Service-Oriented Computing: Emerging Approaches for Web-Based Software Engineering

M. Brian Blake

Associate Professor and Chair
Department of Computer Science
blakeb@cs.georgetown.edu
http://www.cs.georgetown.edu/~blakeb

Full Professor and Associate Dean of Engineering
University of Notre Dame
mb7@cse.nd.edu

Copyright 2009  Dr. M. Brian Blake, Georgetown University
My Background…

- **Preparation**
  - Bachelor of Electrical Engineering ‘94, Master of Electrical Engineering ‘97 (Georgia Institute of Technology, Mercer University) PhD ‘00 Information and Software Engineering (George Mason University)
  - Prior to academic appointment, 7 years as a full-time software engineer with General Electric, Lockheed Martin, General Dynamics, and The MITRE Corporation

- **Professional Activities**
  - 9th Year at Georgetown University on the faculty of the Department of Computer Science
  - Currently, Associate Professor and Department Chair (2nd year of 3 year term)
  - Ongoing consulting for Department of Justice, Department of Defense (and other unmentionables), Federal Aviation Administration, and several law firms

- **My research projects are in the areas of:**
  - Service-oriented computing and Service-oriented architecture, Intelligent software agents, Agent-mediated workflow, Data integration and data management, software engineering education and training

  - *How can you automate the integration of IT systems across organizations that never intended to be integrated? Why is this important currently?*
- Modularity of Web-Based Software
- Introduction to Service-Oriented Computing
- Background: Web Services
- Research Studies
  - Data Engineering for Web Services
  - Service Mashup
- Recently Funded Projects
- Q/A
Web Service Composition - Example

Simple Travel Reservation Capability

Composition of 3 Web Services:
- ReserveFlight
- ReserveHotel
- ReserveCar
A Realistic Travel Scenario

Additional complexities

- **Budget constraints on any part of the trip**
- **Certain reservations can be unsuccessful**
- **Sometimes the user will designate a specific business to use and other times not**
- **Any service can be down or inoperative**
- **Wife wants to come but does not want to come to Utah**
- **Wife has an equally complicated schedule**
Although Still too Simple, This is More Realistic

User Inputs: Start and End Date

1. Restaurant and Date
2a. Confirmed or Denied Reservations
2b. Confirmed Flights with Each Leg
3. Unknown or Specific Businesses and Date
4. Confirmations

Palm Travel Service

Meal Reservation Service

Flight Reservation Service

Rental Reservation Service

Hotel Reservation Service

Taxi Reservation Service
Modularity of Web-Based Software

**Introduction to Service-Oriented Computing**

Background: Web Services

Research Studies
- Data Engineering for Web Services
- Service Mashup

Recently Funded Projects

Q/A
A Service-Oriented Computing

Nirvana

Designing modular services/components for later integration

Discovery technologies that allow you to quickly find what you need.

Database of Service Specifications (UDDI)

WSDL

WSDL

WSDL

Local Business\textsubscript{n}

Engineering inter-organizational systems

Identifying what you have and what you need to procure.

Incorporating services built by others (i.e. the Amazon success)

Mediating data across services

INTERNET

Service Provider or Business\textsubscript{1}

Distributed Business\textsubscript{2}

Distributed Business\textsubscript{3}

Local Business\textsubscript{n}
Web Services are the core of it all…

- Web services are at the core of the service-oriented paradigms
  - Universal messaging format for data exchange (XML)
  - Distributed network-based access (SOAP)
  - Web services execute/evolve on the provider’s server
- A better definition later…

OK … not the panacea, but many new opportunities!
Several *Web Services* available on the NET

- Get historical end of day data for U.S. stock options
- Calls any phone number and speaks text or sound file to the person.
- Get FedEx shipping rate
- Current and historical foreign exchange rates
- Get five days weather report for a given zipcode (USA)
- Get name and address data associated to any telephone number
- Instantly determines the distance between two U.S. ZIP codes.
- Get the Barnes & Noble price by ISBN
The Typical Web Service…*It’s all about managing information.*

Input Message Pointer → Web Service Description (name, location, access) → Output Message Pointer

```
<portType name = "SampleCarRentalServicePortType">
  <operation name = "makeReservation">
    <input message ="tns:customerInformation"/>
    <output message="tns:reservationInformation"/>
    <fault message ="tns:reservationError"/>
  </operation>
</portType>
```

**SOAP**

```
<SOAP: Body xmlns:m ="http://www.example.org/CarRentalService">
  <m:customerInformation>
    <m:name> M. Brian Blake </m:name>
    <m:destination> DullesAirport </m:destination>
    <m:pickupDate> 2/30/2003 </m:pickupDate>
    ............
  </m:customerInformation>
</SOAP: Body>
```

customerInformation.xml (Input Message)
Not so easy in real life though…

WSDL Specification Metamodel

Step 1:
Get Operation Name and
It's Operation/Port Relationship

Step 2:
Get Detailed Message Information by Part Names.
(Sometimes these are inline, other times data must be extracted from types.)

