# Intelligent Systems on the World Wide Web

# 4b Dynamic Services

Lecture Slides
Steffen Staab
Karlsruhe University
With Acknowledgements to
Sheila McIlraith, Stanford University &
David Martin, Stanford Research Institute, Menlo Park



## Queries, Requests, Requirements

"Find information about the researcher James Hendler."

- Can almost be done with keyword search
- Actually "researcher" is not used by keyword search
- Only slight advantage for Semantic Web
- Consider "find computer scientist with first name Steffen" (google rank for Steffen to Steffen Staab, 31; worse for "Steffen and computer scientist")





## Queries, Requests, Requirements

"Find a paper on SHOE with James Hendler as a co-author."

- All concepts identified unambiguously (by URIs)
- SHOE -> many sites on shoes





## Queries, Requests, Requirements

"Find a reference to the latest paper on SHOE with James Hendler as a co-author."

- All concepts identified unambiguously (by URIs)
- Ontologies potentially (likely) distributed
- Reasoning about time, etc.
- "Latest", etc.: no closed-world assumption

## Queries, Requests, Requirements

"Find a reference to a paper having 2 authors, both of which have been PIs for a DARPA project."

- Meta-data for a single response is (very likely) distributed





## Queries, Requests, Requirements

"If Daniel Dennett has published a book about consciousness, request the Stanford Library to hold it for me."

- Combine information queries with service requests





## Queries, Requests, Requirements

"Get me a flight to Washington DC, and reserve a room near the airport"

- Composition of services



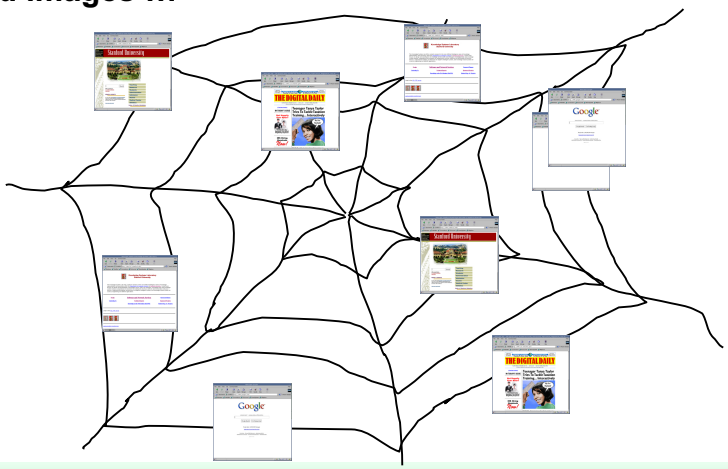


## Queries, Requests, Requirements

"Execute the refinancing of my mortgage, using the following parameters and providers: …"

- Multiple participants in a single transaction (bank, life insurance, land register,…)

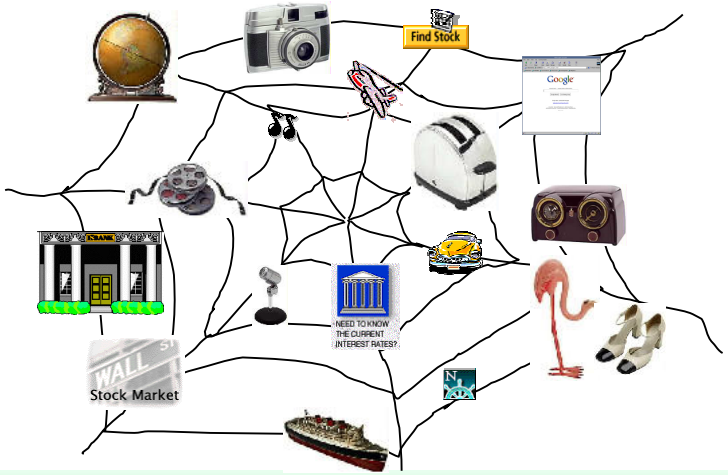
The Web is evolving from a provider of **documents and images …**







… to a provider of **services**








## Services

- Information providing services
  - Current stock rates
  - Text documents
  - Etc.
- Services with effects on the world
  - Booking flights
  - Buying books
- Complex services
  - Travel package: flights booked, travel guides inclusive, accomodation reserved
  - Mobile package: arrange test drive of car, buy car, financing of car, maintaining the car
  - Etc.??





**Problem:** today's Web is designed primarily for human use.

1) Now: Developing **semantic markup** of
   - Web service properties and capabilities,
   - users' preferences and constraints,
   - reusable Web agent procedures

   to make services *computer-interpretable, use-apparent,* and *agent-enabled.*

2) Now: Developing **agent technology** that exploits semantic markup to support automated Web service *discovery*, *execution*, *composition and interoperation*.

## Web Service Markup Will Enable

Automation of:

- Web service discovery
  *Find me an airline service t… …Tuktoyuktuk*
  Markup: declarative …
- Web service execu…
  *Buy me "Harry P… …orcerer's Stone" at www.amazon.com*
  Markup: declarative API (inputs & outp…
- Web service selection, composition… …ation
  *Make the travel arrangements … …10 conference*
  Markup: declarative spec. of as… …onditions & effects)

Industry efforts growing (we are improving with DAML)

Opportunity Area

These are the **drivers** for Web service markup.





