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

(effective 7/2009)

Full Professor and Associate Dean of Engineering

University of Notre Dame

mb7@cse.nd.edu

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?***

Presentation Outline

- **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 Reservati… …bility

Over-used and Unrealistic

Comp… …Web Services:

- *…serveFlight*
- *ReserveHotel*
- *ReserveCar*

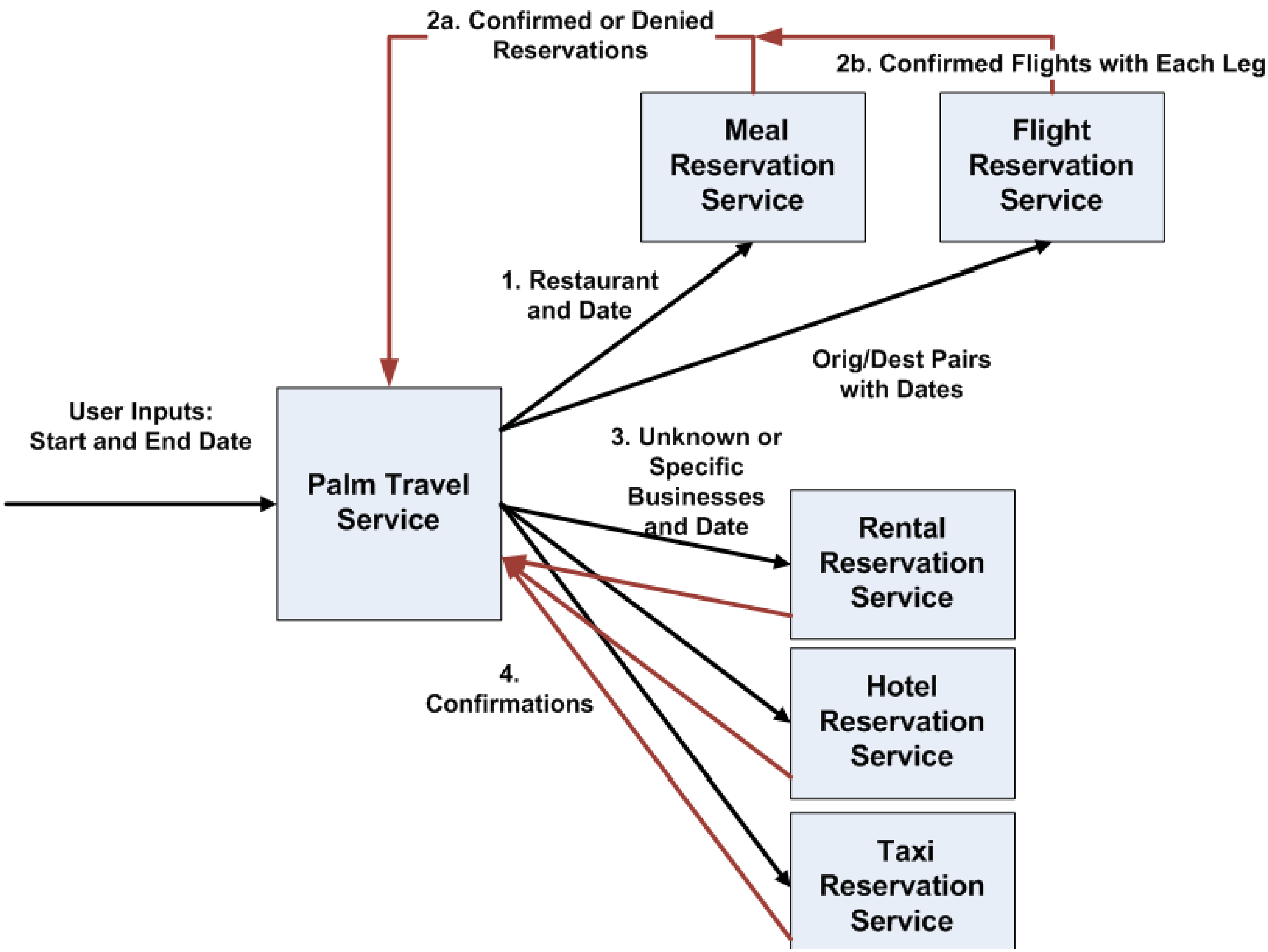
A Realistic Travel Scenario

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY	MONDAY	TUESDAY
		HILTON RESORT HONOLULU	PAPER PRESENTATION HONOLULU				SNOWBIRD SKI RESORT SNOWBIRD, UT		
LATE AFTERNOON TALK TO HIGH SCHOOL STUDENTS WASH, DC	HOUSE OF NANKING SAN FRAN	SESSION CHAIR HONOLULU	PAPER PRESENTATION HONOLULU				CRA SNOWBIRD DEPT CHAIR SESSION SNOWBIRD, UT		
JETBLUE FLIGHT or MORNING (Dep > 8pm & $)					HOTEL (??) HAWAII or IN SAN FRAN ($)	CRUSTACEANS SAN FRAN			HOTEL SNOWBIRD, UT or WASH,DC (Arr < 8pm & $)

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*

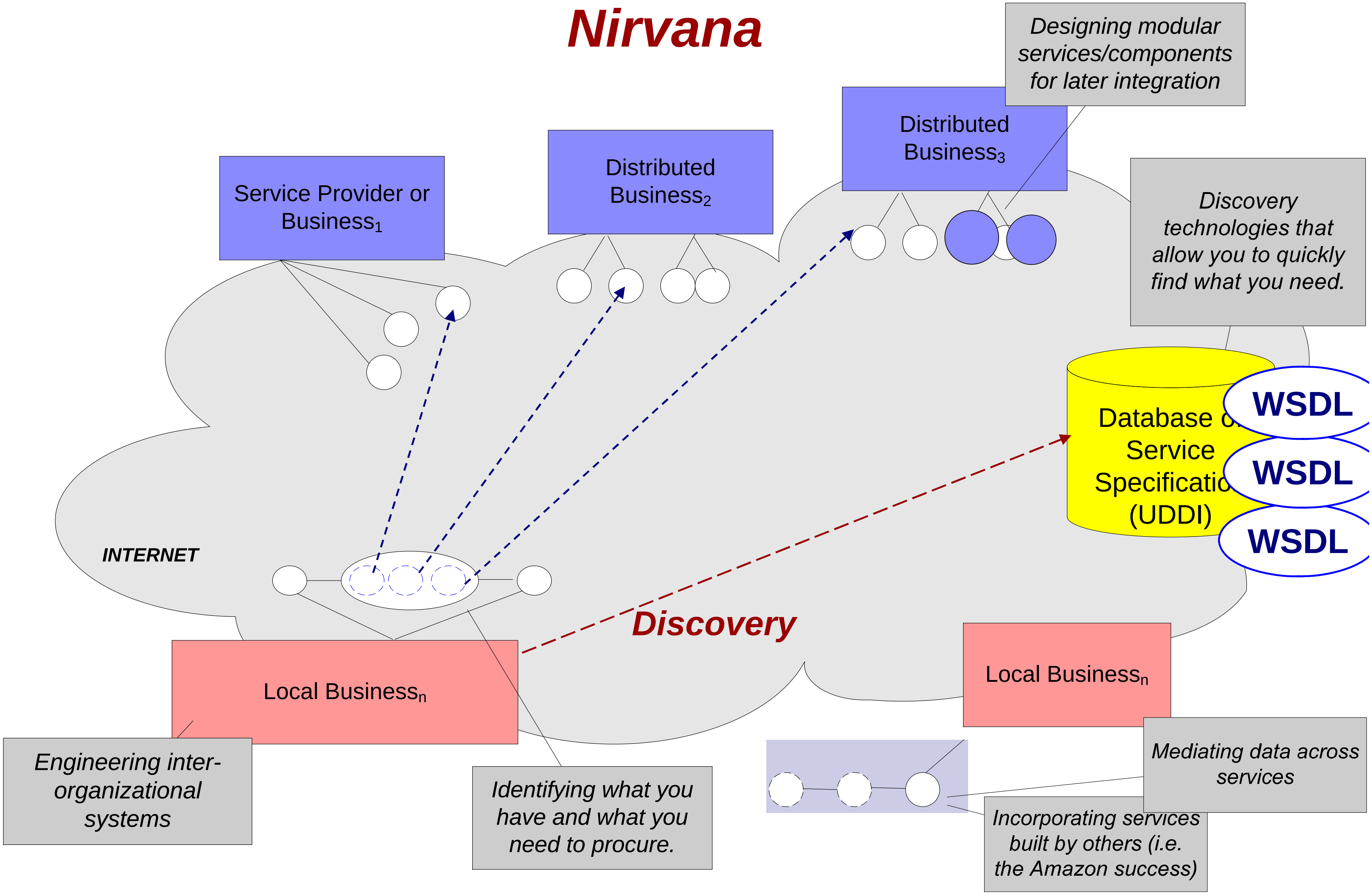
Although Still too Simple, This is More Realistic