Step 3:
If necessary, traverse through connected message information from WSDL types (as
times, types have a nested hierarchy)
Amazon Web Services...Good but Ugly??

```xml
<operation name="ItemLookup">  
  
<input>  
<soap:body use="literal"/>
</input>

<output>  
<soap:body use="literal"/>
</output>

<operation name="BrowseNodeLookup">  
  
<input>  
<soap:body use="literal"/>
</input>

<output>  
<soap:body use="literal"/>
</output>

<operation name="ListSearch">  
  
<input>  
<soap:body use="literal"/>
</input>

<output>  
<soap:body use="literal"/>
</output>
```

Amazon AWSECommerceService
Integrating Software Systems

Introduction to Service-Oriented Computing

Background: **Web Services**

Research Studies
- Data Engineering for Web Services
- Service Mashup

Recently Funded Projects

Q/A

---

**Why Web Services?**

**Using a DOD Scenario for Motivation**
Automated Discovery/Composition: An Army Scenario


4. Service Delivery: Return composite service

1. Mission Objective: Need to exploit situational awareness in order to determine if arms can be delivered from A to B.

User/Mission Context: Consumer Role, Organization Location, Access Level, Priority, Criticality

Service/Operational Context: Current/Anticipated Bandwidth, CPU Utilization, Workload, and Priority

Selection Criteria A Functional: Which types of services matter considering the context?

Selection Criteria B: Nonfunctional: Considering the context, which service instances, what process sequence, and what overall solution?

Virtual Service Repository (i.e. Federation of Web Service Databases (UDDI))

Situational awareness limited to last brief and current contact
Research Studies….

1. Using Service Inputs and Outputs to discover pertinent services and Service Mashup

2. Discovery:
   Identify candidate services

3. Multidimensional Tradeoff Analysis:
   Use context for both functional and nonfunctional selection.

   Selection Criteria A
   Functional:
   Which types of services matter considering the context?

   Selection Criteria B:
   Nonfunctional:
   Considering the context, which service instances, what process sequence, and what overall solution?

4. Service Delivery:
   Return composite service

4. Using State-of-the-Practice Software Engineering to Deliver Composite Capabilities
Research Studies:
Data Engineering for Web Services

How to identify candidate services?
Data Engineering for SOC

1. How do you know if a user’s initially-supplied information is the same as the information required by the service?

Initial Inputs
- firstName
- lastName
- departureDate
- destinationCode
- creditCardNum
- creditCardSecNum

Resultant
- rentalConfirmation

Output-to-Input Mapping
- flightID
- airlineID

Output-to-Input Mapping
- flightConfirmation
- arrivalTime

Output-to-Input Mapping
- carTypeID
- rentalCenterID

2. How do you know if one service’s output is compatible to another service’s input?

findFlight.wsdl
- bookFlight.wsdl
- findRentalCar.wsdl
- bookRentalCar.wsdl

3. How do you know if the resulting workflow is ultimately the correct context of the overall user’s request?
Do you mean what I mean?
(Using ontological approaches)

Developing a local consensus ontology…
Blake et. al. AAMAS 2003, IEEE TKDE 2005
Ontology not widely used in practice

- Web services can embed semantic (ontology-based) notations using several techniques
  - (e.g. RDF, OWL-S, WSDL-S, etc.)
- Industry has not embraced these approaches, to date.

We took a sabbatical on semantic solutions and revisited syntactical approaches using natural language processing techniques
Tendency-Based Syntactical Matching (TSM)

- We introduce a syntactical approach to service discover/composition that uses tendencies of developers to name service inputs/outputs in a characteristic manner.
- Obviously this approach *does not* replace semantic approaches.
- However, this approach can:
  1. Help to understand detrimental software engineering practices currently seen in real services.
  2. Suggest an initial subset of potentially-relevant syntactical techniques that may improve the performance of semantic approaches on open repositories in the future.
Gathering Tendencies

- To derive tendencies, we downloaded real, working services from over 5 internet repositories, as well as exhaustive online searches. We built a repository of ~600 WSDL files, over ~7000 operations, over ~30,000 message names.

