We introduce DAML-S. This talk is based on:

- The DAML Services Coalition. [DAML-S 0.6: Technical Overview and WalkThrough]
- Unknown tag=Massimo Paolucci and Katia Sycara. [Autonomous Semantic Web Services.]

## 1 Introduction

- WSDL is a simple standard for describing webservices. It provides functionality similar to an API.
- If we hope to have agent compose just-in-time services from individual components, we will need more semantically-rich descriptions of services.
- That is, an agent needs to have some understanding of what `getStockQuote (string symbol)` does.
- DAML-Services is a set of ontologies, written in DAML, which can be used to describe (at a higher/more detailed semantic level) what a service does.

## 2 Upper Ontology for Services

- Resources are available out in the net.
- The DAML-S ontology defines a Service as the central class for describing interfaces, part of the DAML-S Service Ontology.
- What does the service require and provide for the users? This is given by the ServiceProfile. An agent uses it to determine whether the service meets it’s needs.
- How is it used? Given by the Service Grounding. It tells how to access the service.
3 Service Profiles

- The **ServiceProfile ontology** supports the use of three types of information.
  2. A specification of the functionalities that are provided by the service.
  3. Attributes which provide additional information and requirements (e.g., quality guarantees, expected response, geographic constraints, etc.)

- The functionalities are specified by declaring the IOPEs:
  - The *Inputs* the service expects.
  - The *Output* information returned.
  - The *Preconditions* that have to be satisfied in order to use the service.
  - The expected *Effects* from running the service.

3.1 ServiceProfile Ontology

3.2 Profile Description Attributes

- **serviceName** is the name (ID).
- **intendedPurpose** tells what constitutes successful accomplishment of service execution.
- **textDescription** English description.
- **providedBy** who provides it.
- **requestedBy** who requests this service.

3.3 Functional Description Attributes

- These attributes describe the interface.
- **input** describes the input(s) the service can receive.
- **output**
- **precondition** describes what must be true in order to use the service.
- **effect** what will happen when the service runs.
3.4 Functional Attributes

- A collection of other attributes that the service might have which do not deal with the process that the service implements.
- geographicRadius
- degreeOfQuality
- serviceParameter
- communicationThru
- serviceType
- serviceCategory
- qualityGuarantees
- qualityRating

4 Service Model

- Services are viewed as *processes* which are defined using a Process Ontology.
- A process can have any number of inputs.
- It can have any number of outputs.
- It has a parameter that specifies the participants in the process.
- It can have any number of preconditions that must hold for the process to be invoked.
- It can have any number of effects.
- Outputs and effects can have conditions associated with them.

4.1 Process Ontology

- An *AtomicProcess* is directly invocable, has not sub-processes, and executed in a single step.
• A **SimpleProcess** is not invocable (not associated with a grounding). Its executed in a single step. Used as an element of abstraction.

• A **CompositeProcess** is decomposable into other process using control constructs. It is **composedOf** a **ControlConstruct** which, in turn, has a **components** property that indicates the ordering and conditional execution of the sub-processes.

### 4.2 Process.daml

```xml
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF
    xmlns:rdfs="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd = "http://www.w3.org/2000/10/XMLSchema#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:service = "http://www.daml.org/services/daml-s/2001/10/Service#"
    xmlns = "http://www.daml.org/services/daml-s/2001/10/Process#"
>
<daml:Ontology rdf:about=""
    <daml:versionInfo>
    </daml:versionInfo>
    <rdfs:comment>
        Upper-level DAML ontology for Processes.
        Part of the DAML-S effort; see http://www.daml.org/services/.
    </rdfs:comment>
    <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil#"/>
</daml:Ontology>

<--######################################################
  Preliminaries
  ######################################################-->

<daml:Class rdf:ID="Condition">
    <rdfs:comment>
        This is a "place-holder" for now, which awaits further work from the DAML/OIL community. An instance of Condition is a logical formula that evaluates to true or false. Eventually we expect this to be defined elsewhere, as part of a DAML+OIL extension allowing for logical expressions.
    </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="ProcessClass">
    <rdfs:comment>
        This is the power set of Process (defined below); that is, the set of all subsets of Process. This class is needed as the range of certain properties. It’s placed in "preliminaries" because of the expectation that a better way of declaring these ranges may be developed.
    </rdfs:comment>
</daml:Class>
```
A ServiceModel describes how a Service works, and a ProcessModel is a type of ServiceModel (the only type being defined for DAML-S). See also the introductory comments to Service.daml.

A ServiceModel describes how a Service works, and a ProcessModel is a type of ServiceModel (the only type being defined for DAML-S). See also the introductory comments to Service.daml.

It is important to understand that DAML-S conceptualizes a specific process as a "class", each of whose instances is a particular use/invocation of the process. Thus, each specific process (such as BuyBook in Congo.daml) is a "subclass", rather than an instance, of Process. This is why the range of hasProcess is specified as ProcessClass (which is the class of all subclasses of Process).

In the case of a composite process, indicates the top-level process class.

A ProcessControlModel is used to monitor and control execution of a Process. This is just a "stub" for now; will be developed further in future releases of DAML-S.

A ProcessModel can have at most one Process and one ProcessControlModel.
Processes can have a name, parameters, preconditions, and (conditional) effects. Input, (conditional) output, and participant are subproperties of parameter. Each input, output, parameter, precondition or effect is a property of process, left unrestricted at this level (it ranges over "Thing"). The basic attributes are similar to the PDDL definition of events with the addition of input, output and participant attributes.

Processes can also be at an instant (atTime) or during an interval (during).

A Process can have at most one name, but names need not be unique.
Precondition is a property of Process. The top level ontology doesn’t restrict them at all. A precondition can be any kind of daml object (Thing). Specific processes will specialize this property by restricting the range appropriately using subPropertyOf; such as knowledge Preconditions (agent knows credit card number) or world Precondition (baud rate > 56Kb)

<!-
Parameters are properties of processes. The top level ontology doesn’t restrict them at all. A parameter can be any kind of daml object (Thing). Specific processes will specialize this property by restricting the range appropriately using subPropertyOf; such as the credit card number sub-property of a buying event would be restricted to have a range of type integer.