- 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

Introduction to Service-Oriented Computing

A Service-Oriented Computing
Nirvana



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…

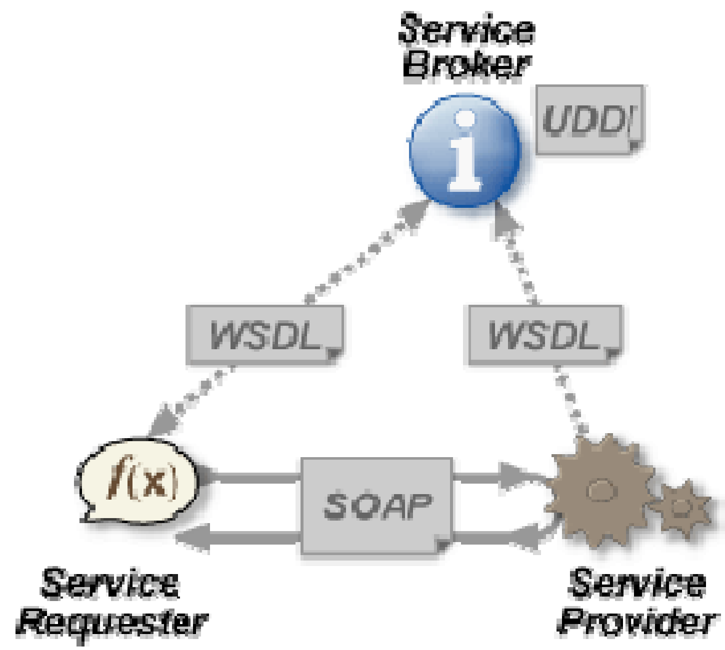


Image from Wikipedia 2007

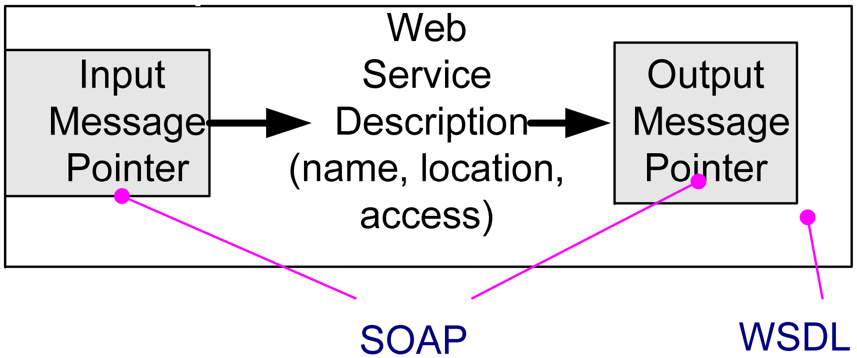
OK … not the panacea, but many new opportunities!

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.*



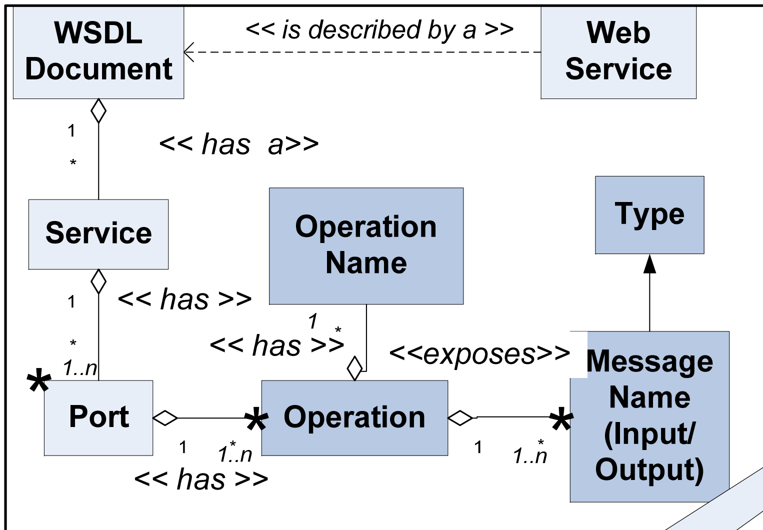

```
<portType name = "SampleCarRentalServicePortType">
          <operation name = "makeReservation">
                    <input message ="tns:customerInformation"/>
                    <output message="tns:reservationInformation"/>
                    <fault message ="tns:reservationError"/>
      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

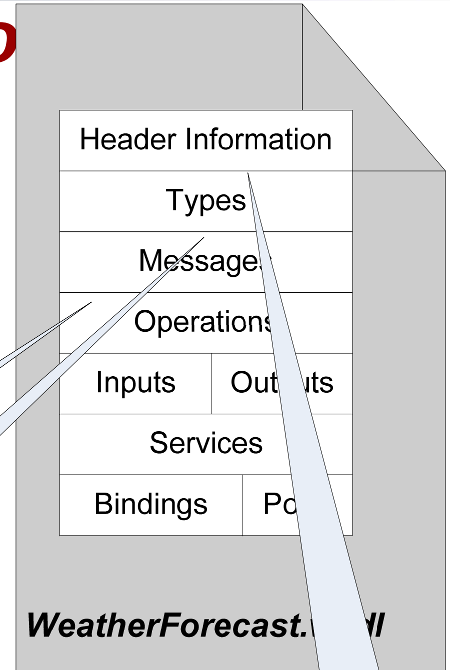
```
<SOAP: Body xmlns:m "http:/www.example.org/
                              CarRentalService">
<m:customerInformation>
          <m:name> M. Brian Blake </m:name>
          <m:destination> DullesAirport </m:destination>
          <m:pickupDate> 2/30/2003 </m:pickupDate>
          ..............
</m:customerInformation>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

customerInformation.xml (Input Message)

Not so easy in real life tho

WSDL Specification Metamodel






```
<wsdl:operation name = "GetWeatherByPlaceN                >
    <wsdl:i
    <wsdl:i
</wsdl:ope
```

Get
Inp

```
<wsdl:message name="getWeatherByPlaceNameHttpPostIn">
        <wsdl:part name="PlaceName" type="s:string" />
</wsdl:message>

<wsdl:message name="GetWeatherByPlaceNameHttpPostOut">
      <wsdl:part name="parameters" element="tns:GetWeatherByPlaceNameRespon
</wsdl:message>
```

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

```
<wsdl:types>
<s:element name="GetWeatherByPlaceNameResponse">
   <s:complexType>
     <s:sequence>
         <s:element minOccurs="0" maxOccurs="1"
name="GetWeatherByPlaceNameResponse"  type="tns:WeatherForecast
     </s:sequence>
   </s:complexType>
</s:element>

<s:complextype name = "WeatherForecasts">
     <s:sequence>
         <s:element minOccurs="0" maxOccurs="1" name="Latititude"  type
         <s:element minOccurs="0" maxOccurs="1" name="Longitude"  type
         <s:element minOccurs="0" maxOccurs="1" name="Temperature"  t
         <s:element minOccurs="0" maxOccurs="1" name="Details"  type="s
     </s:sequence>
   </s:complexType>
<wsdl:types>
```

Step 3:
If necessary, traverse through connecte
message information from WSDL types (
times, types have a nested hierarchy)

Amazon Web Services…*Good but Ugly??*

```
      <operation name="ItemLookup">
<soap:operation soapAction="http://soap.amazon.com"/>
      <input>
<soap:body use="literal"/>
</input>
      <output>
<soap:body use="literal"/>
</output>
</operation>
      <operation name="BrowseNodeLookup">
<soap:operation soapAction="http://soap.amazon.com"/>
      <input>
<soap:body use="literal"/>
</input>
      <output>
<soap:body use="literal"/>
</output>
</operation>
      <operation name="ListSearch">
<soap:operation soapAction="http://soap.amazon.com"/>
      <input>
<soap:body use="literal"/>
</input>
      <output>
<soap:body use="literal"/>
</output>
```

```
vices.amazon.com/AWSECommerceService/2
vices.amazon.com/AWSECommerceService/2
'>
Id" type="xs:string" minOccurs="0"/>
" type="xs:string" minOccurs="0"/>
pe="xs:string" minOccurs="0"/>
="tns:HelpRequest" minOccurs="0"/>
e="tns:HelpRequest" minOccurs="0"
lpRequest">
"xs:string" minOccurs="0"
pe" minOccurs="0">
:string">
on"/>
p" type="xs:string" minOccurs="0"
arch">
Id" type="xs:string" minOccurs="0"/>
" type="xs:string" minOccurs="0"/>
 type="xs:string" minOccurs="0"/>
pe="xs:string" minOccurs="0"/>
="tns:ItemSearchRequest" minOccurs="0"
e="tns:ItemSearchRequest" minOccurs="0
emSearchRequest">
"xs:string" minOccurs="0"/>
="xs:string" minOccurs="0"/>
ating" minOccurs="0" maxOccurs="unboun
="xs:string" minOccurs="0"/>
"xs:string" minOccurs="0"/>
 type="xs:string" minOccurs="0"/>
xs:string" minOccurs="0"/>
pe="xs:string" minOccurs="0"/>
" minOccurs="0"/>
```