## 1) Semantic Markup: DAML-S (initiated 02/2001)

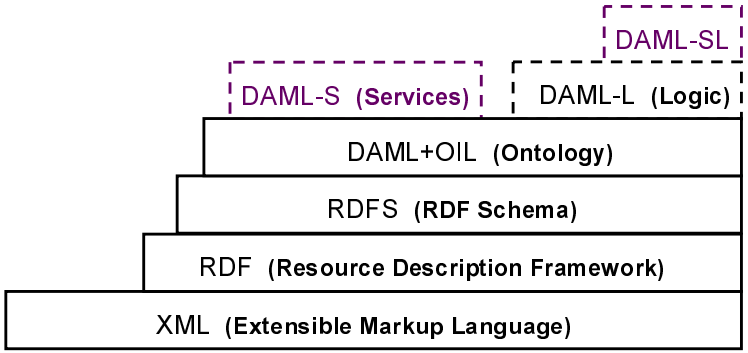
DAML-S: A DARPA Agent Markup Language for Services

- AI-Inspired markup language for Web services:
  - well-defined semantics
  - ontologies support reuse, mapping, succinct markup, ...
- Developed by a coalition of researchers from Stanford, SRI, CMU, BBN, and Nokia, under the auspices of DARPA.
- Pre-release of DAML-S version 0.5 (May, 2001)
- Watch http://www.daml.org/services/daml-s/ for details





## Layered Language Development




[Fensel+others, 2000]





## Semantic Markup of Web Services

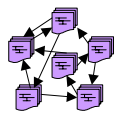
Each Web site provides a **set of services**

- information providing *(e.g., flight schedules, camera views)* &/or
- world-altering *(e.g., flight booking, home temp. adjustment)*

Web service sites are annotated with DAML markup.

Service described as **processes** (sequence, if-then-else, while, iteration, …) of

- inputs & outputs (**function** metaphor)
- preconditions & effects (**action** metaphor)



**DAML-enabled web pages**

## Key Feature: Function/Dataflow Metaphor

AIFB

**Input:**

- customer name
- flight number
- credit card
- …

www.acmeair.com
book flight service

**Output:**

- confirmation no.
- …

flight available + valid credit card ?

Y

N

- failure notification
- …







## Key Feature: AI-inspired Action Metaphor

AIFB

**Output:**

- confirmation no.
- …

**Input:**

- customer name
- flight number
- credit card
- …

**Effect:**

- ticket purchased
- credit card debited
- …

www.acmeair.com
book flight service

flight available + valid credit card ?

Y

N

**Preconditions:**

- knowledge of the input
- own credit card
- …

**Output:**

- failure notification
- …

**Effect:** <no effect>







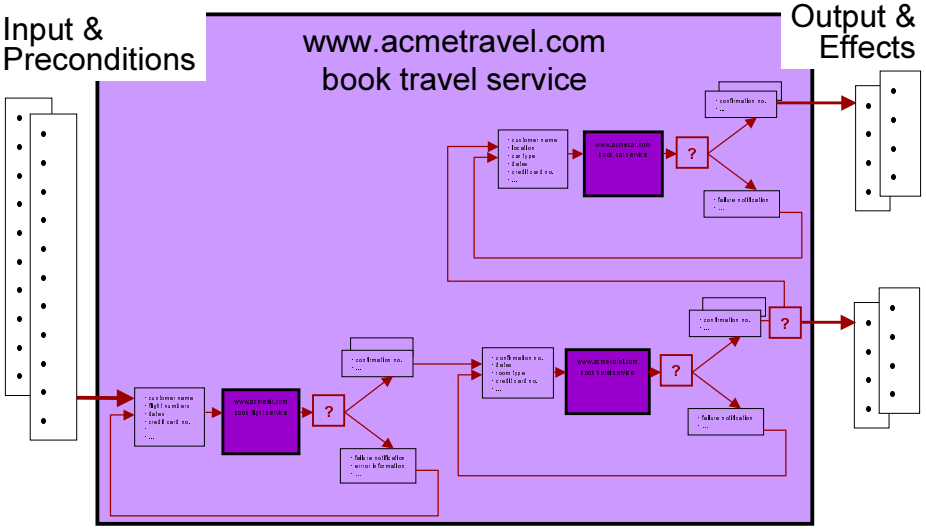
## Key Feature: Process of Functions & Actions

AIFB

Input & Preconditions

www.acmetravel.com
book travel service

Output & Effects









## Web Services Markup Exploits Ontologies

AIFB

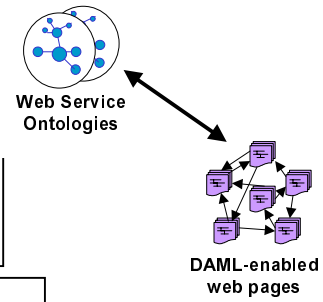
Ontologies support sharing, reuse, succinct markup:

**action**

**purchase**
*<input> customer-name </input>*
*<precond> know(customer-name) </precond> …*

**purchase-plane-ticket**
*<input> flight-no </input>*
*<input> credit-card </input>*
*<precond> know(flight-no) </precond>*
*<precond> own(credit-card) </precond>…*

**purchase-UAL-plane-ticket**
*<effect> purchase(ticket) </effect>*
*<effect> debit(credit-card) </effect> …*




Collectively markup create a distributed KB of services.