<!-
Effect is a property of process. The top level ontology doesn’t restrict them at all. An effect has the range ConditionalEffect. Processes will have conditional effects. For example, if the book is in stock, it will be purchased after executing the buy-book process. In the trivial case, the condition is "true" and the effect unconditionally holds.

<!-
ConditionalEffect is a thing that consists of a condition, and an effect thing. It has two properties, the condition (ceCondition) of the conditional effect, and the effect (ceEffect) of the conditional effect. In the buy-book example ceCondition would be "inStock(book)" and the ceEffect would be "purchased(book)".

<!-
We provide three common subtypes of parameters in the top level ontology. The types are input, conditionalOutput, and participant.

An input is a type of parameter and is a property of a process. The top level ontology doesn’t restrict them at all. An input can be any kind of daml object (Thing). Specific processes will specialize this property by restricting the range appropriately; such as required inputs, derivable inputs, etc.

An output is a type of parameter and is a property of a process. The top level ontology doesn’t restrict them at all. An output can be any kind of daml object (Thing). Each output may (optionally) be associated with a Condition.

ConditionalOutput is a thing that consists of a condition, and an output thing.
4.3 Control Constructs

- A **Sequence** is a list of **Processes** to be done in order.
- A **Split** contains a bag of process components to be executed concurrently.
- **Unordered** specifies a bag of process components that can be executed in any order.
- **Split+Join** consists of concurrent execution of process components with barrier synchronization.
- A **Choice** has further properties **chosen** and **chooseFrom** which let you create customized subsets.
- The **If-Then-Else** class has properties **ifCondition**, **then**, and **else**, which implement the statement.
- **Iterate** does just that until the **whileCondition** or **untilCondition** are met.
- **Repeat-Until** does a similar job.

4.4 Process Control Ontology

- It's an ontology that represent methods for monitoring and controlling the progress of an executing process.
- It does not exist yet.

4.5 Time Ontology

- DAML-S also defines a simple Time Ontology.
- It has two classes: **Instants** and **Intervals**.
- It has three properties that go from Interval to Instant:
  - **start-of**
  - **end-of**
  - **inside**
5 Resources

- There is also a Resource Ontology.
- Processes generally require (consume) resources.
- Resources have an AllocationType property which can be used to tell if the resource is consumable (e.g., time) or reusable (e.g., paint).

6 Congo Example

- This example is from the walkthrough.
- Congo is a website that sells books.
- Their services are LocateBook, PutInCart, SignIn, CreateAcct, CreateProfile, LoadProfile, SpecifyDeliveryDetails, FinalizeBuy.

6.1 Describe the Program

- Congo offers the CongoBuy service which is composed of smaller programs.
- You should describe these programs first.
- These individual programs are defined as Process.

6.2 Process Input and Output

- The process ontology (DAML) shows the various types of processes we can have.
- The LocateBook service is atomic, so we say that its a subclass of an atomic process. We also limit its cardinality to 1:

```xml
<daml:Class rdf:ID="LocateBook">
  <rdfs:subClassOf rdf:resource="&process;#AtomicProcess"/>
  <daml:Restriction daml:cardinality="1">
    <daml:onProperty rdf:resource="#bookName"/>
  </daml:Restriction>
  <rdfs:subClassOf rdf:resource="&process;#RelativeProcess"/>
</daml:Class>
```

- Processes have properties associated with them. The input property tells us what the process takes as input. If we want LocateBook to take a book name as input we say:

```xml
<daml:Class rdf:ID="LocateBook">
  <rdfs:subPropertyOf rdf:resource="&process;#Input"/>
  <rdfs:domain rdf:resource="#LocateBook"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</daml:Class>
```

- The output property will depend on whether the call worked, and what it returns. Its what DAML-S calls a ConditionalOutput (class).
• If the book described by "bookName" is in Congo’s catalog, then the output of LocateBook is the description of the book and the prices. If the book is not in Congo’s catalogs, then the output is a message to this effect. The output of LocateBook is thus described as a conditional output. We illustrate the notion of a conditional output below in terms of the bookDescription.

```
<rdf:Property rdf:ID="bookDescription">
  <rdfs:subPropertyOf rdf:resource="&process;#conditionalOutput"/>
  <rdfs:domain rdf:resource="#LocateBook"/>
  <rdfs:range rdf:resource="InCatalogueBookDescription"/>
</rdf:Property>

<daml:Class rdf:ID="InCatalogueBookDescription">
  <rdfs:subClassOf rdf:resource="&process;#ConditionalOutput"/>
</daml:Class>

<rdf:Property rdf:ID="condInCatalogueBookDescription">
  <rdfs:subPropertyOf rdf:resource="&process;#coCondition"/>
  <rdfs:domain rdf:resource="#InCatalogueBookDescription"/>
  <rdfs:range rdf:resource="#InCatalogueBook"/>
</rdf:Property>

<rdf:Property rdf:ID="outInCatalogueBookDescription">
  <rdfs:subPropertyOf rdf:resource="&process;#coOutput"/>
  <rdfs:domain rdf:resource="#InCatalogueBookDescription"/>
  <rdfs:range rdf:resource="#TextBookDescription"/>
</rdf:Property>

<daml:Class rdf:ID="TextBookDescription">
  <rdfs:subClassOf rdf:resource="&daml;#Thing"/>
</daml:Class>
```

6.3 Process Preconditions and Effects

• In order to tie a bunch of processes together (compose) we also need to know their preconditions for execution and any side-effects they might have.

• So, DAML-S also has precondition and effect (yes, like AI planner operators. 1970’s AI research might yet find an application :-).)

• CongoBuy service has two preconditions: you must have an account and credit

```
<rdf:Property rdf:ID="congoBuyAcctExistsPrecondition">
  <rdfs:subPropertyOf rdf:resource="&process;#precondition"/>
  <rdfs:domain rdf:resource="#CongoBuy"/>
  <rdfs:range rdf:resource="#AcctExists"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyCreditExistsPrecondition">
  <rdfs:subPropertyOf rdf:resource="&process;#precondition"/>
  <rdfs:domain rdf:resource="#CongoBuy"/>
  <rdfs:range rdf:resource="#CreditExists"/>
</rdf:Property>
```

11
• **CongoBuy** has an effect of either shipping the order or failure:

```xml
<daml:Class rdf:ID="BuyEffectType">
  <daml:oneOf rdf:parseType="daml:collection">
    <BuyEffectType rdf:ID ="OrderShipped"/>
    <BuyEffectType rdf:ID ="Failure"/>
  </daml:oneOf>
</daml:Class>
```

```xml
<rdfs:Property rdf:ID="congoBuyEffect">
  <rdfs:subPropertyOf rdf:resource="&process;#effect"/>
  <rdfs:domain rdf:resource="#CongoBuy"/>
  <rdfs:range rdf:resource="#BuyEffectType"/>
</rdfs:Property>
```

### 6.4 Composite Processes

• A **CompositeProcess** is composed of a bunch of **ControlConstructs** which can be things like sequence, if-then-else, fork, while, etc.