N + 46

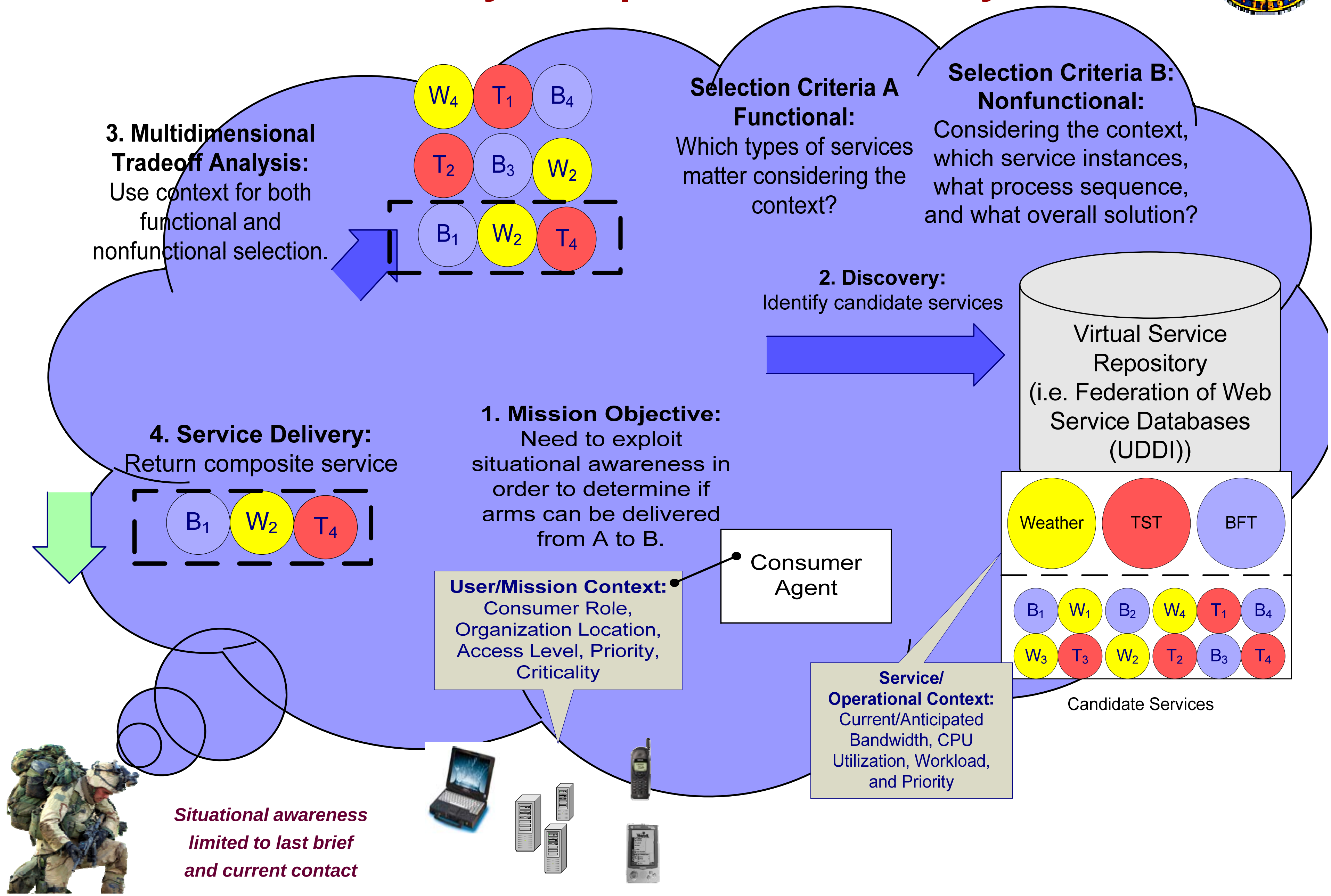
Amazon AWSECommerceService

- 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

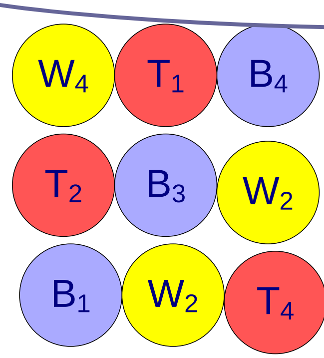
Automated Discovery/Composition: An Army Scenario



Research Studies….

2. Discovery:
Identify candidate services

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



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

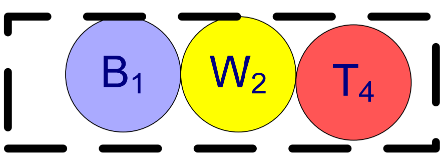
2. Using Context to Aid Discovery

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

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

3. Using Service Level Agreements to Aid Discovery

4. Service Delivery:
Return composite service



4. Using State-of-the-Practice Software Engineering to Deliver Composite Capabilities

- Modularity of Web-Based Software
- Introduction to Service-Oriented Computing
- Background: Web Services
- **Research Studies**
  - **Data Engineering for Web Services**
  - Service Mashups
- Recently Funded Projects
- Q/A

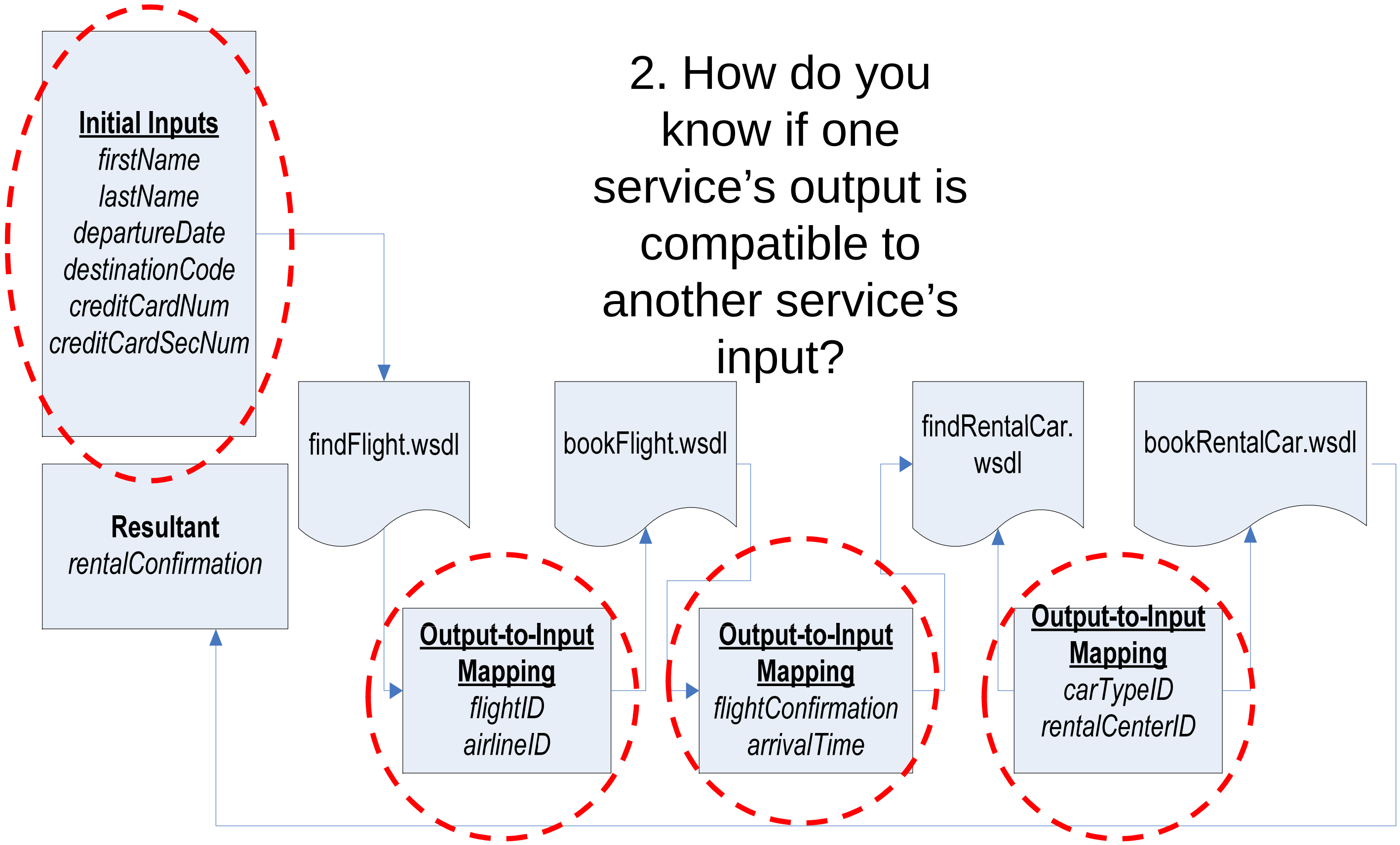
Research Studies: Data Engineering for Web Services

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?