Problem: Automated Service Composition

Action metaphor in markup
➔ exploitation of AI technology for reasoning about action

One Approach: **plan** sequences of services that realize user's objective.

Our Approach: reusable **generic procedures** with **customizing user constraints**
Theme: usability and customization





Agents are tasked using high-level, reusable **"generic procedures."**

"The what not necessarily the how."

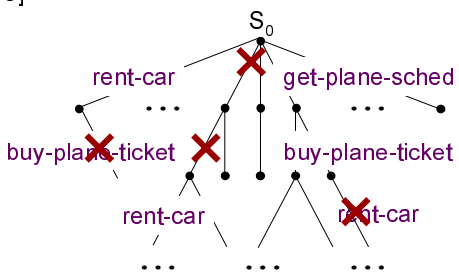
```
E.g.,
Book-transportation(origin,dest,date-d,date-r,purpose)
    pick-one-of
            Book-plane(origin,dest) | Drive(origin,dest) |  Book-train(origin,dest)
    end pick;
end
```

Generic procedures stored in markup ontologies using the same DAML-S like markup.





**Situation Calculus** [McCarthy, 68]
**+**
**Golog** [Levesque et al, 97]
procedural constructs:
- sequencing
- if-then-else
- nondeterministic choice
- while-do, etc.




```
E.g.,
Book-travel(origin,dest,date-d,date-r,purpose)
    pick-one-of
            Book-plane(origin,dest) | Drive(origin,dest) | Book-train(origin,dest)
    end pick;
end
```





Generic procedures can be further constrained by

**DAML-defined user constraints**

- personal constraints/preferences,
- group constraints, or
- instance-specific constraints.

E.g.,
- Bob would like to drive if the driving distance is less than 3 hours.
- KSL business air travel should be on an American carrier.

…

Agent's **KB is automatically constructed** relative to the generic procedure and user constraints.

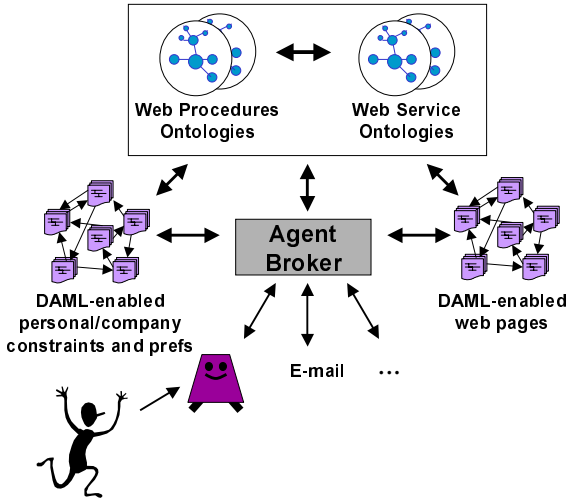
**Deductive machinery** instantiates the generic procedure wrt. constraints and world state to generate Web service requests that the **agent broker** executes.

Middle ground interpreter balances information gathering services with delayed execution of world-altering services.

Agent's KB is updated by Web service responses.





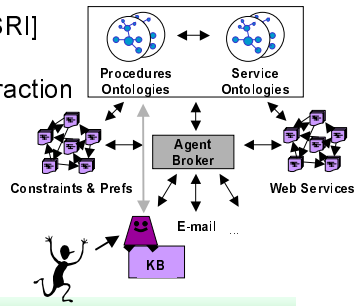






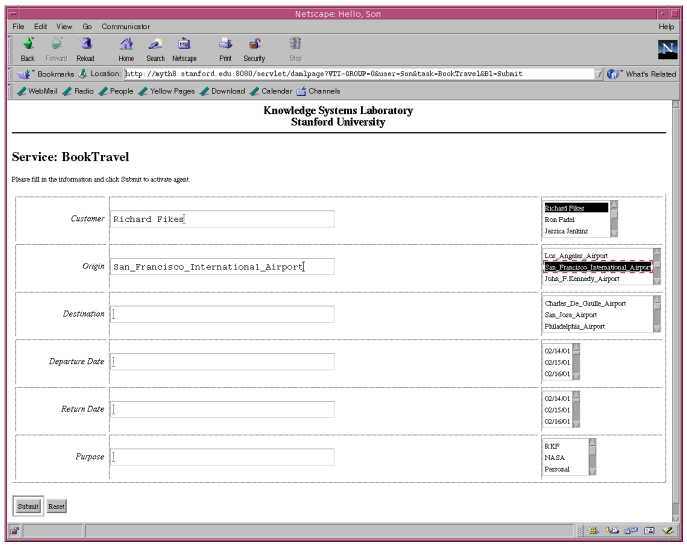

## Status of KSL Architecture

- ✓ DAML = First-order logic and some DAML+OIL (soon to be DAML-S)
- ✓ Agent KB representation language = Situation Calculus
- ✓ Generic procedures representation language = Golog
- ✓ Deductive machinery = ConGolog interpreter written in Prolog
- ✓ Agent Broker = Open Agent Architecture [SRI]
- ✓ Web Service output = HTML + W4 info extraction (eventually DAML)
- ✓ Prototype agent/agent broker constructed.
- ✓ Preliminary service ontology constructed.