- We developed a matching approach (TSM-LP) based on the tendencies

Our group has perhaps the most complete repository of *real* Web services for experimentation.
Most Common Service Input/Output Naming

Tendencies

- **Tendency 1:**
  - Similar Input/Output names tend to have subsumption relationships
    - (i.e. name = lname, name = firstname, and name = user_name)

- **Tendency 2:**
  - Similar input/output names tend to have equivalent subsets
    - (i.e. first_name and user_name)

- **Tendency 3:**
  - Developers tend to use abbreviations
    - (i.e. building = bldg)

- **Tendency 4:**
  - Words less than 3 characters or greater than 15 are impractical for matching in this context.
Our Approach: TSM-LP

- We call this similarity approach Tendency-based Syntactic Matching – (Levenhstein Distance) (Letter Pairings).
- TSM-LP combines four different matching methods:
  1. Exact string equivalency
  2. Subsumption of Str1 in Str2 or Str2 in Str1
  3. Levenhstein Distance: Number of Transformations.
  4. Percentage of Letter Pairings present in both words. Str1 and Str2 have two equivalent pairings

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSM-LP(Si, Sj)</td>
<td>TSM-L Function</td>
</tr>
<tr>
<td>LD(Si, Sj)</td>
<td>Levenshtein Distance function</td>
</tr>
<tr>
<td>FT1(Si)</td>
<td>Tendency-Based Threshold</td>
</tr>
<tr>
<td>FT2(Si)</td>
<td>Tendency-Based Threshold for Letter Pairing</td>
</tr>
<tr>
<td>Si, Sj</td>
<td>Two strings for comparison</td>
</tr>
<tr>
<td>Length()</td>
<td>String length functions</td>
</tr>
<tr>
<td>CS</td>
<td>Web Service Category (e.g. Business)</td>
</tr>
</tbody>
</table>

\[
FT1(S_i) = \left\lfloor \frac{(\text{Length}(S_i) \times 2)}{3} \right\rfloor - 2
\]
return temp

\[
FT2(S_i) = \text{Sensitivity}(CS)
\]
return temp

\[
\text{TSM-LP}(S_i, S_j) = \begin{cases} 
\text{true} & \text{if } (LD(S_i, S_j) \leq FT1(S_i)) \text{ or } \text{LP}(S_i, S_j) \geq FT2(S_i) \text{ or } (S_i \subseteq S_j \text{ or } S_j \subseteq S_i) \text{ and } (S_i > 3 \text{ and } S_j > 3) \text{ and } (S_i < 3 \text{ and } S_j < 15) \\
\text{false} & \text{else}
\end{cases}
\]
return TRUE

return FALSE
TSM-LP Application

- Reasonable approach for service recommendation
  - But not, real-time service integration
Matching Results

Contribution of tendencies in matching
(1,054,137 matches of 1,322,988)
Relatively small overlap.

Accuracy of Top 50 most common message names for matching
Sample Recommendations

Use uniqueness of message names by category to set recommendation thresholds

<table>
<thead>
<tr>
<th>Category</th>
<th>Self-Similarity Percentage</th>
<th>TSM-LP Sensitivity</th>
<th>LD Threshold</th>
<th>LP Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>12.5 - 25%</td>
<td>High</td>
<td>[\frac{(\text{Length}(S_i) \times 2)}{3}] - 3</td>
<td>55.0%</td>
</tr>
<tr>
<td>Graphics</td>
<td>25 - 50%</td>
<td>Medium</td>
<td>[\frac{(\text{Length}(S_i) \times 2)}{3}] - 2</td>
<td>47.5%</td>
</tr>
<tr>
<td>Communication</td>
<td>50 - 75%</td>
<td>Low</td>
<td>[\frac{(\text{Length}(S_i) \times 2)}{3}] - 1</td>
<td>40.0%</td>
</tr>
</tbody>
</table>