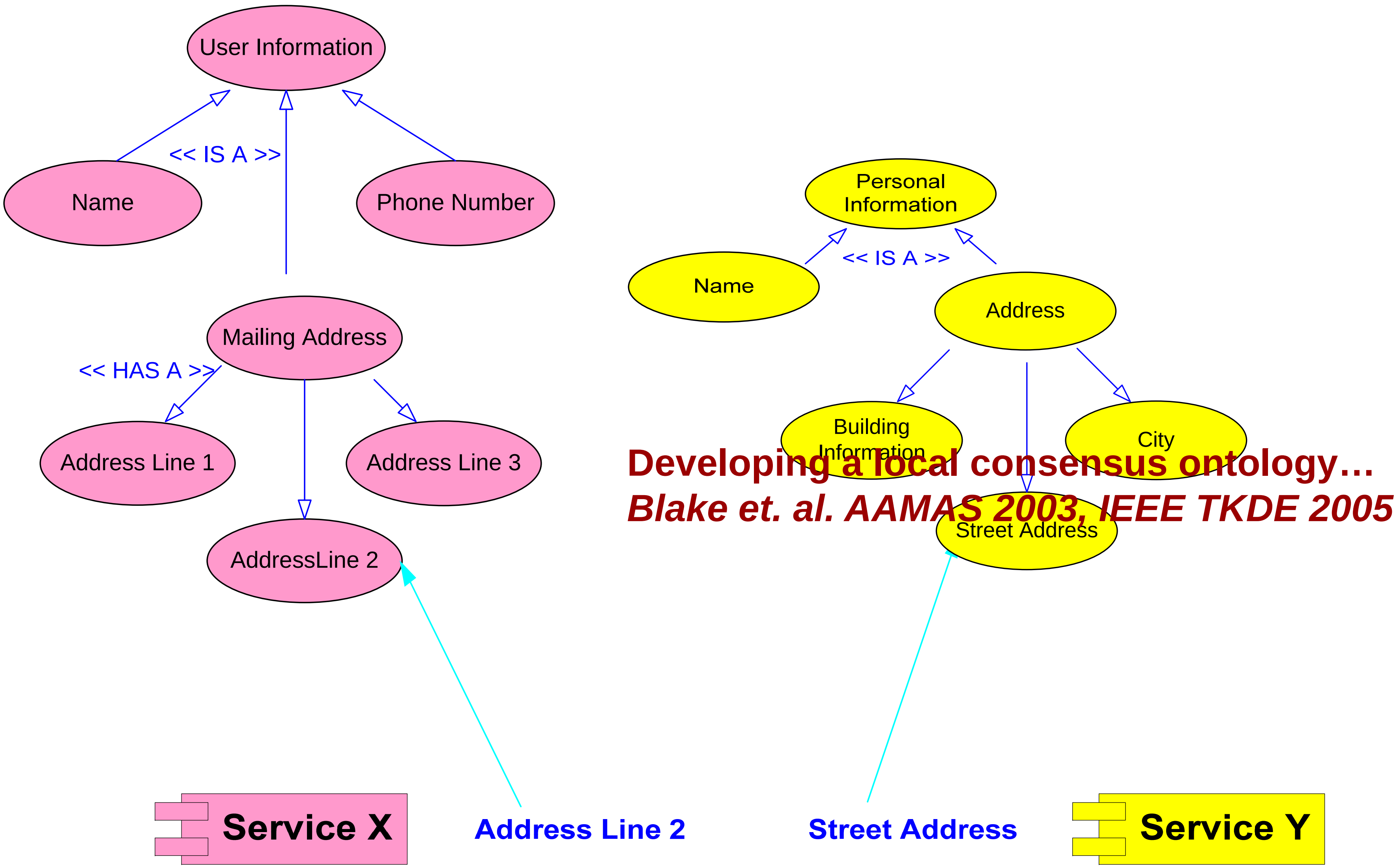
2. How do you know if one service's output is compatible to another service's input?



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

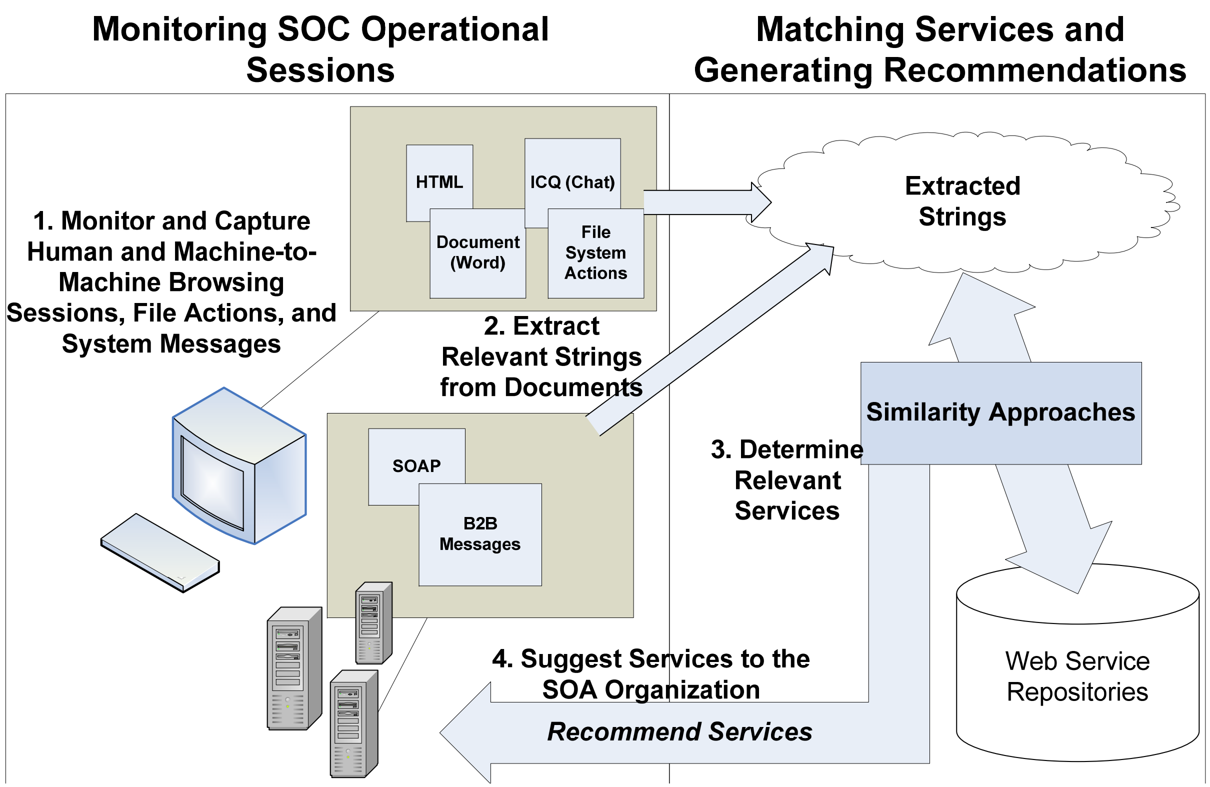
TSM-LP(S_i , S_j):	TSM-L Function
L_D(S_i , S_j):	Levenshtein Distance function
$F_{T1}(S_i)$:	Tendency-Based Threshold
$F_{T2}(S_i)$:	Tendency-Based Threshold for Letter Pairing
S_i , S_j :	Two strings for comparison
Length():	String length functions
C_S	Web Service Category (e.g. Business)

$F_{T1}(S_i)$
    temp = [(*Length*(S_i) * 2) / 3] – 2
    return *temp*
$F_{T2}(S_i)$
    temp = *Sensitivity* (C_S)
    return *temp*

