashups)
- The Swashup Platform and DSL
IBM Sharable Code

An Online Platform for Situational Web Applications and Mashups Development and Management

- Started out as a research project at IBM Research
- Now an IBM alphaworks services offering: http://services.alphaworks.ibm.com/isc/

- Joint work with E. Michael “max” Maximilien, IBM ARC
  - http://www.maximilien.com
  - http://blog.maximilien.com
Motivation

- Mashups are **ad hoc**
  - Difficult to reuse and to share
  - Difficult to discover, to find, and to manage
- (Too) Many service models and types, e.g., SOAP/WSDL, REST, APP, Atom, RSS, ...
- JavaScript UI-based mashups lead to nightmarish code maintenance and security issues
- Lack of unified models and design for mashups
A uniform platform for developing, sharing, and managing Web 2.0 applications and mashups

- Common service models and mashup design
- Domain-specific languages (DSL) – abstraction
  - data, APIs, services, and mashup
  - tagging, commenting, digging, searching
- Web-based tooling
Philosophy

- Platform as a set of smaller parts
  - Web applications (tools)
  - Web APIs (services)
  - Reusable components (plugins and gems)
- Interactions between components via APIs
- The platform is itself mashable

- Web Computing is social, ergo sharable models, sharable components, and sharable designs are key to easier and more productive Web tools
The platform itself is mashable and designed as a modern RESTful SOA

- **Builder** application (mashup creator)
  - Enables “easy” design and creation of mashups
  - Agile and social development of mashups parts

- **Community** application
  - Advertisement and community of deployed mashups
  - Discussions, usages, tagging, searching, and ratings

- **Manager** application
  - One-click deployment of mashups to different servers or clouds
  - Management of deployments, e.g., clone, delete, version, ...
Big Picture

Amazon's compute cloud (i.e., EC2, S3, and simpleDB)

User PC or server

Server farms

MySQL or DB2
A Domain-specific language (DSL) for mashups

- “mini” language built on top of a hosting language that provides a common syntax and semantics
- High-level, mostly declarative, design and programming particular to mashup concepts and behaviors
- Enables easier sharing of parts
- Facilitates the creation of code and tools
Mashups’ designs as recipes (like cooking recipes)
- Web APIs, services, data, views, ... are ingredients
- Mashups are unique combinations of Web “ingredients”

Constructs
- Represent all basic concepts for creating mashups
- data, api, service, mashup, connection to user (wiring), connection between services (step), data transformations (mediator), and user interfaces (views)
DSL overview
DSL details

- **recipe** – Top level construct; aggregates all others
- **data** – Describes the schema (data elements) of service’s data
- **api** – Describes the service’s API interface
- **service** – Binds to the actual endpoint, service type, and any special considerations (e.g., caching, headers, ...)
- **mashup** – One or more services that comprise a collection of wiring and mediation declarations
- **mediation** – Facilitates data transformation
- **wiring** – Captures one point of interaction with user
- **step** – One atomic step in wiring; interaction between services
- **view** – Captures the view template code for a wiring
- **css** – Captures view styling via cascading stylesheets
- **tag, tags, desc** – Captures metadata and English text
Sample Recipe

```ruby
desc "Recipe for flickr.com REST and Google News Atom mashups"
tags ['mashup', 'rest', 'atom']
recipe :flickr_google_news_recipe do
  desc "Flickr's error response part"
data :FlickrErr do
    desc "Flickr's error <rsp> part"
data :FlickrErrRsp do
      desc "Flickr's <username>Some name</username> response part"
data :FlickrUsername do
        desc "Flickr's <photo>... </photo> response part"
data :FlickrPhoto do
          xml_name :photo
          member :id, :integer, :xml_attribute
          member :owner, :string, :xml_attribute
          member :secret, :string, :xml_attribute
          member :server, :integer, :xml_attribute
          member :form, :integer, :xml_attribute
          member :title, :string, :xml_attribute
          member :ispublic, :integer, :xml_attribute,
              :desc => 'integer encoded boolean'
          member :isfriend, :integer, :xml_attribute,
              :desc => 'integer encoded boolean'
          member :isfamily, :integer, :xml_attribute,
              :desc => 'integer encoded boolean'
        end
      end
    end
  end
end
desc "Flickr's <photos>... </photos> response part"
data :FlickrPhotos do
  xml_name :photos
  member :page, :integer, :xml_attribute
  member :pages, :integer, :xml_attribute
  member :perpage, :integer, :xml_attribute
  member :total, :integer, :xml_attribute
  member :photos, :FlickrPhoto, :xml_object_array,
      :desc => 'an array of FlickrPhoto'
end
```
Sample Recipe
Screen shots: Builder

Welcome to the Mashup creator
This mashup creator enables you to create interesting mashups easily using the swashup platform.
Please log in to use this mashup builder. If you don't have a password please register. Nothing to worry it is completely free :)

What is Swashup?
Swashup is a framework for making mashups. The Swashup (Scripting in Specific Language) makes it possible to create a single description of the mashup.
Screen shots: Builder

Create your Mashups - Easily and Effectively - Mozilla Firefox

project: new_project

Your Project Details

Create new Project

Data Schema

Create

Custom From Template From Example

Give a name for the data source

Name:
Description:
Xml Name:

Create

Add members to this Data Schema

Name Description Type Representation

Add
Screen shots: Builder

Project Details

You currently have 1 active project: new_project

View: sample_view

Create Delete

```
1. View: sample_view,
2. mashups => [:mashup_1],
3. content => %{
4.     \html \body
5.     <!-- add the components according to wiring detail here -->
6.   \}</body>\html>
```

Steps

View: sample_view

CSS

Position: Ln 8, Ch 3    Total: Ln 8, Ch 332

Toggle editor

update
Screen shots: Builder
Big Picture


IBM Sharable Code repository: http://knoesis.wright.edu/isc
Some Open Research Questions

- **Mashups programming model**
  - End-user-oriented integration paradigms
    - Mashup exploration, organization, search and integration
    - Reusable and customizable patterns for mashup creation
    - Combined user interface and application integration
  - Semantic Interoperability

- **Mashup platform**
  - Lightweight support of enterprise features (e.g. security, transactions)
  - Deployment and management in (various) clouds

- **Community-driven & collaborative mashup development**
  - Challenges of social networking applications
  - Integration with advances in mobile networking
Thank You!