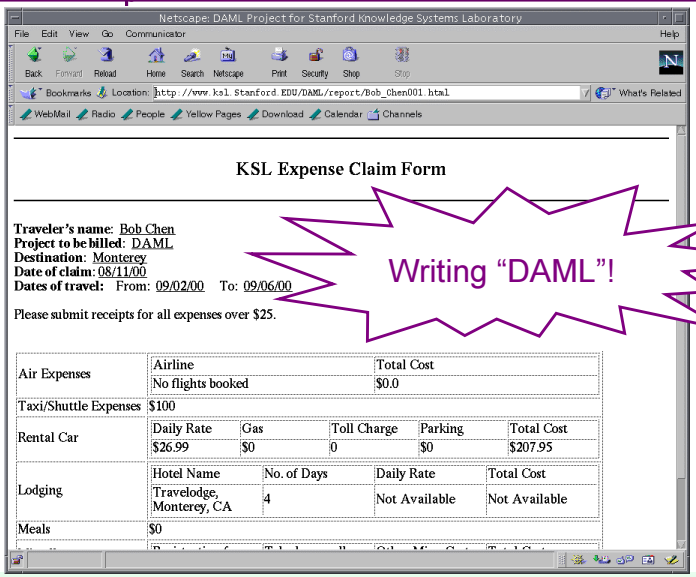




AIFB

## Agent creates expense claim for customer










AIFB

## Summary of KSL Approach

**1) DAML Markup of Web Services, User Constraints, Agent Procedures:**

- Computer-interpretable, use-apparent, agent-enabled services.
- Ontologies facilitate construction, sharing, reuse, and composition; support succinct web site markup.
- Markup not specific to particular agent implementation.

**2) A DAML-Enabled Agent Technology for Web Service Composition:**

- Theme: usability and customization
- Approach: Generic procedures and customizing user constraints
- Deductive machinery instantiates procedures generating web service requests that are sent to the agent broker.
- Procedures & deductive machinery provide middle ground between planning & programming.
- Logic-based approach enables verification wrt. certain properties







AIFB

## Some more details and examples….





AIFB

## Characteristics of DAML

- Based on XML & RDF(S)
  - Beyond RDF: properties of properties, equivalence and disjointness of classes, more constraints, etc.
    Feature comparison: https://www.daml.org/language/features.html
  - Layered approach
    XML => RDF(S) => DAML+OIL => (DAML-L) => DAML-S
- Semantics for Web resources from Knowledge Representation concepts
  - DAML+OIL: can be regarded as a description logic
  - Ontologies
  - Logical rules & inference
- DAML-S: Extension to Services

## Long term goals like…..

"Execute the refinancing of my mortgage, using the following parameters and providers: …"

- **Multiple participants in a single transaction**





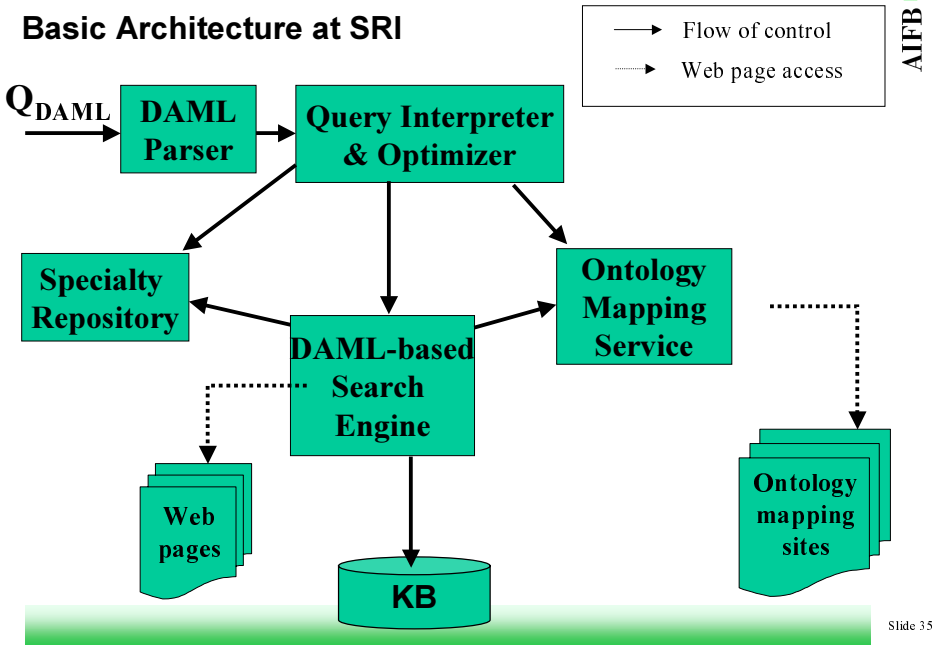
## Requirements

- All concepts identified unambiguously (by URIs)
- Ontologies potentially (very likely) distributed
- Reasoning about time, etc.
- "Latest", etc.: no closed world assumption
- Data for a single response is (very likely) distributed
- Combines information plus service requests
- Composition of services
- Multiple participants in a single transaction
- **Ontology mappings needed**