• Build them in a top-down manner.

• **CongoBuy** has two steps: locating the book and then buying the book. While (for exposition) we assume that the locate book is an atomic process (without components), the buying of a book involves a sequence of subprocesses (other atomic or composite processes) that correspond to specifying a payment method, specifying the details of delivery (address, wrapping type, etc.) and finalizing the buy process.

```xml
<daml:Class rdf:ID="ExpandedCongoBuy">
  <rdfs:subClassOf rdf:resource="&process;#CompositeProcess"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="&process;#composedOf"/>
      <daml:toClass>
        <daml:Class>
          <daml:intersectionOf rdf:parseType="daml:collection">
            <daml:Class rdf:about="process:Sequence"/>
            <daml:Restriction>
              <daml:onProperty rdf:resource="&process;#components"/>
              <daml:toClass>
                <daml:Class>
                  <daml:listOfInstancesOf rdf:parseType="daml:collection">
                    <daml:Class rdf:about="#LocateBook"/>
                    <daml:Class rdf:about="#CongoBuyBook"/>
                  </daml:listOfInstancesOf>
                </daml:Class>
              </daml:toClass>
            </daml:Restriction>
            </daml:intersectionOf>
          </daml:Class>
        </daml:toClass>
      </daml:Restriction>
    </daml:Class>
  </rdfs:subClassOf>
</daml:Class>
```
6.5 Creating a Simplified View

- Optional.
- Define a simple CongoBuy service which expands to the ExpandedCongoBuy we just saw.
A B2C bookbuying example of DAML-S (DARPA Agent Markup Language for Services; see http://www.daml.org/services/) usage, illustrating a simple use of the process model. This is a sketch; not a complete example.

The service described here is a book buying (or selling, depending on your perspective) service from www.congo.com. We present two forms of the service. First, there's a one-step form, with the service treated as ATOMIC; i.e., no interactions between buying and selling agents are required, apart from invocation of the...
service and receipt of its outputs by the buyer. Given certain inputs and parameters, the service provides certain outputs and has specific effects. Second, a full-fledged version of the service is specified, showing its composition from its component services. The atomic service is CongoBuy, which has specified inputs, outputs, preconditions and effects (IOPEs). The full-fledged version of the service, ExpandedCongoBuy, includes an arrangement of subprocesses LocateBook, PutInCart, SignIn, CreateAcct, CreateProfile, LoadProfile, SpecifyDeliveryDetails, FinalizeBuy each with its own IOPE spec.

Created by Srini Narayanan (srini@ai.sri.com).
Modifications by Mark Burstein, David Martin, Sheila McIlraith, Srini Narayanan, Terri Payne.

NOTES ON LANGUAGE USE:
NOTE: This DAML code employs 2 proposed properties:
  daml:sameValuesAs
daml:listOfInstancesOf
which are not currently part of DAML+OIL.

Input, output, precondition, and effect properties of composite processes can, in principle, be automatically generated by tools. Since such tools don’t yet exist, they have been manually generated for this example.

<!--
<!-###############################################################################
DATA TYPES
###############################################################################->

<daml:Class rdf:ID="CreditCardType"/>
<daml:oneOf rdf:parseType="daml:collection">
  <CreditCardType rdf:ID="MasterCard"/>
  <CreditCardType rdf:ID="VISA"/>
  <CreditCardType rdf:ID="AmericanExpress"/>
  <CreditCardType rdf:ID="DiscoverCard"/>
</daml:oneOf>
</daml:Class>

<daml:Class rdf:ID="PackagingType"/>
<daml:oneOf rdf:parseType="daml:collection">
  <PackagingType rdf:ID="Giftwrap"/>
  <PackagingType rdf:ID="Ordinary"/>
</daml:oneOf>
</daml:Class>

<daml:Class rdf:ID="DeliveryType">
  <daml:oneOf rdf:parseType="daml:collection">
    <DeliveryType rdf:ID="FedExOneDay"/>
    <DeliveryType rdf:ID="FedEx2-3day"/>
    <DeliveryType rdf:ID="UPS"/>
    <DeliveryType rdf:ID="OrdinaryMail"/>
  </daml:oneOf>
</daml:Class>

<daml:Class rdf:ID="ValidityType">
  <daml:oneOf rdf:parseType="daml:collection">
    <ValidityType rdf:ID="Valid"/>
    <ValidityType rdf:ID="Expired"/>
    <ValidityType rdf:ID="InvalidCCNumber"/>
    <ValidityType rdf:ID="InvalidCCTYPE"/>
    <ValidityType rdf:ID="AuthorizationRefused"/>
  </daml:oneOf>
</daml:Class>

<daml:Class rdf:ID="BuyEffectType">
  <daml:oneOf rdf:parseType="daml:collection">
    <BuyEffectType rdf:ID="OrderShipped"/>
    <BuyEffectType rdf:ID="Failure"/>
  </daml:oneOf>
</daml:Class>

<!-- stub most of these for now -->

<daml:Class rdf:ID="Book">
  <rdfs:subClassOf rdf:resource="#daml:#Thing"/>
</daml:Class>

<daml:Class rdf:ID="EReceipt">
  <rdfs:subClassOf rdf:resource="#daml:#Thing"/>
</daml:Class>

<daml:Class rdf:ID="ShippingOrder">
  <rdfs:subClassOf rdf:resource="#daml:#Thing"/>
</daml:Class>

<daml:Class rdf:ID="SignInData">
  <rdfs:subClassOf rdf:resource="#daml:#Thing"/>
</daml:Class>

