



Almaden Services Research

## Semantic Web Services for Activity-Based Computing

E. Michael (Max) Maximilien  
<http://maximilien.org>

IBM Almaden Research Center  
Almaden Services Research  
San Jose, CA  
USA

IEEE SCC, Chicago, IL, USA

September 19<sup>th</sup>, 2006 – © E. Michael Maximilien

Almaden Services Research



## Agenda

- **Motivations**
- **Approach**
- **Use case**
- **Ontologies**
- **Architecture**
- **Implementation**
- **Discussion**
- **Future research**
- **Reality checks and sneak peak**

E.M. (Max) Maximilien, <http://maximilien.org>

September 19<sup>th</sup>, 2006

## Motivation

- **SWS**
  - Annotate Web services with semantic information
  - Facilitate usage of services, i.e., automatic (or semi-automatic) discovery, selection, invocation, and composition of Web services
  - In line with the vision of the semantic Web
- **Activity-Based Computing (ABC)**
  - Representation for human activities or *every day processes*
  - Embedding contexts in activity representation as semantic information
  - ABC thesis is that human realization of work is loose, malleable, and flexible
- **Taking advantage of semantics – “research questions”**
  - How can the semantics of activities be reflected in exposed services?
  - How can activity-based widgets and applications be enhanced using the semantics?

## Agenda

- **Motivations**
- **Approach**
- **Use case**
- **Ontologies**
- **Architecture**
- **Implementation**
- **Discussion**
- **Future research**
- **Reality checks and sneak peak**

## Approach

- **Our approach and driving principles**
- **Don't fight heterogeneity, embrace it**
  - Annotate the human tasks rather than the resources
  - Allow humans to naturally come up with agreements
  - Learn from human tasks and then attempt to automate
- **Activities as services**
  - API for building activity-based solutions
  - Integrate other services into activity-based solutions
- **Platform for sharing common activities**
- **Tooling to allow humans to discover new capabilities and integrate into solutions**

## Agenda

- **Motivations**
- **Approach**
- **Use case**
- **Ontologies**
- **Architecture**
- **Implementation**
- **Discussion**
- **Future research**
- **Reality checks and sneak peak**

## Use case

- **Reading group activities**
  - Common, shared activity among knowledge workers (or students)
  - Participants share reading items
  - Domain ontology is *Reading Documents* and *Reading Group Activity*
  - Implicit goal is to gather knowledge from group that otherwise individual would not realize or discover
- **Integrating discovered properties**
  - Availability of Web services in the domain, e.g., Amazon's E-Commerce Services (ECS) 3.0
  - Adapt Amazon ECS 3.0 with domain ontology – create Amazon SWS
  - Dynamically discover new information to enrich Reading Group Activity applications and widgets

## Agenda

- **Motivations**
- **Reality checks**
- **Approach**
- **Use case**
- **Ontologies**
- **Architecture**
- **Implementation**
- **Discussion**
- **Future research**
- **Sneak peak**

## Ontologies

- **Activity**

- Unifying concept for things that occur in domain
- Recursive and fractal in nature
- Has a collection of *Artifacts* and involves some *Actor(s)*

- **Artifact**

- Represents all *non-active* elements of an activity
- Contains references to some resource via the resource's URI