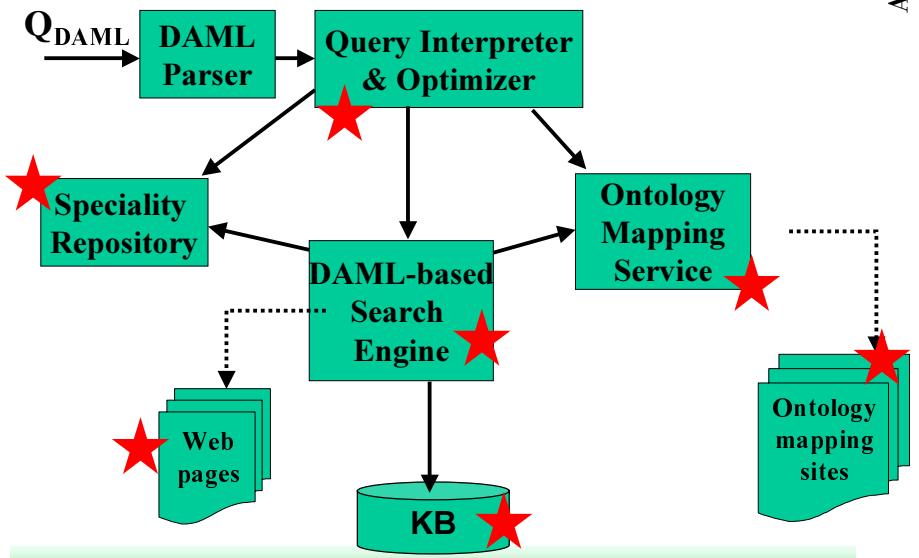
## Basic Architecture at SRI








## Where can logical rules & inference be used?



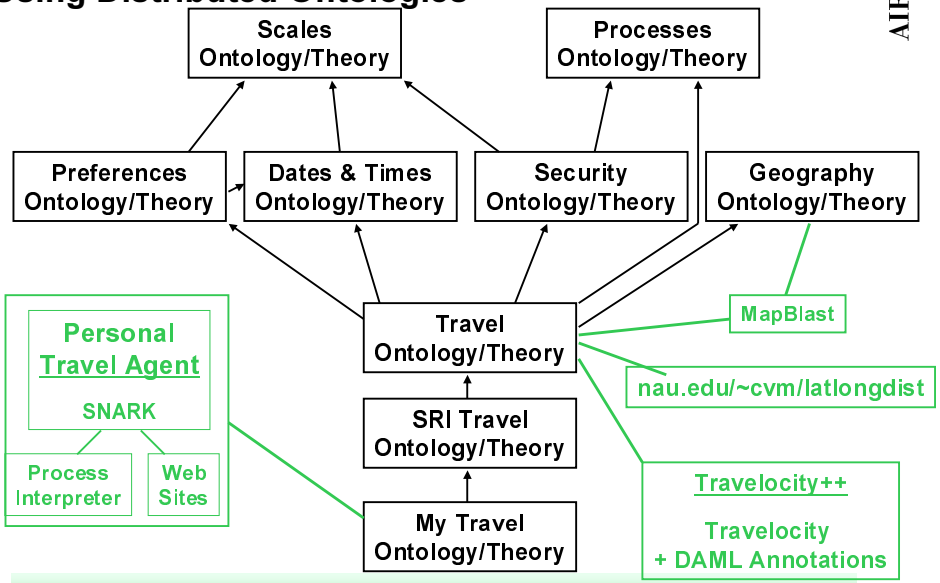
## Rules, Theories (=Ontologies), Inference

- Needed everywhere
  - Class declarations
  - Expressing and answering (decomposing) queries
  - Service descriptions, advertisements
  - Service requests, possibly service bindings
  - Composing services
  - Expressing and using ontology mappings
  - Background knowledge
- Challenge: balance expressive power, high performance, desire for simplicity