<rdf:Property rdf:ID="acctName">
  <rdfs:domain rdf:resource="#SignInData"/>
  <rdfs:range rdf:resource="#xsd:#string"/>
</rdf:Property>
<rdf:Property rdf:ID="password">
    <rdfs:domain rdf:resource="#SignInData"/>
    <rdfs:range rdf:resource="&xsd;#string"/>
</rdf:Property>

<daml:Class rdf:ID="ProfileInfo">
    <rdfs:subClassOf rdf:resource="&daml;#Thing"/>
</daml:Class>

<daml:Class rdf:ID="AcctInfo">
    <daml:unionOf rdf:parseType="daml:collection">
        <daml:Class rdf:about="#SignInData"/>
        <daml:Class rdf:about="#ProfileInfo"/>
    </daml:unionOf>
</daml:Class>

<daml:Class rdf:ID="ENotify">
    <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Class rdf:about="daml:List"/>
        <daml:Restriction>
            <daml:onProperty rdf:resource="&daml;#item"/>
            <daml:toClass rdf:resource="#Book"/>
        </daml:Restriction>
    </daml:intersectionOf>
</daml:Class>

<!-- Shopping cart
Holding the books is the Cart Class defined below
various processes like AddToCart, RemoveFromCart, itemInCart?, etc can be defined using the Cart Class as the structure manipulated.
So the output/effect of these processes would manipulate the cart to add, delete items. (These manipulations are not yet specified here.) -->

<daml:Class rdf:ID="Cart">
    <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Class rdf:about="daml:List"/>
        <daml:Restriction>
            <daml:onProperty rdf:resource="&daml;#item"/>
            <daml:toClass rdf:resource="#Book"/>
        </daml:Restriction>
    </daml:intersectionOf>
</daml:Class>

<!--THE BLACKBOX PROCESS CongoBuy
------------------------------------------->

<daml:Class rdf:ID="CongoBuy">
    <rdfs:subClassOf rdf:resource="&process;#SimpleProcess"/>
</daml:Class>
<!- Inputs -->

<rdf:Property rdf:ID="congoBuyBookName">
    <rdfs:subPropertyOf rdf:resource="&process;#input"/>
    <rdfs:domain rdf:resource="#CongoBuy"/>
    <rdfs:range rdf:resource="&xsd;#string"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuySignInInfo">
    <rdfs:subPropertyOf rdf:resource="&process;#input"/>
    <rdfs:domain rdf:resource="#CongoBuy"/>
    <rdfs:range rdf:resource="#SignInData"/>
</rdf:Property>

<!- 2 Preconditions: AccountExists and CreditExists. The range of the preconditions is a condition object with a conditionValue property. TBD: How to indicate that the conditionValue (output) property of CreditExists holds the operative value. -->

<daml:Class rdf:ID="CreditExists">
    <rdfs:subClassOf rdf:resource="&process;#Condition"/>
</daml:Class>

<daml:Class rdf:ID="AcctExists">
    <rdfs:subClassOf rdf:resource="&process;#Condition"/>
</daml:Class>

<rdf:Property rdf:ID="congoBuyAcctExistsPrecondition">
    <rdfs:subPropertyOf rdf:resource="&process;#precondition"/>
    <rdfs:domain rdf:resource="#CongoBuy"/>
    <rdfs:range rdf:resource="#AcctExists"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyCreditExistsPrecondition">
    <rdfs:subPropertyOf rdf:resource="&process;#precondition"/>
    <rdfs:domain rdf:resource="#CongoBuy"/>
    <rdfs:range rdf:resource="#CreditExists"/>
</rdf:Property>

<!- TWO OUTPUTS, E-RECEIPT AND SHIPPING-ORDER -->

<rdf:Property rdf:ID="congoBuyReceipt">
    <rdfs:subPropertyOf rdf:resource="&process;#output"/>
    <rdfs:domain rdf:resource="#CongoBuy"/>
    <rdfs:range rdf:resource="#EReceipt"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyShippingOrder">
    <rdfs:subPropertyOf rdf:resource="&process;#output"/>
    <rdfs:domain rdf:resource="#CongoBuy"/>
</rdf:Property>
<rdfs:range rdf:resource="#ShippingOrder"/>
</rdf:Property>

<!-- Other parameters -->

<rdf:Property rdf:ID="congoBuyCreditCardValidity">
  <rdfs:subPropertyOf rdf:resource="&process;#parameter"/>
  <rdfs:domain rdf:resource="#CongoBuy"/>
  <rdfs:range rdf:resource="#ValidityType"/>
</rdf:Property>

<!-- THE EFFECT IS EITHER ORDESRISHED OR FAILURE. Conditional Effects would require an If-Then-Else range. TBD... -->

<rdf:Property rdf:ID="congoBuyEffect">
  <rdfs:subPropertyOf rdf:resource="&process;#effect"/>
  <rdfs:domain rdf:resource="#CongoBuy"/>
  <rdfs:range rdf:resource="#BuyEffectType"/>
</rdf:Property>

<!--###################################################
THE COMPOSITE PROCESS: ExpandedCongoBuy
(given in top-down order)
###################################################-->

<!-- Now for the full-fledged book-buying process, ExpandedCongoBuy. This is more complicated since it has to specify the process schema including alternatives, conditional executions, etc. 

We build the class recursively in a top-down manner. The basic idea is that each CompositeProcess is composedOf a ControlConstruct, which may be a Sequence, Alternative, If-then-else, etc. Each such ControlConstruct, in turn, has a "components" property (a list or bag), which specifies the classes of the subcomponents of the ControlConstruct. These classes may themselves be processes or control constructs. Finally we bottom out when the components of a composite process are atomic processes.

In the Congo example we assume the following basic structure for the composition. The main steps are to locate a book and to then buy it. While (for exposition) we assume that the locate book is an atomic process (without components), the buying of a book involves a sequence of subprocesses that correspond to specifying a payment method, specifying the details of delivery (address, wrapping type, etc.) and finalizing the buy process. These are the component subprocesses of the sequence corresponding to the expanded buying process. The detailed buying sequence involves a sequence of component sub-processes corresponding to putting the book into a cart, followed by a signing-in choice, followed by a method of payment selection. The
signing-in choice process is itself a composite process which offers the alternatives of signing in as a new user (creating an account) or using a stored profile instead. Each of these processes is itself composite, bottoming out in a sequence of atomic processes corresponding to signing in or creating a new account/profile.