Services recommended after using random files

<table>
<thead>
<tr>
<th>Type of File</th>
<th>Operation Name</th>
<th>Relevancy Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itinerary generated from Travel website</td>
<td>GetStations</td>
<td>2350</td>
</tr>
<tr>
<td></td>
<td>IsValidExchange</td>
<td>2350</td>
</tr>
<tr>
<td></td>
<td>IsExchangeOpen</td>
<td>2200</td>
</tr>
<tr>
<td>Currency conversions webpage</td>
<td>GetSearchTerms</td>
<td>1050</td>
</tr>
<tr>
<td></td>
<td>NumberToDollars</td>
<td>1050</td>
</tr>
<tr>
<td></td>
<td>Search</td>
<td>1000</td>
</tr>
<tr>
<td>Random book search from online bookseller</td>
<td>ListBooks</td>
<td>1600</td>
</tr>
<tr>
<td></td>
<td>BooksInfo</td>
<td>1400</td>
</tr>
<tr>
<td></td>
<td>WishlistSearchRequest</td>
<td>1250</td>
</tr>
<tr>
<td>Finance homepage on Yahoo.com</td>
<td>IsValidExchange</td>
<td>1200</td>
</tr>
<tr>
<td></td>
<td>GetCurrentMortgageIndex</td>
<td>1150</td>
</tr>
<tr>
<td></td>
<td>IsExchangeOpen</td>
<td>1100</td>
</tr>
<tr>
<td>Sports homepage on msn.com</td>
<td>GetSportNews</td>
<td>1850</td>
</tr>
<tr>
<td></td>
<td>WorldCupFootball</td>
<td>1650</td>
</tr>
<tr>
<td></td>
<td>GetBriefings</td>
<td>1200</td>
</tr>
</tbody>
</table>
Research Studies: Service Mashups
What is a Service Mashup?

- Taking the outputs from, potentially unrelated, web services to create new capabilities or information

  - In Practice: ProgrammableWeb.com & Yahoo Pipes
  - Example: Overlaying a map with shipment routing information
Other Interesting Mashups…

WiiFinder: Find the nearest Wii for sale.

- Combining Amazon eCommerce, eBay, and Google Maps
Other Interesting Mashups…

Cell Phone Reception: Cell towers by location

- Combining
  - Various telecom sites
  - GoogleMaps
Other Interesting Mashups…

Visual Traceroute: Show the tracert command

- **Combining:**
  - TraceRt
  - DNS
  - GoogleMaps
Research Questions..

- Considering open web services over the Internet, services in a federated registry, or even services in a intranet-based repository….

  - What are the common characteristics of two services that make them qualified for mashup?
  - What are the relations between the messages of such services?
  - What techniques can be exploited to evaluate service messages in order to predict viable service mashups?
Related Work

- Service mashup is an emerging approach to software and data integration

  - Traditional software engineering approaches attempted to match software interfaces in standard programming environments (Zaremski and Wing, 1997)
  - Most recent projects for service mashup concentrate on toolkits that enable the data integration (Liu et. al, 2007; Sabbouh et. al., 2007)
  - Other approaches attempt to protect mashup data (Zou et al., 2007)

- Our work attempts to derive a mining approach for service mashup by evaluating real services
Considering an open repository of “real” web services, we performed experimentation to determine:

- The likelihood that similar message part names can predict candidate services for mashup
- Whether input or output part names are more meaningful for predicting candidates
- What thresholds dictate when message names or syntactically similar enough for candidate prediction

- Of course, in the absence of semantic metadata (i.e. OWL, WSDL-S, etc.)
Leveraging Similarity Studies for Mashup

Evaluate multiple web services for similar message parts

- Disregard services that have too many parts in common

New Work:
- Gather insight from Web2.0 sites
- Use congenial services more frequently in prediction
Experimentation

- From our repository of 6,000 services, we experimented with 100 services randomly selected for experimentation.

- Assessments:
  - Total number of Predicted Mashups considering variable similarity strictness and 1 similar output messages
  - Total number of Predicted Mashups considering variable similarity strictness and variable similar output messages
  - Precision of Predicted Mashups

- Visual inspections were used to determine precision and recall which required smaller experimental sets.
Predicted Mashups considering Variable Strictness

- Levenstein Distance and subsumption were most effective
- In earlier service discovery work (i.e. discovering 1 service), TSM-LP was most effective
- As would be expected, more stringent requirements for similar messages reduces the total number of predicted mashups
Precision of Predicted Mashups

Although there is a gain in precision with more stringent requirements, that gain only varies 5-15% which is not proportional to reduction in total predictions.
Other Results: Sample Mashup and Most Commonly Correlated Messages