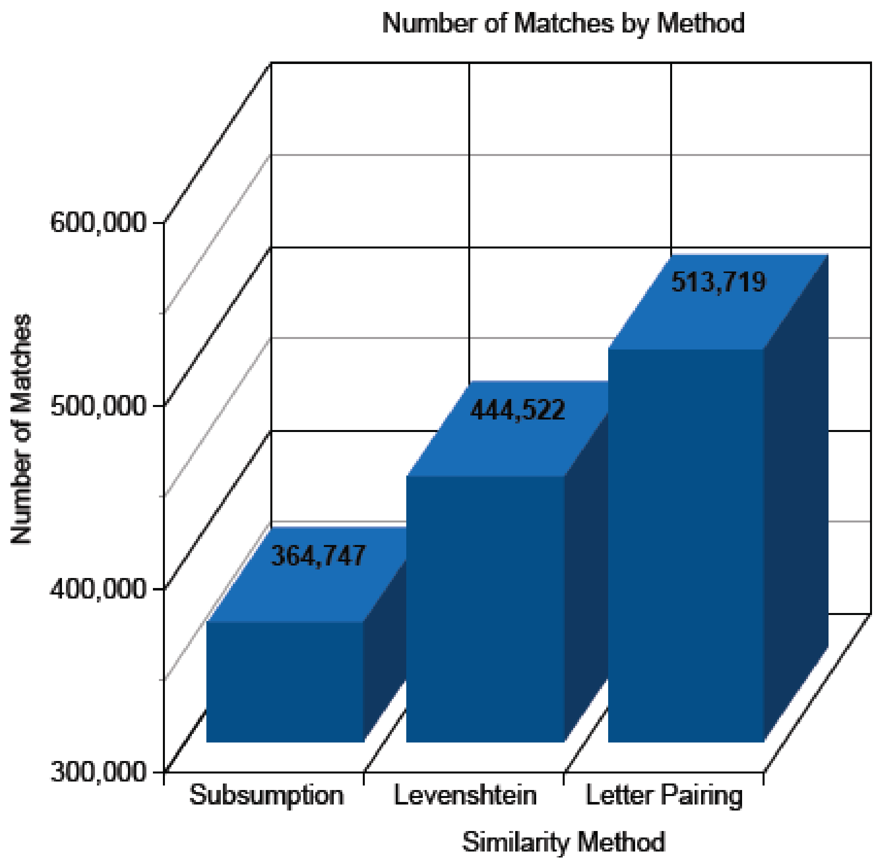
TSM-LP(S_i , S_j)
    if ($L_D(S_i , S_j) <= F_{T1}(S_i)$) or
        ($L_P(S_i , S_j) >= F_{T2}(S_i)$) or
        ($S_i \subseteq S_j$ or $S_j \subseteq S_i$) and
        ($S_i > 3$ and $S_j > 3$) *and*
        ($S_i < 3$ and $S_j < 15$)
        return *TRUE*
    else
        return *FALSE*

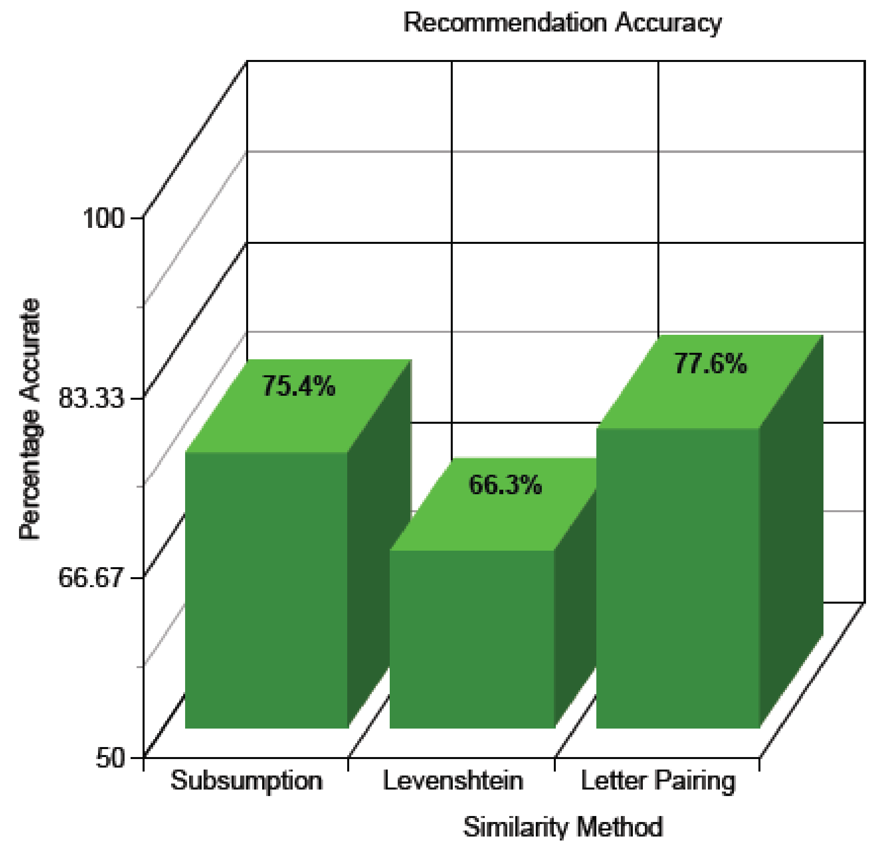
TSM-LP Application

- Reasonable approach for service recommendation
  - But not, real-time service integration
    - Blake & Nowlan (2007), "Recommending Web Services via an Agent Federation" *Multiagent and Grid Systems Journal*



Matching Results

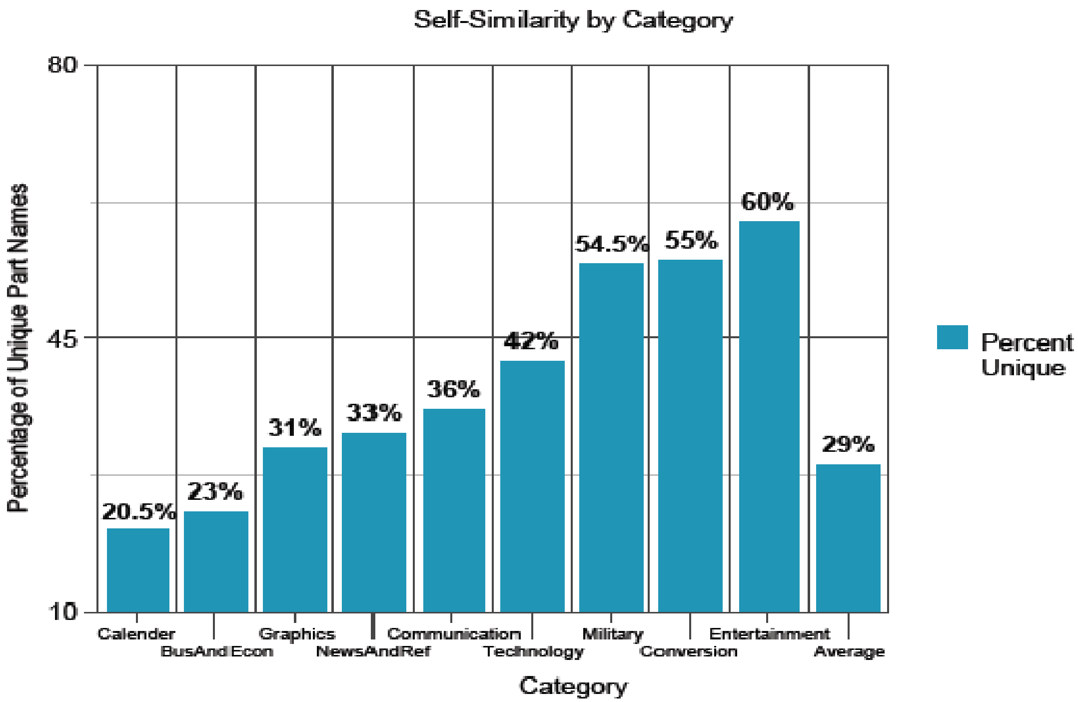




Contribution of tendencies in matching
(1,054,137 matches of 1,322,988)
***Relatively small overlap*.**

Accuracy of Top 50 most common message names for matching

Sample Recommendations



Use uniqueness of message names by category to set recommendation thresholds

Self-Similarity Percentage	**TSM-LP Sensitivity**	**LD Threshold**	**LP Threshold**
12.5 - 25%	High	[(*Length*(S_i) * 2) / 3] – 3	55.0%
25 - 50%	Medium	[(*Length*(S_i) * 2) / 3] – 2	47.5%
50 - 75%	Low	[(*Length*(S_i) * 2) / 3] – 1	40.0%

Type of File	***Operation Name***	***Relevancy Score***
Itinerary generated from Travel website	GetStations	2350
	IsValidExchange	2350
	IsExchangeOpen	2200
Currency conversions webpage	GetSearchTerms	1050
	NumberToDollars	1050
	Search	1000
Random book search from online bookseller	ListBooks	1600
	BooksInfo	1400
	WishlistSearchRequest	1250
Finance homepage on Yahoo.com	IsValidExchange	1200
	GetCurrentMortgageIndex	1150
	IsExchangeOpen	1100
Sports homepage on msn.com	GetSportNews	1850
	WorldCupFootball	1650
	GetBriefings	1200