<!-- Expand and Collapse relations for the CongoBuy Process -->

<daml:Class rdf:about="#CongoBuy">
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="&process;#expand"/>
         <daml:toClass rdf:resource="#ExpandedCongoBuy"/>
      </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#ExpandedCongoBuy">
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="&process;#collapse"/>
         <daml:toClass rdf:resource="#CongoBuy"/>
      </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<!--
The top level process is a sequence whose components are an atomic process, LocateBook and a composite process CongoBuyBook
-->
<!-- The following inputs and outputs of ExpandedCongoBuy are derived from the corresponding inputs and outputs of its atomic subprocesses (as indicated by their sameValuesAs properties), and will normally be computed automatically by DAML-S tools. For example, ExpandedCongoBuy uses an input, congoBuyBookName, that’s the same as the bookName input to atomic process LocateBook. -->

<rdfs:Property rdf:ID="expCongoBuyBookName">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
  <daml:sameValuesAs rdf:resource="#bookName"/>
</rdfs:Property>
<rdf:Property rdf:ID="expCongoBuySignInInfo">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="#SignInData"/>
  <daml:sameValuesAs rdf:resource="#signInInfo"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyCreateAcctInfo">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="#AcctInfo"/>
  <daml:sameValuesAs rdf:resource="#createAcctInfo"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyCreditCardNumber">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="&xsd;#decimal"/>
  <daml:sameValuesAs rdf:resource="#creditCardNumber"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyCreditCardType">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="#CreditCardType"/>
  <daml:sameValuesAs rdf:resource="#creditCardType"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyCreditCardExpirationDate">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="&time;#Instant"/>
  <daml:sameValuesAs rdf:resource="#creditCardExpirationDate"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyDeliveryAddress">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
  <daml:sameValuesAs rdf:resource="#deliveryAddress"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyPackagingSelection">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="#PackagingType"/>
  <daml:sameValuesAs rdf:resource="#packagingSelection"/>
</rdf:Property>

<rdf:Property rdf:ID="expCongoBuyDeliveryTypeSelection">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#ExpandedCongoBuy"/>
  <rdfs:range rdf:resource="#DeliveryType"/>
  <daml:sameValuesAs rdf:resource="#deliveryTypeSelection"/>
</rdf:Property>
<!-- CongoBuyBook is a sequence whose components are a composite process, BuySequence, followed by two atomic processes SpecifyDeliveryDetails, and then FinalizeBuy -->

<daml:Class rdf:ID="CongoBuyBook">
    <rdfs:subClassOf rdf:resource="&process;#CompositeProcess"/>
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="&process;#composedOf"/>
            <daml:toClass>
                <daml:Class>
                    <daml:intersectionOf rdf:parseType="daml:collection">
                        <daml:Class rdf:about="process:Sequence"/>
                        <daml:Restriction>
                            <daml:onProperty rdf:resource="&process;#components"/>
                            <daml:toClass>
                                <daml:Class>
                                    <daml:listOfInstancesOf rdf:parseType="daml:collection">
                                        <daml:Class rdf:about="#BuySequence"/>
                                        <daml:Class rdf:about="#SpecifyDeliveryDetails"/>
                                        <daml:Class rdf:about="#FinalizeBuy"/>
                                    </daml:listOfInstancesOf>
                                </daml:Class>
                            </daml:toClass>
                            <daml:Restriction>
                                <daml:intersectionOf>
                                    <daml:Class>
                                        <daml:toClass>
                                        </daml:Class>
                                    </daml:intersectionOf>
                                </daml:Restriction>
                            </daml:Restriction>
                        </daml:Class>
                    </daml:intersectionOf>
                </daml:Class>
            </daml:toClass>
            <daml:Restriction>
                <daml:intersectionOf>
                    <daml:Class>
                        <daml:toClass>
                        </daml:Class>
                    </daml:intersectionOf>
                </daml:Restriction>
            </daml:Restriction>
        </daml:Restriction>
    </daml:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#congoBuyBookBookName"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#congoBuyBookCreditCardNumber"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#congoBuyBookCreditCardType"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
<!-- The following inputs and outputs of CongoBuyBook are derived from the corresponding inputs and outputs of its atomic subprocesses (as indicated by their sameValuesAs properties), and will normally be computed automatically by DAML-S tools. -->

<rdfs:Property rdf:ID="congoBuyBookBookName">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#CongoBuyBook"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
  <daml:sameValuesAs rdf:resource="#bookName"/>
</rdfs:Property>

<rdfs:Property rdf:ID="congoBuyBookCreateAcctInfo">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#CongoBuyBook"/>
  <rdfs:range rdf:resource="#AcctInfo"/>
  <daml:sameValuesAs rdf:resource="#createAcctInfo"/>
</rdfs:Property>

<rdfs:Property rdf:ID="congoBuyBookCreateAcctOutput">
  <rdfs:subPropertyOf rdf:resource="&process;#output"/>
  <rdfs:domain rdf:resource="#CongoBuyBook"/>
  <rdfs:range rdf:resource="#CreateAcctOutputType"/>
  <daml:sameValuesAs rdf:resource="#createAcctOutput"/>
</rdfs:Property>

<rdfs:Property rdf:ID="congoBuyBookSignInInfo">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#CongoBuyBook"/>
  <rdfs:range rdf:resource="#SignInData"/>
  <daml:sameValuesAs rdf:resource="#signInInfo"/>
</rdfs:Property>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyBookCreditCardNumber">
<rdfs:comment>NOTE: INTEGER IS NOT SUPPORTED YET IN DAML-L</rdfs:comment>
<rdfs:subPropertyOf rdf:resource="#input"/>
<rdfs:domain rdf:resource="#CongoBuyBook"/>
<rdfs:range rdf:resource="#creditCardNumber"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyBookCreditCardType">
<rdfs:subPropertyOf rdf:resource="#input"/>
<rdfs:domain rdf:resource="#CongoBuyBook"/>
<rdfs:range rdf:resource="#CreditCardType"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyBookCreditCardExpirationDate">
<rdfs:subPropertyOf rdf:resource="#input"/>
<rdfs:domain rdf:resource="#CongoBuyBook"/>
<rdfs:range rdf:resource="#creditCardExpirationDate"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyBookDeliveryAddress">
<rdfs:subPropertyOf rdf:resource="#input"/>
<rdfs:domain rdf:resource="#CongoBuyBook"/>
<rdfs:range rdf:resource="#string"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyBookPackagingSelection">
<rdfs:subPropertyOf rdf:resource="#input"/>
<rdfs:domain rdf:resource="#CongoBuyBook"/>
<rdfs:range rdf:resource="#PackagingType"/>
</rdf:Property>