## Using Distributed Ontologies






## Linking Ontologies



Travelocity++

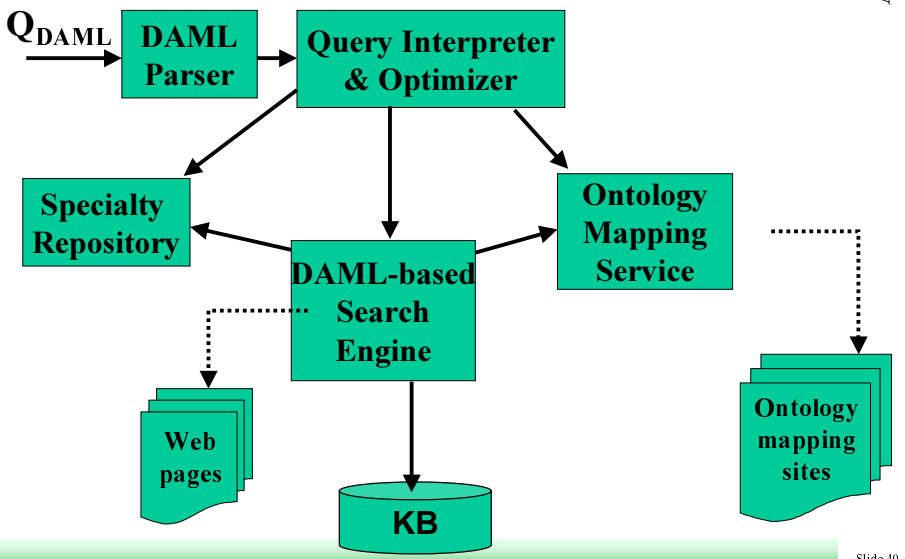
```
Linking Axioms:
   (implies
      (Travelocity:Get_Itinerary ?OrigAPC ?DestAPC ?DepTime ?ArrTime
                ?Flt#  ?Fare)
      (and (DepCity ?Flt (city-of ?OrigAPC)) (DestCity ?Flt (city-of ?DestAPC))
           (DepTime ?Flt ?DepTime) (ArrTime ?Flt ?ArrTime)
           (FlightNo ?Flt ?Flt#)(Price ? Flt ?Fare)))
   .....
Protocol:  OAA, IDL, JINI
```

Personal Travel Agent
SNARK
Process Interpreter
Web Sites
Travel Ontology/Theory
SRI Travel Ontology/Theory
My Travel Ontology/Theory
MapBlast
nau.edu/latlongdist
Travelocity++
Travelocity
+ DAML Annotations

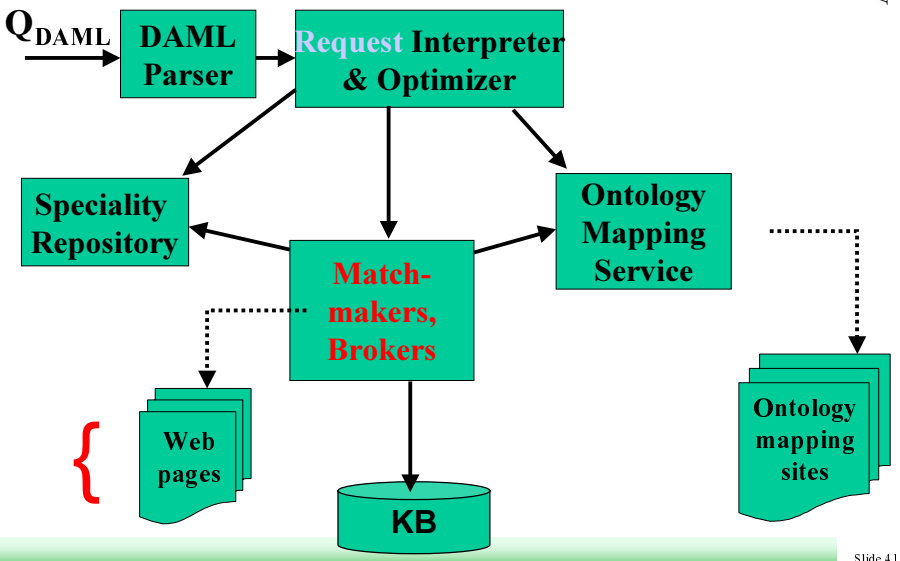




## How do services change the picture?



## How do services change the picture?








## DAML-S: Goals

- Full automation of service use
  - DAML markups provide enough info for an agent to find, select, and use a service never before encountered
- Service requests handled seamlessly with information queries
  - Allow for composition of both
  - Many components & tools can work for both
    - **Search & selection, ontology translation, …**
- Support inference in selecting and using services





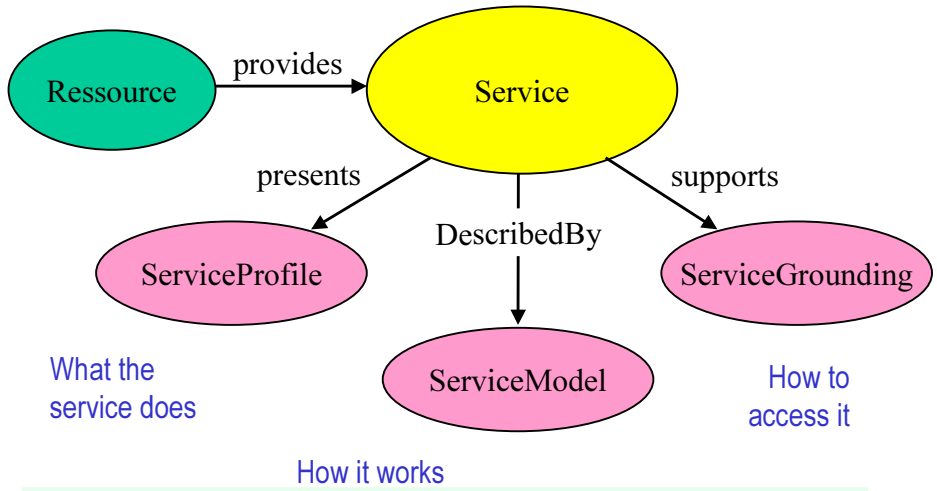
## Upper Ontology for Services

- A foundation for creating many service ontologies
  - But not a mandate for specific ontologies
  - May provide deeper ontologies for Meta-services
- Can be specialized in many different ways
- No one “official hierarchy” of services
  - But agents always know how to get started
  - Top-level specs provide consistency
- Existing taxonomies can be mapped