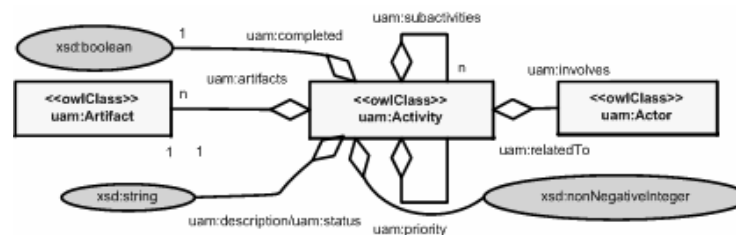
- **Actor**

- Models the *active* element of an activity
- Can be either human or software

## Ontologies (cont.)

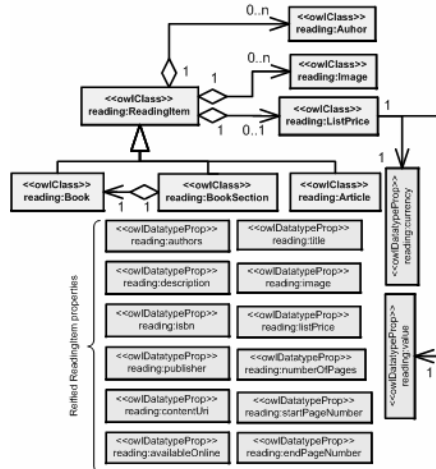
- **Upper ontology for activities**

- Represented in OWL
- UML diagram showing key concepts and relationships



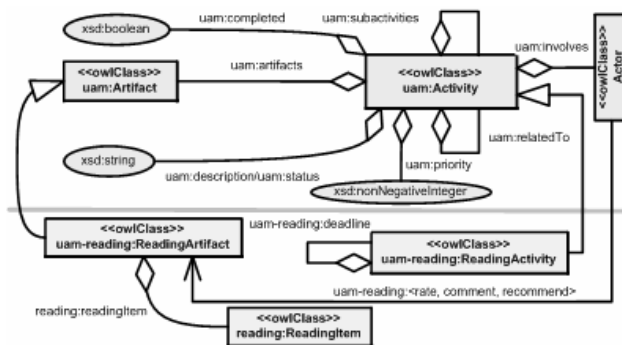
## Ontologies (cont.)

- **Reading documents ontology**
  - Simplified
  - Full document ontology is out of scope
  - Enough information for interesting widget and SWS
- *Reading Item*
  - title, publisher, description, ...
- *Book, BookSection*
  - isbn, numberOfPages, ...
- *Article, WebArticle*
- *Author*
  - firstName, lastName
- *Image*
  - resourceUri
- *ListPrice*
  - currency, value, formattedPrice



## Ontologies (cont.)

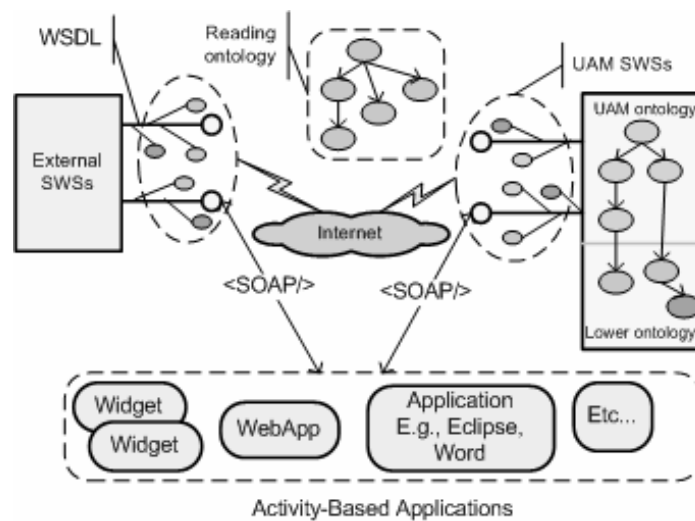
- **Reading group activities**
  - Extends activity ontology
  - Incorporates Reading Document ontology
- **Key concepts**
  - *ReadingActivity*
    - *readingArtifacts*
  - *ReadingArtifact*
    - *readingItem*



## Agenda

- Motivations
- Reality checks
- Approach
- Use case
- Ontologies
- **Architecture**
- Implementation
- Discussion
- Future research
- Reality checks and sneak peak

## Architecture – Overview





## Agenda

- Motivations
- Approach
- Use case
- Ontologies
- Architecture
- **Implementation**
- Discussion
- Future research
- Reality checks and sneak peak

## Implementation – Reading Group widget example

- **User interface**
  - Simple *ReadingActivity* list
  - Configure *Actor* and SWS server
  - Ability to add and remove *ReadingArtifacts*
  - Discovery of properties
- **Implementation**
  - Jython (Python in Java)
  - Leverages Python's scripting prowess
  - Leverages Python's meta programming facilities, OO support, and general *coolness* ☺



## Implementation – Matching algorithm sketch

```
[...]
hServices = ... # list of domain Hailmary services
services = ... # get from registry of SWS (currently fixed)
for s in services:
    if s.owlS.profile.sParameter.uri == self.domainOnto:
        props = self.matchProperties( s, hServices )
        self.displayProps( props ) # displays DiscoveredProp
[...]

def matchProperties( self, service, hServices ):
    props = {}
    for s in hServices:
        self.findMatchingProperties( service, s, props )
    return props
```

## Implementation – Matching algorithm sketch (*cont.*)

```
def findMatchingProperties( self, service, hService, props ):
    for atomicProcess in service.atomicProcesses:
        matched = 0
        inputs = atomicProcess.inputs
        params = []
        for output in hService.profile.outputs: # decomposed output
            if self.matchInsOut( inputs, output ):
                inputs.remove( output )
                params.append( output )
        if len( inputs ) == 0:
            dProp = DiscoveredProp(atomicProcess, params, hService)
            props[ atomicProcess.name ] = dProp
```