<rdf:Property rdf:ID="congoBuyBookDeliveryTypeSelection">
<rdfs:subPropertyOf rdf:resource="#input"/>
<rdfs:domain rdf:resource="#CongoBuyBook"/>
<rdfs:range rdf:resource="#DeliveryType"/>
</rdf:Property>

<!-- BuySequence is a sequence whose components are an atomic process PutInCart, followed by a composite process SignInAlternatives, followed by an atomic process SpecifyPaymentMethod -->

<daml:Class rdf:ID="BuySequence">
<rdfs:subClassOf rdf:resource="#CompositeProcess"/>
</daml:Class>
<daml:Restriction>
  <daml:onProperty rdf:resource="&process;#composedOf"/>
  <daml:toClass>
    <daml:Class>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Class rdf:about="process:Sequence"/>
      </daml:Class>
      <daml:Restriction>
        <daml:onProperty rdf:resource="&process;#components"/>
        <daml:toClass>
          <daml:Class>
            <daml:listOfInstancesOf rdf:parseType="daml:collection">
              <daml:Class rdf:about="#PutInCart"/>
              <daml:Class rdf:about="#SignInAlternatives"/>
              <daml:Class rdf:about="#SpecifyPaymentMethod"/>
            </daml:listOfInstancesOf>
          </daml:Class>
        </daml:toClass>
        <daml:Restriction>
          <daml:intersectionOf>
            <daml:Class>
              <daml:toClass>
                <daml:Restriction>
                  <daml:Restriction daml:cardinality="1">
                    <daml:onProperty rdf:resource="#buySequenceBookName"/>
                  </daml:Restriction>
                  <daml:Restriction>
                    <daml:Restriction daml:cardinality="1">
                      <daml:onProperty rdf:resource="#buySequenceCreditCardNumber"/>
                    </daml:Restriction>
                  </daml:Restriction>
                </daml:toClass>
                <daml:Restriction>
                  <daml:Restriction daml:cardinality="1">
                    <daml:onProperty rdf:resource="#buySequenceCreditCardType"/>
                  </daml:Restriction>
                </daml:toClass>
                <daml:Restriction>
                  <daml:Restriction daml:cardinality="1">
                    <daml:onProperty rdf:resource="#buySequenceCreditCardExpirationDate"/>
                  </daml:Restriction>
                </daml:toClass>
              </daml:Class>
            </daml:intersectionOf>
          </daml:Restriction>
        </daml:toClass>
        <daml:Restriction>
          <daml:intersectionOf>
            <daml:Class>
              <daml:toClass>
                <daml:Restriction>
                  <daml:onProperty rdf:resource="&process;#input"/>
                </daml:toClass>
              </daml:Class>
            </daml:intersectionOf>
          </daml:Restriction>
        </daml:toClass>
      </daml:Class>
    </daml:Restriction>
</daml:Restriction>
<!-- and some other stuff.....-->
</rdf:RDF>

6.7 Advertising the Services

- Build a profile. The profile talks about who, where, how good, and how to access the service.
- I think it’s meant more for human consumption, except for the input part which looks a lot
like an interface definition.

```xml
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
<!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
<!ENTITY process "http://www.daml.org/services/daml-s/2001/10/Process">
<!ENTITY service "http://www.daml.org/services/daml-s/2001/10/Service">
<!ENTITY profile "http://www.daml.org/services/daml-s/2001/10/Profile">
<!ENTITY country "http://www.daml.ri.cmu.edu/ont/Country.daml">
<!ENTITY concepts "http://www.daml.ri.cmu.edu/ont/DAML-S/concepts.daml">
<!ENTITY congo "http://www.daml.org/services/daml-s/2001/10/Congo.daml">
<!ENTITY time "http://www.ai.sri.com/daml/ontologies/sri-basic/1-0/Time.daml">
<!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema.xsd">
<!ENTITY DEFAULT "http://www.daml.org/services/daml-s/2001/10/CongoProfile.daml">
]
</!

This document uses entity types as a shorthand for URIs.
Download the source for a version with unexpanded entities.
-->

<rdf:RDF
  xmlns:rdf="&rdfs;#"
  xmlns:rdfs="&rdfs;#"
  xmlns:daml="&daml;#"
  xmlns:service="&service;#"
  xmlns:process="&process;#"
  xmlns:profile="&profile;#"
  xmlns:congo="&congo;#"
  xmlns:xsd="&xsd;#"
  xmlns="&DEFAULT;#">
  <daml:Ontology about=""/>
  <daml:versionInfo>
    $Id: CongoProfile.daml,v 1.5 2002/03/20 01:45:09 martin Exp $
  </daml:versionInfo>
  <rdfs:comment>
    DAML-S Coalition: CongoBuy Example for DAML-S release 0.6, 15th Sept 2001
    Profile description
  </rdfs:comment>
</rdf:RDF>
```
<!- ################################################################### ->
<!- # Instance Definition of BravoAir Reservation Agent Advertisement # -->
<!- ################################################################### ->

<service:ServiceProfile rdf:ID="Profile_Congo_BookBuying_Service">
  <service:isPresentedBy rdf:resource="congo:#Congo_BookBuying_Agent"/>

  <profile:serviceName>Congo_BookBuying_Agent</profile:serviceName>
  <profile:textDescription>
    This agentified service provides the opportunity to browse a book selling site and buy books there.
  </profile:textDescription>

  <profile:providedBy>
    <profile:ServiceProvider rdf:ID="CongoBuy">
      <profile:name>CongoBuy</profile:name>
      <profile:phone>412 268 8780</profile:phone>
      <profile:fax>412 268 5569</profile:fax>
      <profile:email>Bravo@Bravoair.com</profile:email>
      <profile:physicalAddress>
        somewhere 2, OnWeb, Montana 52321, USA
      </profile:physicalAddress>
    </profile:ServiceProvider>
  </profile:providedBy>