Services recommended after using random files

- Integrating Software Systems
- Introduction to Service-Oriented Computing
- Background: Web Services
- **Research Studies**
  - Data Engineering for Web Services
  - ***Service Mashups***
- Recently Funded Projects
- Q/A

Research Studies: Service Mashups

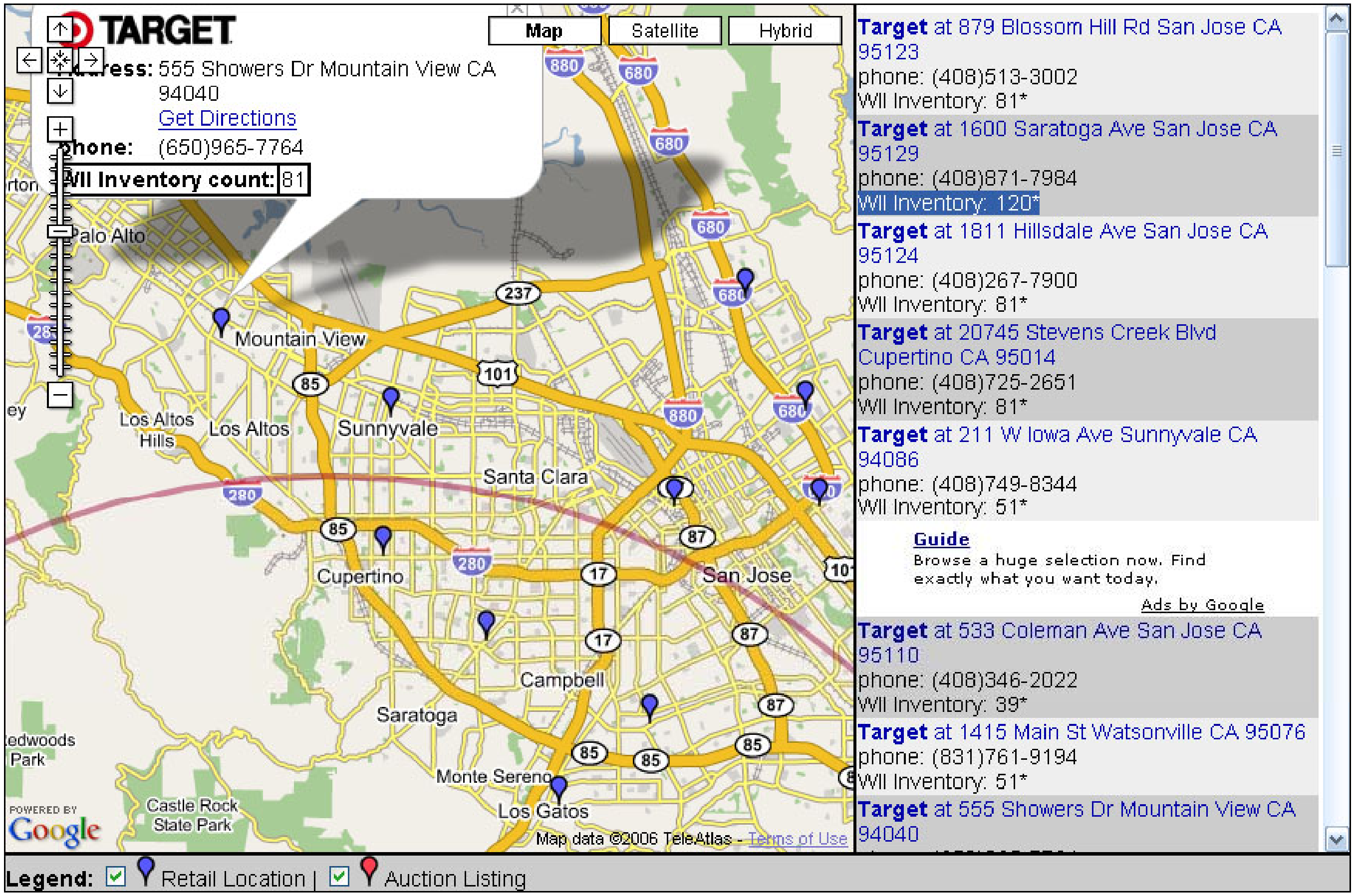
What is a Service Mashup?

- Taking the outputs from, potentially unrelated, web services to create new capabilities or information
  - *In Practice: ProgrammableWeb.com &YahooPipes*
  - *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…

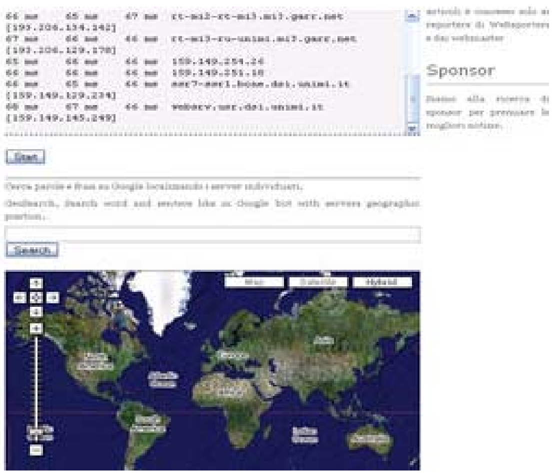
Cell Phone Reception: Cell towers by location

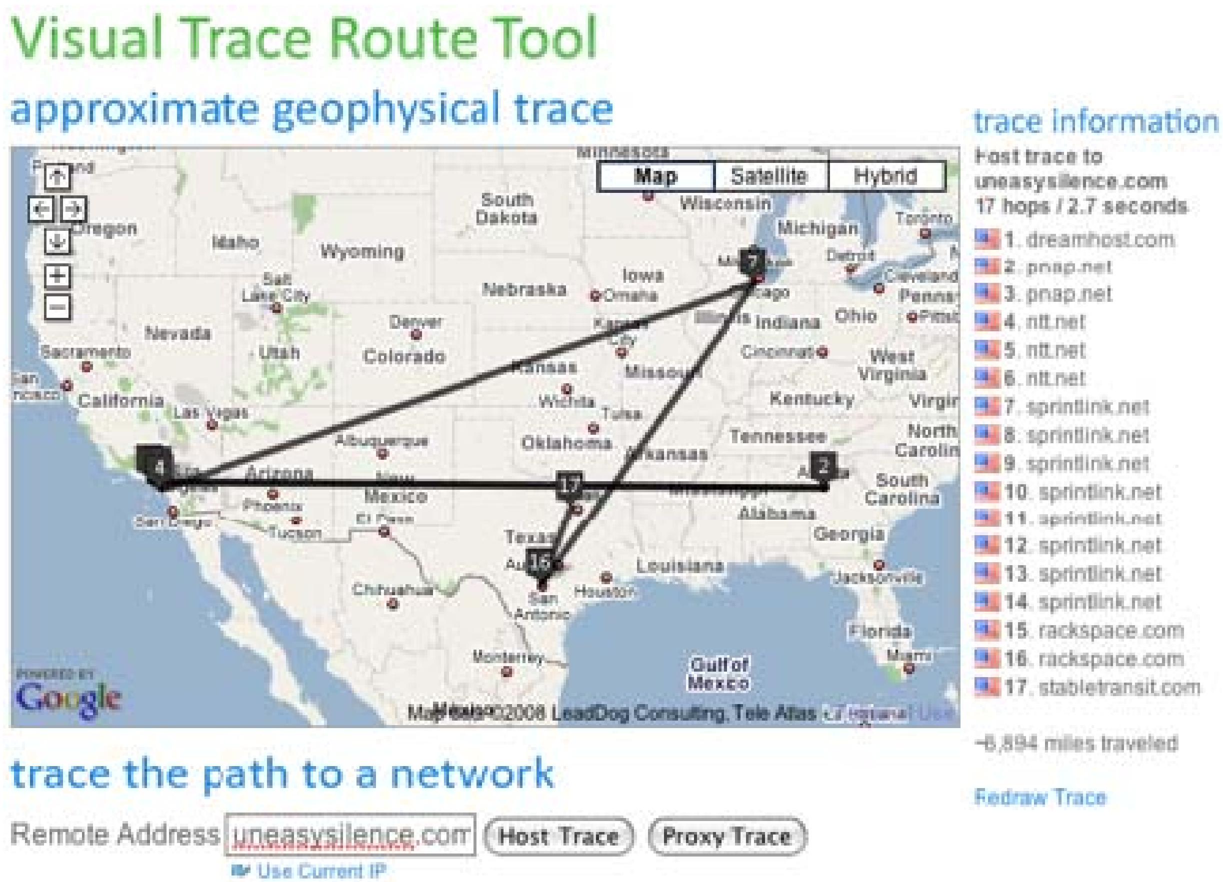
- Combining
  - *Various telecom sites*
  - *GoogleMaps*

Other Interesting Mashups…

Visual Traceroute: Show the tracert command

- Combining:
  - *TraceRt*
  - *DNS*
  - *GoogleMaps*





Research Questions..