## Implementation – Matching algorithm sketch (*cont.*)

```
def matchInsOut( self, ins, out ) :  
    for i in ins :  
        if out.class == i.class or i.class.subsumes( out.class ) :  
            return 1  
    return 0
```

### ▪ Example Matching

#### – BookService SWS

- BookService.getCoverArt( Book.isbn ) : Image
- BookService.getListPrice( Book.isbn ) : ListPrice

#### – Hailmary SWS outputs

- FindReadingWs.findBook(...) : Book
- CreateReadingActivityWs.createBook( ... ) : Book

## Agenda

- Motivations
- Approach
- Use case
- Ontologies
- Architecture
- Implementation
- Discussion
- Future research
- Reality checks and sneak peak

## Discussion – Other use cases and domains

### ▪ **Asset-based IT services**

- Service engineers build service solutions using assets
- Asset ontology includes: *HardwareAsset*, *SoftwareAsset*, *PeopleAsset*, *DocumentAsset*, and so on
- Services to match
  - Software asset service, e.g., [sourceforge.net](http://sourceforge.net)
  - Technical document repository service, e.g., [freshmeat.org](http://freshmeat.org), [oreilly.com](http://oreilly.com), and so on
  - Social network sites for developers, e.g., [slashdot.org](http://slashdot.org)

### ▪ **Other use cases**

- Activities of professional knowledge workers
- Insurance, sales, healthcare, and so on

## Discussion – Limitations

### ▪ **Domain ontology**

- Exists and in use
- This is a big assumption
- Unrealistic?

### ▪ **Scalability**

- How to discover multiple SWS in one domain?
- How to display many discovered properties?
- Ranking discovered properties?

## Discussion – Limitations

- **Complete dynamic SWS generation**
  - Domain ontology and operator ontology (or partial OWL-S *ServiceProfile* definition)
  - Generate complete code and descriptions for SWS
  
- **Back-end is currently a flatfile**
  - Use a RDF DB-based back end with support for transactions and so on
  - Other databases?
  
- **Support for query language, i.e., XQuery, in find<Xyz>(…) services**

## Agenda

- **Motivations**
- **Approach**
- **Use case**
- **Ontologies**
- **Architecture**
- **Implementation**
- **Discussion**
- **Future research**
- **Reality checks and sneak peak**

## Future research

### ▪ WSDL-S

- Simpler and more pragmatic SWS description, uses WSDL extensions
- Good tool support (Eclipse plug-in)
- Ontology language agnostic (supports OWL and UML)
- Missing standard service ontology?

### ▪ Other use cases and examples

- Implement other use cases
- Web application example
- Other applications or widgets

## Future research (*cont.*)

### ▪ Matching algorithm

- *AtomicProcess* chaining to generate some output?
- *CompositeProcess* or some BPEL-like composition language?

### ▪ Is there a hybrid, ontology and *folksonomy* (i.e., <http://del.icio.us>, Yahoo! Flykr), approach that may work to create domain ontology?

- Address limitation of having existing and agreed ontology
- Enables some level of automation and captured users "crowd wisdom"

### ▪ Tooling

- Assist human designer in service composition
  - Data and field mediation
  - Protocol mediation
- Assist human designer in sharing resulting high-level composition

## Agenda

- **Motivations**
- **Approach**
- **Use case**
- **Ontologies**
- **Architecture**
- **Implementation**
- **Discussion**
- **Future research**
- **Reality checks and sneak peak**

## Reality checks

- **Heterogeneity is central to the Web's success**
- **Simple interaction model, e.g., HTTP**
- **Humans create sites and applications**
- **Services (or APIs) are exposed with XML**
  - WSDL,
  - REST,
  - XML-RPC
  - RSS
- **Services are primarily software agents' consumption**

## Reality checks

### ▪ Problems

- How to facilitate service usages?
- Can we achieve automatic service composition? E.g., “I want a service that finds all events in San Jose and puts them on a visual map?”

### ▪ Current SWS approach

- Agree *a priori* on common semantics
- Use common semantics as a means to help automate service programming and usage
- Program agents using common semantics
- Agents, automate integration tasks? acquire goals from humans?