  <profile:geographicRadius rdf:resource="&country;#UnitedStates"/>
  <profile:qualityRating rdf:resource="&concepts;#qualityRatingGood"/>
  <profile:has_process rdf:resource="&congo;#CongoBuy"/>

  <input>
    <profile:ParameterDescription rdf:ID="BookTitle">
      <profile:parameterName>bookTitle</profile:parameterName>
      <profile:restrictedTo rdf:resource="&xsd;#string"/>
      <profile:refersTo rdf:resource="&congo;#bookName"/>
    </profile:ParameterDescription>
  </input>

  <input>
    <profile:ParameterDescription rdf:ID="SignInInfo">
      <profile:parameterName>signInInfo</profile:parameterName>
      <profile:restrictedTo rdf:resource="&congo;#SignInData"/>
      <profile:refersTo rdf:resource="&congo;#signInInfo"/>
    </profile:ParameterDescription>
  </input>
</service:ServiceProfile>
<input>
<profile:ParameterDescription rdf:ID="CreateAcctInfo">
<profile:parameterName> createAcctInfo </profile:parameterName>
<profile:restrictedTo rdf:resource="&congo;#CreateAcct"/>
<profile:refersTo rdf:resource="&congo;#createAcctInfo"/>
</profile:ParameterDescription>
</input>

<input>
<profile:ParameterDescription rdf:ID="CreditCardNumber">
<profile:parameterName> creditCardNumber </profile:parameterName>
<profile:restrictedTo rdf:resource="&xsd;#decimal"/>
<profile:refersTo rdf:resource="&congo;#creditCardNumber"/>
</profile:ParameterDescription>
</input>

<input>
<profile:ParameterDescription rdf:ID="CreditCardType">
<profile:parameterName> creditCardType </profile:parameterName>
<profile:restrictedTo rdf:resource="&congo;#CreditCardType"/>
<profile:refersTo rdf:resource="&congo;#creditCardType"/>
</profile:ParameterDescription>
</input>

<input>
<profile:ParameterDescription rdf:ID="CreditCardExpirationDate">
<profile:parameterName> creditCardExpirationDate </profile:parameterName>
<profile:restrictedTo rdf:resource="&time;#Time"/>
<profile:refersTo rdf:resource="&congo;#creditCardExpirationDate"/>
</profile:ParameterDescription>
</input>

<input>
<profile:ParameterDescription rdf:ID="DeliveryAddress">
<profile:parameterName> deliveryAddress </profile:parameterName>
<profile:restrictedTo rdf:resource="&xsd;#string"/>
<profile:refersTo rdf:resource="&congo;#deliveryAddress"/>
</profile:ParameterDescription>
</input>

<input>
<profile:ParameterDescription rdf:ID="PackagingSelection">
<profile:parameterName> packagingSelection </profile:parameterName>
<profile:restrictedTo rdf:resource="&congo;#PackagingType"/>
<profile:refersTo rdf:resource="&congo;#packagingSelection"/>
</profile:ParameterDescription>
</input>

<input>
<profile:ParameterDescription rdf:ID="DeliveryType">
<profile:parameterName> DeliveryType </profile:parameterName>
<profile:restrictedTo rdf:resource="&congo;#DeliveryType"/>
<profile:refersTo rdf:resource="&congo;#DeliveryType"/>
</profile:ParameterDescription>
</input>
<output>
<profile:ParameterDescription rdf:ID="EReceipt">
  <profile:paramName> EReceipt </profile:paramName>
  <profile:restrictedTo rdf:resource="&congo;#EReceipt"/>
  <profile:refersTo rdf:resource="&congo;#congoBuyReceipt"/>
</profile:ParameterDescription>
</output>

<output>
<profile:ParameterDescription rdf:ID="ShippingOrder">
  <profile:paramName> ShippingOrder </profile:paramName>
  <profile:restrictedTo rdf:resource="&congo;#ShippingOrder"/>
  <profile:refersTo rdf:resource="&congo;#congoBuyShippingOrder"/>
</profile:ParameterDescription>
</output>

<output>
<profile:ParameterDescription rdf:ID="AccountType">
  <profile:paramName> AccountType </profile:paramName>
  <profile:restrictedTo rdf:resource="&congo;#CreateAcctOutputType"/>
  <profile:refersTo rdf:resource="&congo;#createAcctOutput"/>
</profile:ParameterDescription>
</output>

<!-- Preconditions -->

<precondition>
<profile:ConditionDescription rdf:ID="AcctExists">
  <profile:conditionName> AcctExists </profile:conditionName>
  <profile:statement rdf:resource="&congo;#AcctExists"/>
  <profile:refersTo rdf:resource="&congo;#congoBuyAcctExistsPrecondition"/>
</profile:ConditionDescription>
</precondition>

<precondition>
<profile:ConditionDescription rdf:ID="CreditExists">
  <profile:conditionName> CreditExists </profile:conditionName>
  <profile:statement rdf:resource="&congo;#CreditExists"/>
  <profile:refersTo rdf:resource="&congo;#congoBuyCreditExistsPrecondition"/>
</profile:ConditionDescription>
</precondition>

<effect>
<profile:ConditionDescription rdf:ID="BuyEffectType">
  <profile:conditionName> BuyEffectType </profile:conditionName>
  <profile:statement rdf:resource="&congo;#BuyEffectType"/>
  <profile:refersTo rdf:resource="&congo;#congoBuyEffect"/>
</profile:ConditionDescription>
</effect>

</service:ServiceProfile>
7 BravoAir-Service.daml

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef>  
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">  
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">  
<!ENTITY daml "http://www.daml.org/2001/03/daml+oil.daml">  
<!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">  
<!ENTITY service "http://www.daml.org/services/daml-s/2001/10/Service">  
<!ENTITY ba_profile "http://www.daml.org/services/daml-s/2001/10/BravoAir-Profile.daml">  
<!ENTITY ba_process "http://www.daml.org/services/daml-s/2001/10/BravoAir-Process.daml">  
<!ENTITY DEFAULT "http://www.daml.org/services/daml-s/2001/10/BravoAir-Service.daml">  
>