<table>
<thead>
<tr>
<th>Top 6 Most Common Message Parts for Predicting Mashups</th>
<th>Percentage of Top 6 Used for Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>State, City, Name, Date, Time, Zip</td>
<td>29%</td>
</tr>
</tbody>
</table>

### Diagram

- **GetAddress**
- **CountryFIPS**
- **Zip**
- **CityType**
- **FirstName**
- **LastName**
- **Username**
- **PhoneIntel**
- **ValidateEmail**
- **FraudLabs** (credit card check)
- **NameSuffix**
- **Address**
- **AgeIndex**
- **CellPhone**
- **Name Prefix**
- **Postal**

**Shared Outputs:**
- **City**
Summary and Future Work

- Syntactic matching applied to similar outputs can be an effective/efficient approach to process large repositories for service mashups
  - ~80% precision, 100 service comparisons in 900 ms

Future Work.....

- Perform assessments that combine input messages and output messages
- Using positive service mashups to derive semantic meaning from existing services
- Clustering approaches for chaining groups of mashups
- Modularity of Web-Based Software
- Introduction to Service-Oriented Computing
- Background: Web Services

- Research Studies
  - Data Engineering for Web Services
  - Service Mashup

- Recently Funded Projects & Conclusions

- Q/A
SOA at Georgetown University

- Focuses on service-oriented computing incorporating intelligent agents and workflow management techniques

- SOC Projects (Over $5.5 Million from 2003-present)
  - Current (~$5 Million)
    - Service Composition Techniques and Evaluation – NSF
      (http://www.ws-challenge.org)
    - Service-Oriented Training Modules for Human Learning – NSF, BMW
    - Integrating SOC with the High Performance Computing – DARPA, US Council of Competitiveness
    - Service Level Agreements – The MITRE Corp, DOD, other agencies
    - Service-Oriented Architecture Curriculum – IBM, Allstate, US Mint, DOD
  - Pending, Past, or Awaiting Phase II
    - Integrating SOC with HPC – AFOSR (pending)
    - Sharing Services and Intelligence Information – AFRL, SAIC (past)
    - Context-Based Service-Oriented Computing – The MITRE Corp (past)
    - Integrating Components for Surgical Interventions – Georgetown University Medical Center, NIH (on-going)
# Meet the Team....

<table>
<thead>
<tr>
<th>PostDocs (Jan ‘08)</th>
<th>Graduate Students</th>
<th>Undergraduates</th>
<th>Undergraduates (non-CS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ajay Bansal, PhD, UT-Dallas</td>
<td>ImanMoustafa CS, PhD Student Virginia Tech</td>
<td>Michael Nowlan, Senior, CS</td>
<td>Erik Muller Senior, Business</td>
</tr>
<tr>
<td>Srividya Kona PhD, UT-Dallas</td>
<td>Ahmed Hamza CS, Master’s Student</td>
<td>Brian Miller, Sophomore, CS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mustafa Dustani CS, Master Student</td>
<td>Ryan Butler, Senior, CS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Michael Lefebvre CS, Master Student</td>
<td>Alex Yale-Loehr Freshman, CS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Khaled El-Goarany CS, MS Student Virginia Tech</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Graduates:**
  - Amy L. Sliva, PhD Candidate, University of Maryland-College Park
    - ACM CRA Research Award Runner-Up
    - ACM National Research Competition Finalist
  - Wendell Norman, Software Engineer, The MITRE Corporation
  - Georgina Saez, Software Engineering Consultant, Accenture
  - Todd Cornett, Master Student, Stanford University
  - Tepring Piquado, PhD Candidate, Brandeis University
**Contribution Summary**

**Component Workflows for Distributed Data Management**

**Agent-Based Workflow Management of Distributed Components**

**Software Engineering Training for Distributed Group Projects**

**Agent-Mediated Training**
ICALT 2006, AAMASWkshp 2007, IJAIED2009

**Service-Oriented Computing**

**Service Composition**

**Service-Based Discovery, Recommendation, & Management**
ACMTWEB (pending), IEEE TKDE (pending)

**Service-Oriented Software Engineering**
Integrating Software Systems

Introduction to Service-Oriented Computing

Background: Web Services

Research Studies
- Data Engineering for Web Services
- Service Mashup

Recently Funded Projects

Q/A

Thank you. Questions....

mb7@cse.nd.edu