## Reality checks 2.0 – return to pragmatism

### ▪ The Web version 2.0

### ▪ Humans manually (and painfully) create composed services (or mashups) via various mediation tasks (or activities)

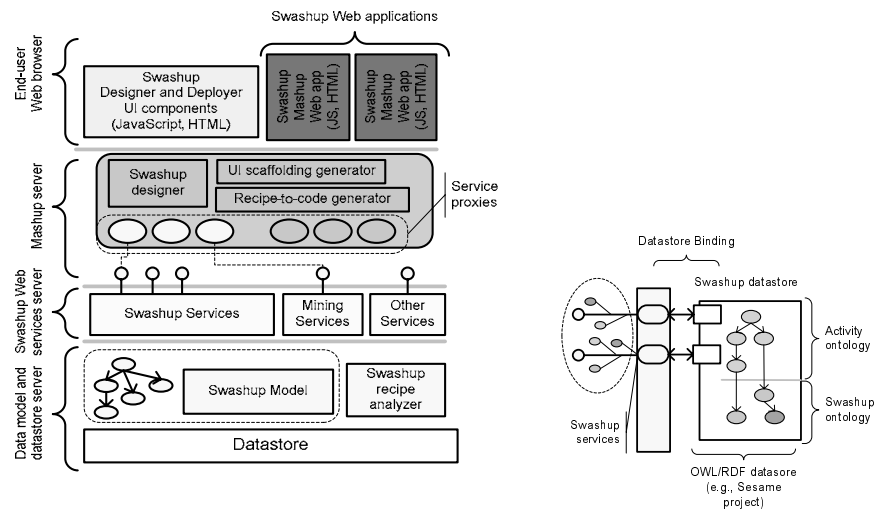
- [housingmaps.com](http://housingmaps.com), [podbot.com](http://podbot.com),
- [sforce.com](http://sforce.com) GoogleMapSControl,
- Yahoo! <http://del.icio.us>, Yahoo! <http://flykr.com>

### ▪ Humans tag resources to help catalogue, search, and resolve semantic issues

- Results in *folksonomies*

### ▪ Dynamic, clean, and highly interactive UIs (using AJAX)

## Sneak peak: Swashup<sub>ALPHA</sub> Architecture overview



## Sneak peak: Swashup<sub>ALPHA</sub> address mapping example

- **MapService**

```
map( title, Address ) : void
```

- **AddressBookService**

```
findAddress( searchString ) : Address
```

```
addressAsString( Address ) : String
```

- **Simple mashup**

- Search
- Convert address to string
- Map

# Sneak peak: Swashup<sub>ALPHA</sub> Address mapping (cont.)



E.M. (Max) Maximilien, <http://maximilien.org>

September 19<sup>th</sup>, 2006

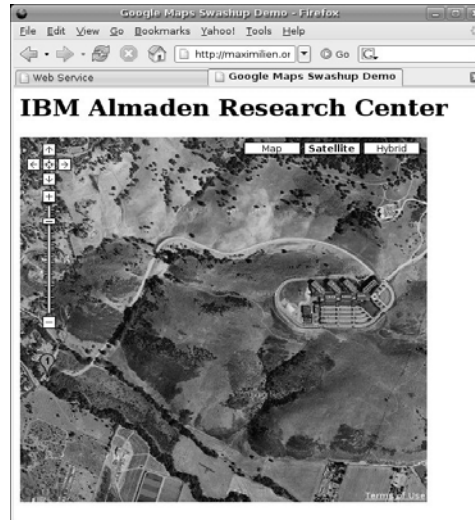
# Sneak peak: Swashup<sub>ALPHA</sub> address mapping (cont.)



E.M. (Max) Maximilien, <http://maximilien.org>

September 19<sup>th</sup>, 2006

## Sneak peak: Swashup<sub>ALPHA</sub> address mapping (cont.)



## References

Maximilien, E. M., et al. “Semantic Web Services for Activity-Based Computing”, In Proceedings of ICSOC 2005

Moran, T.P., et al. “Unified Activity Management”, Communications of the ACM, Dec. 2005

Sycara, K., et al. “Automated Discovery, Interaction, and Composition of Semantic Web Services”, Journal on Web Semantics, Sept. 2003

Berners-Lee, T. et al. “The Semantic Web”, Scientific American, May 2001

OWL-S 1.0, <http://www.daml.org/services/owl-s/1.0/>

WSDL-S, <http://www.w3.org/Submission/WSDL-S/>

Amazon ECommerce Services, <http://www.amazon.com/aws>

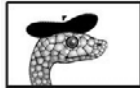
# Acknowledgements



<http://rubyonrails.org>



<http://eclipse.org>



<http://python.org>



<http://protege.stanford.edu>



धन्यवाद  
Hindi

多謝  
Traditional Chinese

ขอบคุณ  
Thai

Спасибо  
Russian

Gracias  
Spanish

شكراً  
Arabic

Thank You  
English

Obrigado  
Brazilian Portuguese

Grazie  
Italian

多谢  
Simplified Chinese

Danke  
German

Merci  
French

நன்றி  
Tamil

ありがとうございました  
Japanese

감사합니다  
Korean