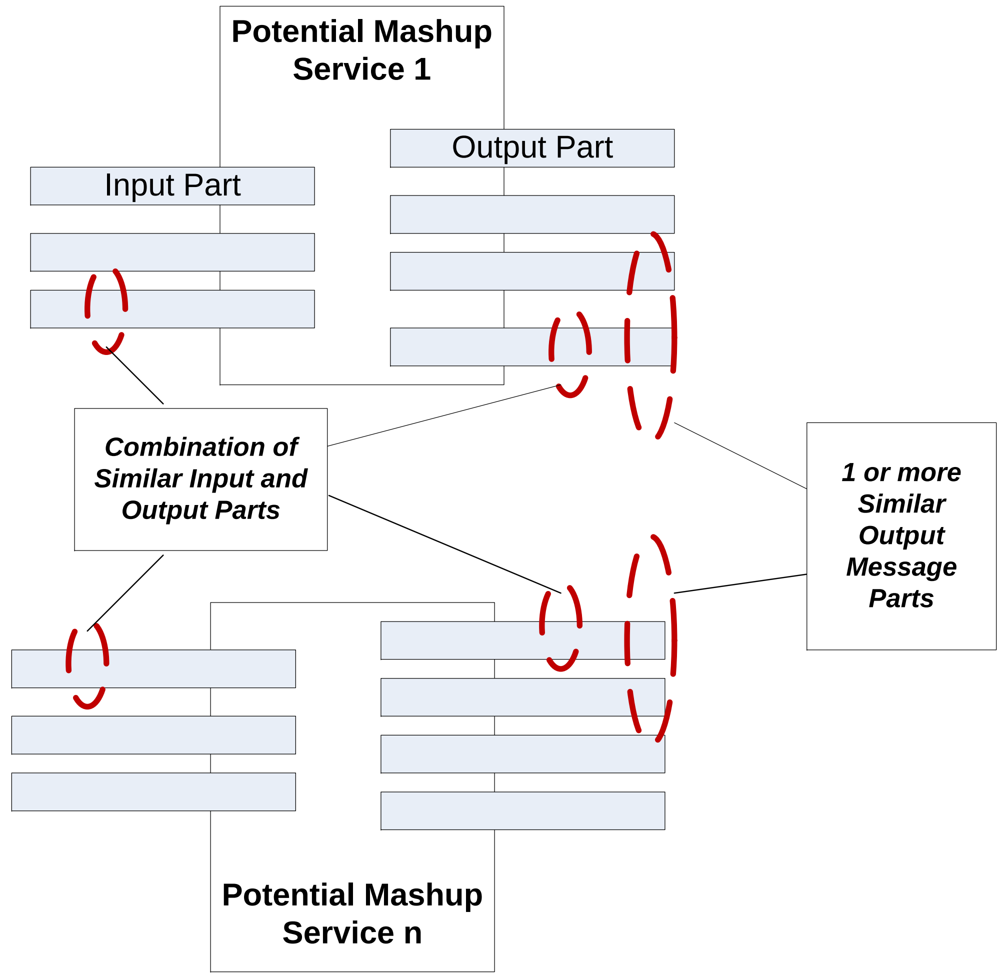
- Considering open web services over the Internet, services in a federated registry, or even services in a intranet-based repository….
  - *What are the common characteristics of two services that make them qualified for mashup?*
  - *What are the relations between the messages of such services?*
  - *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*

Straightforward Technical Approach…

- 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

$Mash(OP_1, OP_2)$:	Mashup Prediction Function
$TSM\text{-}LP(Pn_1, Pn_2)$:	Similarity Function (Section 3)
OP_X	Web Service Operation
Pn_X	Message Part
match	Number of Similar Matches
size	Number of parts in an operation

```
Mash(OP1 , OP2 )
    forAll(Pn1)
      forAll(Pn2 )
         if(TSM-LP(Pn1 , Pn2 ))
            match++
            break
      endFor
   endFor
   if(match / OP1.size < .75)
      return true
   else
      return false
```

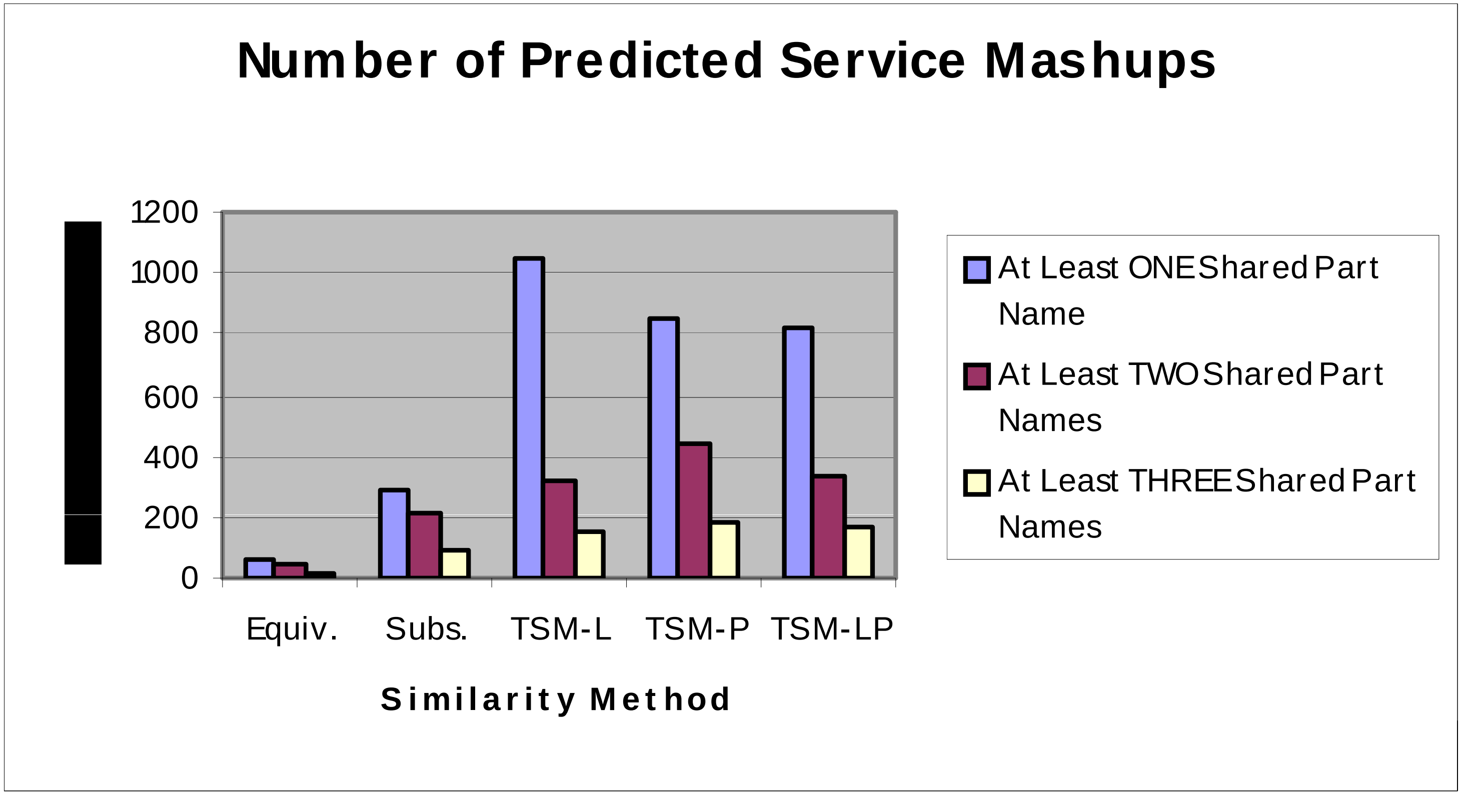
- Evaluate multiple web services for similar message parts
  - Disregard services that have two many parts in common
  - New Work:
    - *Gather insight from Web2.0 sites*
    - *Use congenial services more frequently in prediction*