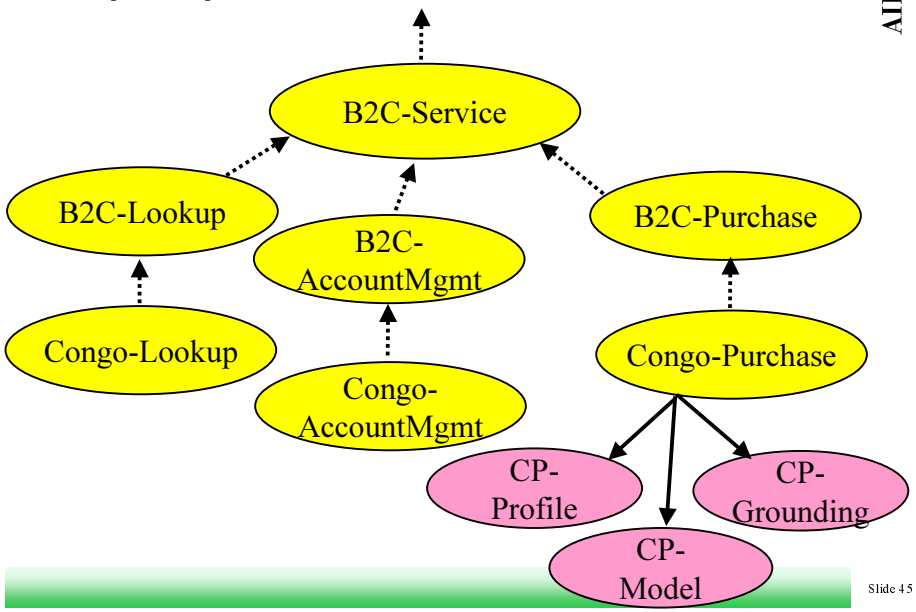




## Service Ontology: Top-level Classes



## Example: Specialization for a B2C Site








## Service Profile

- Requirements for use; results of use
  - "Black box" view: Information needed to find and select a service
  - Inputs, outputs, preconditions, effects, …
  - "Binding rules" for inputs, outputs
  - "Roles" involved
- May vary for different service classes
- Can employ logical rules
- Analogous to procedure header, DB schema





## B2C Purchase: Profile

- Input: ItemDescription (several forms), PriceRange, AcctName, Passwd, CreditCard#, Shipping-address, …
- Input usage rule:
  Exists(Acct) => Defined(CreditCard#, Shipping-Address)
- Precondition:
  Exists(Acct) | CanCreate(Acct)
- Output: 'Succeed' + Receipt | 'Cancel' | 'Fail'
- Effect: 'Succeed' ➔ ShippingOrderPlaced





## Service Model

**"How does it work?"**

- Semantic description of a service
  - "Glass box" view
  - Detailed characterization of what it does
- May vary for different service classes
- Can employ logical rules
- Analogous to procedure body (but abstract)

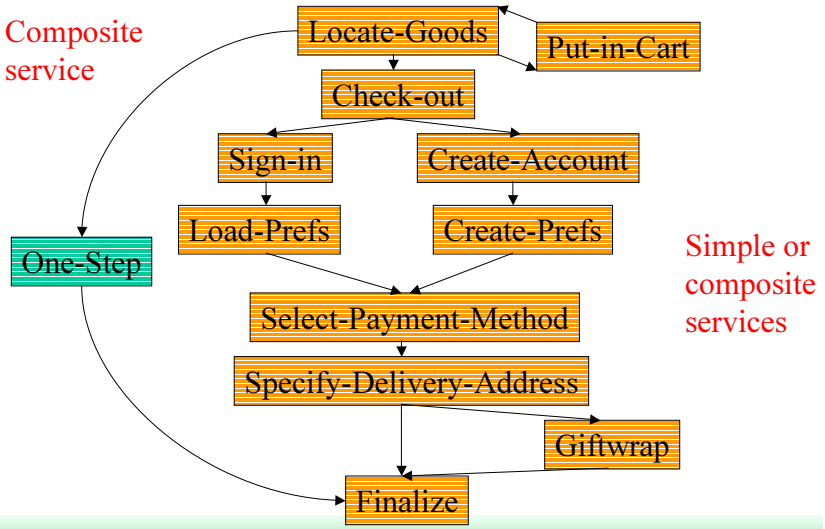
# Process Model
**(SubClass of ServiceModel)**

- Shared knowledge to coordinate service's "roles"
- Expresses state changes
- Describes sequences of possible interactions for an extended transaction
- Executable semantics
- Draws on work in AI planning, workflow, …
- Can be used for task planning, scheduling, reachability analysis, etc.

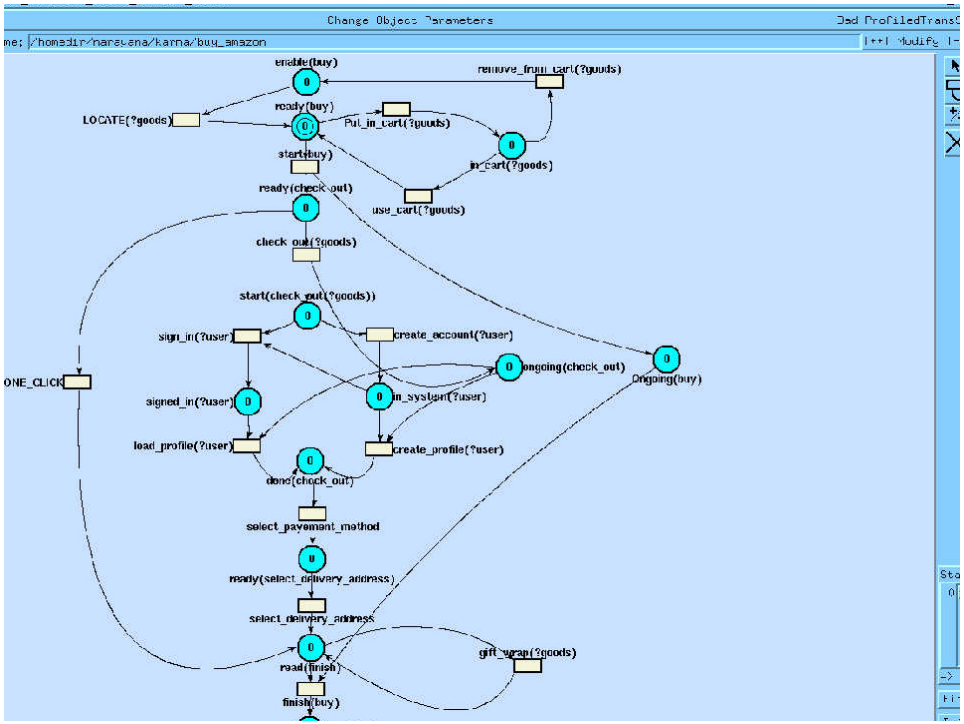




# B2C Purchase: ProcessModel






## CongoPurchaseProcessModel.daml (sketch)



```
<process>
   <name> purchase </name>
   <sequence>
      <process> locate_goods (?goods)</process>
       <alternative>
        <alt1><task> One_step </task></alt1>
        <alt2> <sequence>
          <task> put_in_cart(?goods)</task>
            <process>
              <name> check_out_info</name>
              <alternative>
                <alt1> <sequence>
                   <task> Sign_in (?user)</task>
                    <task> Load_profile(?user.profile) </task>
                </sequence></alt1>
                <alt2> <sequence>
                    <process> Create_Account (?user)</process>
                    <process> Create_profile(?user.profile) </process>
                </sequence></alt2>
              </alternative>
            </process>
          <task> Select_Payment_Method </task>
         <process> Select_Delivery_Address </process>
        </sequence></alt2>
       </alternative>
      <task> Finalize </task>
   </sequence>
</process>
```

## Service Grounding

- Implementation-specific details for accessing the service
- Message formatting, transport mechanisms, protocols, serializations of all types
- Service Model + Grounding give everything needed for using the service
- Examples: HTTP forms, SOAP, KQML, CORBA IDL, OAA ICL, Java RMI





## B2C-Purchase: Grounding

- Transport: Secure HTTP
- Protocol: HTTP Forms
- Address: https://buybot.congo.com:4040/initsub.htm
- Type Serializations
  - ItemDescription (keywords): Set of DAML literals
  - PriceRange: pair of monetary units, ISO 5678
  - CreditCard: https://transcredit.com/S1.daml#SecureTransferFormat





## Recap of Upper Ontology for Services

- Profile supports service selection
- Model + Grounding support execution, monitoring, composition, …
- Profile and Model are abstract; Grounding makes it concrete
- There can be multiple of each
  - One-to-one correspondence not required





## Summary

- The Semantic Web will be big
- It will support a wide variety of (mixed) queries and requests, in a semantically-grounded way
- KB representational techniques, ontologies, axioms, reasoning are likely to be important elements
- Services can be advertised, found, executed, monitored, and composed using DAML-S
- Search engines & portals will evolve; ontology translation services will become essential
- Interesting new challenges for distributed DB/KB technology and Web architecture

## Literature

S. McIlraith et al. 2001 Mobilizing the Semantic Web with DAML-Enabled Web Services. Semantic Web Workshop, Hongkong, May 1, 2001.

Denker et al. 2001. Accessing Information and Services on the DAML-enabled Web. Semantic Web Workshop, Hongkong, May 1, 2001.

http://semanticweb2001.aifb.uni-karlsruhe.de





## Related Work

**Academic Research**

- Golog variants [Levesque et al., 97], [de Giacomo & Levesque, 99]
- SRI work on Web services [Denker et al., 01], [Waldinger, 00]
- University of Washington softbot work [Etzioni et al., 94]
- IBROW3 [Benjamins et al., 98]
- Lark [Sycara et al., 99], etc.

**Industrial Initiatives**

- UDDI
- WSDL
- ebXML
- .Net (Microsoft Initiative – Registrierung von Services), Biztalk, e-speak, etc.