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*

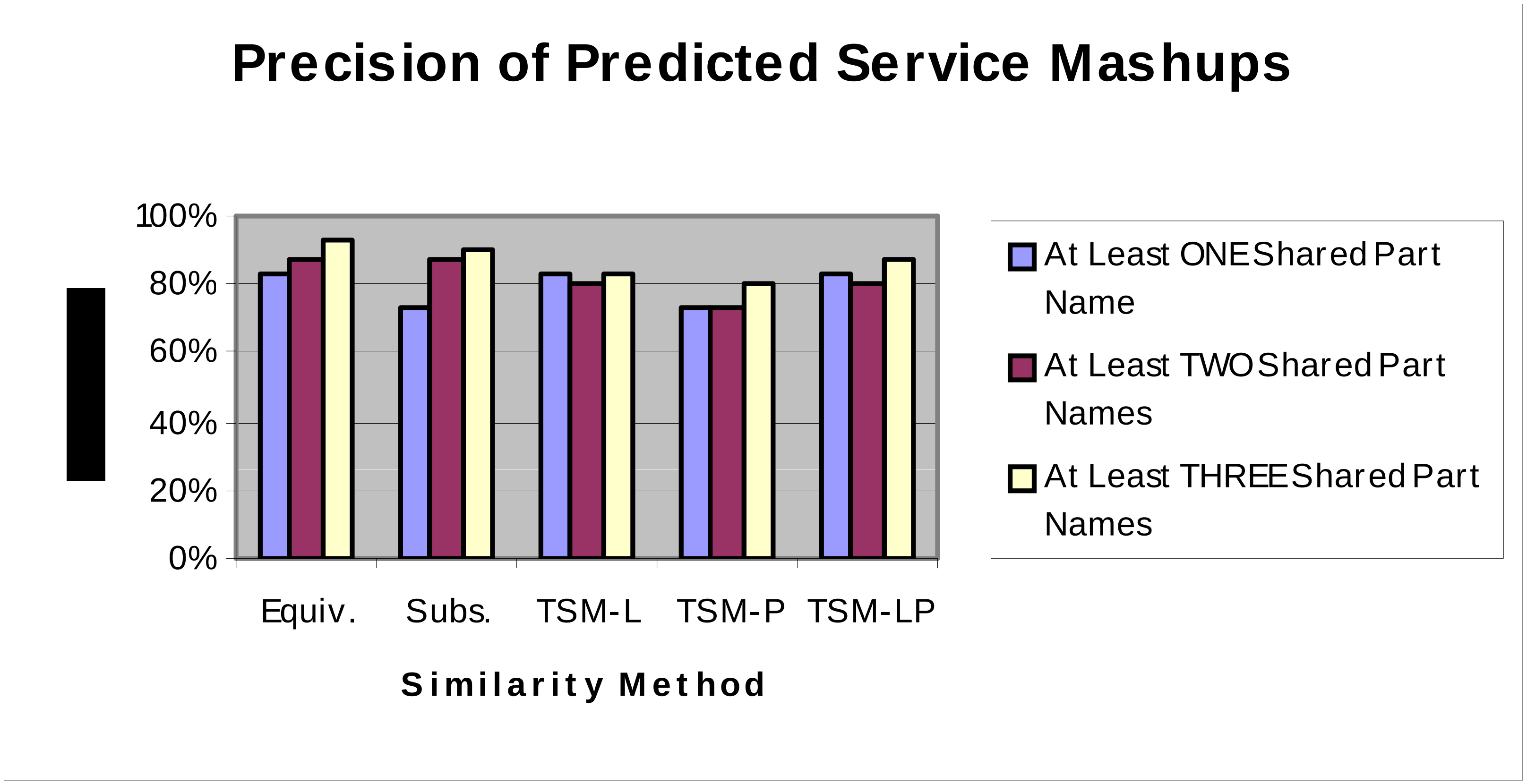
- *Visual inspections were used to determine precision and recall which required smaller experimental sets.*

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

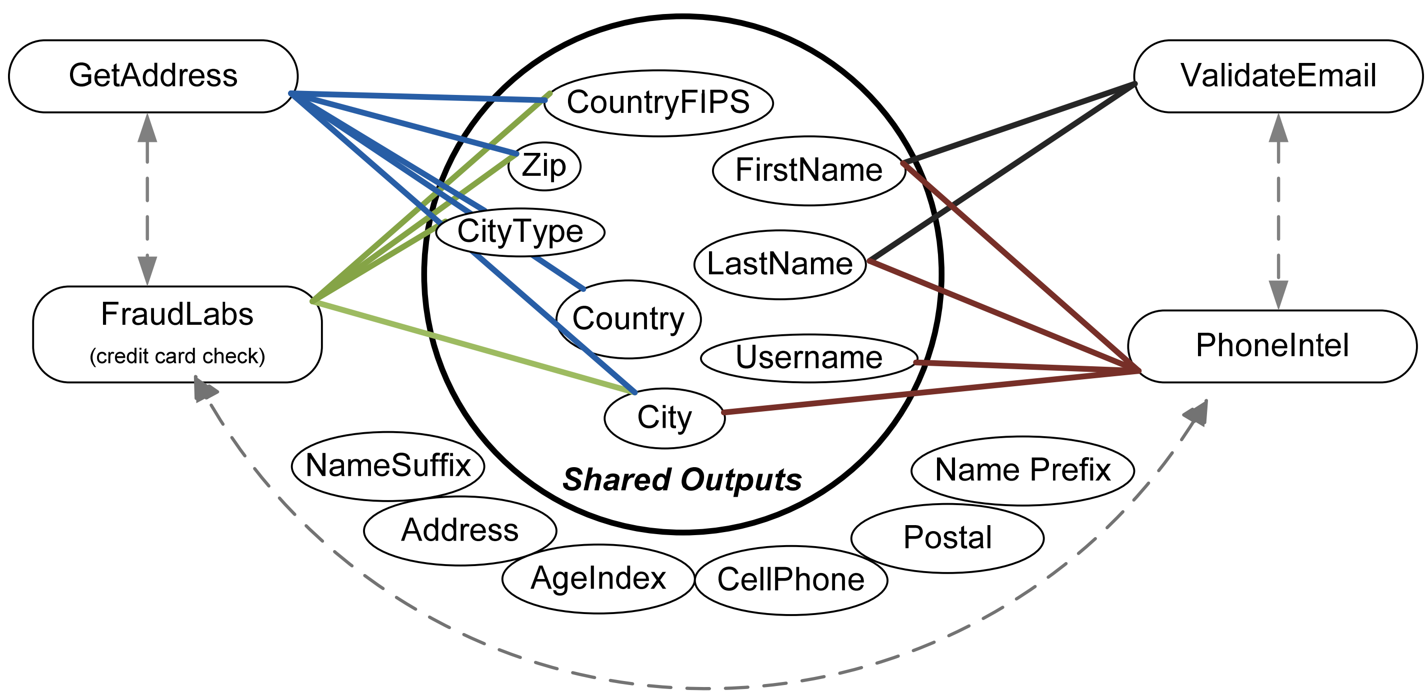
Precision of Predicted Mashups



Although there is a gain in precision with more stringent requires, that gain only varies 5-15% which is not proportional to reduction in total predictions.

Other Results: Sample Mashup and Most Commonly Correlated Messages

Top 6 Most Common Message Parts for Predicting Mashups	*Percentage of Top 6 Used for Predictions*
State, City, Name, Date, Time, Zip	29%



Summary and Future Work

- Syntactic matching applied to similar outputs can be a 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

Recent Projects & Conclusions

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….

PostDocs (Jan '08)
Ajay Bansal,
PhD, UT-Dallas

Srividya Kona
PhD, UT-Dallas

Graduate Students
ImanMoustafa
CS, PhD Student
Virginia Tech

Ahmed Hamza
CS, Master's Student

Mustafa Dustani
CS, Master Student

Michael Lefebvre
CS, Master Student

Khaled El-Goarany
CS, MS Student
Virginia Tech

Undergraduates
Michael Nowlan,
Senior, CS

Brian Miller,
Sophomore, CS

Ryan Butler,
Senior, CS

Alex Yale-Loehr
Freshman, CS

Undergraduates (non-CS)
Erik Muller
Senior, Business

- 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

Service-Oriented Computing

Component Workflows for Distributed Data Management

AnnalsSE2002, SEKE 2003, SPE 2005, IJITWE 2006

Agent-Based Workflow Management of Distributed Components

Agents2000, ISADS 2001, CoopIS2002, IJAIT2003

Software Engineering Training for Distributed Group Projects

CSEET 2002,IEEE TransEdu2003, IEEE Trans Edu2005, ASEE2005, IEEE Computer 2006

Agent-Mediated Training

ICALT 2006, AAMASWkshp 2007, IJAIED2009

Service Composition

AAMAS2003, IEEE TKDE2005, ISEB 2003, DSS 2005, IJWSR2007, ICWS 2008

Service-Based Discovery, Recommendation, & Management

WWW 2005, ICWS 2006, DAPD 2007, MAGSJ 2007, ICWS 2007 IEEE Internet Computing 2007, IEEE Internet Computing 2008, SCC 2008 ACMTWEB (pending), IEEE TKDE (pending)

Service-Oriented Software Engineering

WETICE 2002, SELMAS 2003, ICWS 2006, IEEESoftware 2007, SCC 2007, IJSEKE 2008

- 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