<daml:Ontology>
  <daml:versionInfo>
  $Id: BravoAir-Service.daml,v 1.1 2002/03/20 01:44:52 martin Exp$
  </daml:versionInfo>
  <rdfs:comment> This ontology represents the DAML-S service description for the 
  BravoAir web service example. </rdfs:comment>
  <daml:imports rdf:resource="#&daml;"/>
  <daml:imports rdf:resource="#&service;"/>
  <daml:imports rdf:resource="#&ba_profile;"/>
  <daml:imports rdf:resource="#&ba_process;"/>
</daml:Ontology>

<service:Service rdf:ID="BravoAirReservationAgent">  
  <!-- Reference to the BravoAir Profile -->  
  <service:presents rdf:resource="#&ba_profile;#Profile_BravoAirReservationAgent"/>
  <!-- Reference to the BravoAir Process Model -->  
  <service:describedBy rdf:resource="#&ba_process;#BravoAirProcess"/>
</service:Service>

</rdf:RDF>

7.1 BravoAir-Profile.daml

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef>
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
<!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
<!ENTITY service "http://www.daml.org/services/daml-s/2001/10/Service">
<!ENTITY profile "http://www.daml.ri.cmu.edu/ont/DAML-S/Profile">
<!ENTITY process "http://www.daml.org/services/daml-s/2001/10/Process">
<!ENTITY ba_service "http://www.daml.org/services/daml-s/2001/10/BravoAir-Service.daml">
<!ENTITY ba_process "http://www.daml.org/services/daml-s/2001/10/BravoAir-Process.daml">
<!ENTITY country "http://www.daml.ri.cmu.edu/ont/Country.daml">
<!ENTITY concepts "http://www.daml.ri.cmu.edu/ont/DAML-S/concepts.daml">
<!ENTITY DEFAULT "http://www.daml.org/services/daml-s/2001/10/BravoAir-Profile.daml">

<!-- This document uses entity types as a shorthand for URIs. Download the source for a version with unexpanded entities. -->

<rdf:RDF
  xmlns:rdf="&rdf;#"
  xmlns:rdfs="&rdfs;#"
  xmlns:daml="&daml;#"
  xmlns:service="&service;#"
  xmlns:process="&process;#"
  xmlns:profile="&profile;#"
  xmlns="&DEFAULT;#">

  <daml:Ontology about=""/>
  <daml:versionInfo>
    $Id: BravoAir-Profile.daml,v 1.3 2002/03/20 01:45:09 martin Exp $
  </daml:versionInfo>
  <rdfs:comment>DAML-S Coalition: BravoAir Example for DAML-S release 0.6, 15th Sept 2001
  Profile description</rdfs:comment>
  <daml:imports rdf:resource="&rdf;"/>
  <daml:imports rdf:resource="&rdfs;"/>
  <daml:imports rdf:resource="&daml;"/>
  <daml:imports rdf:resource="&service;"/>
  <daml:imports rdf:resource="&process;"/>
  <daml:imports rdf:resource="&profile;"/>
  <daml:imports rdf:resource="&country;"/>
  <daml:imports rdf:resource="&ba_service;"/>
  <daml:imports rdf:resource="&ba_process;"/>
  <daml:imports rdf:resource="&concepts;"/>
</daml:Ontology>

<!-- # Instance Definition of BravoAir Reservation Agent Advertisement # -->

<!-- # Instance Definition of BravoAir Reservation Agent Advertisement # -->
<service:ServiceProfile rdf:ID="Profile_BravoAir_ReservationAgent">
</service:ServiceProfile>

<service:isPresentedBy rdf:resource="&ba_service;#BravoAir_ReservationAgent/>

<profile:hasProcess rdf:resource="&ba_process;#BravoAir_Process/>

<profile:serviceName>BravoAir_ReservationAgent</profile:serviceName>
<profile:textDescription>
This agentified service provides flight reservations based on the specification of a flight request. This typically involves a departure airport, an arrival airport, a departure date, and if a return trip is required, a return date. If the desired flight is available, an itinerary and reservation number will be returned.
</profile:textDescription>

<profile:providedBy>
<profile:ServiceProvider rdf:ID="BravoAir">
<profile:name>BravoAir</profile:name>
<profile:phone>412 268 8780</profile:phone>
<profile:fax>412 268 5569</profile:fax>
<profile:email>Bravo@Bravoair.com</profile:email>
<profile:physicalAddress>
Airstrip 2,
Teetering Cliff Hights,
Florida 12321,
USA
</profile:physicalAddress>
<profile:webURL>
http://www.daml.org/services/daml-s/2001/05/BravoAir.html
</profile:webURL>
</profile:ServiceProvider>
</profile:providedBy>

<profile:geographicRadius rdf:resource="&country;#UnitedStates"/>
<profile:qualityRating rdf:resource="&concepts;#qualityRatingGood"/>
</profile:textDescription>

<profile:input>
<profile:ParameterDescription rdf:ID="DepartureAirport">
<profile:parameterName>DepartureAirport</profile:parameterName>
<profile:restrictedTo rdf:resource="&concepts;#Airport"/>
<profile:refersTo rdf:resource="&ba_process;#departureAirport_In"/>
</profile:ParameterDescription>
</profile:input>

<profile:input>
<profile:ParameterDescription rdf:ID="ArrivalAirport">
<profile:parameterName>ArrivalAirport</profile:parameterName>
<profile:restrictedTo rdf:resource="&concepts;#Airport"/>
<profile:refersTo rdf:resource="&ba_process;#arrivalAirport_In"/>
</profile:ParameterDescription>
</profile:input>
<profile:input>
  <profile:ParameterDescription rdf:ID="confirmation">
    <profile:parameterName> confirmation </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#Confirmation"/>
    <profile:refersTo rdf:resource="&ba_process;#Confirmation"/>
  </profile:ParameterDescription>
</profile:input>

<profile:input>
  <profile:ParameterDescription rdf:ID="ReservationNumber">
    <profile:parameterName> ReservationNumber </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#ReservationNumber"/>
    <profile:refersTo rdf:resource="&ba_process;#ReservationNumber"/>
  </profile:ParameterDescription>
</profile:input>

<profile:output>
  <profile:ParameterDescription rdf:ID="ReservationNumber">
    <profile:parameterName> FlightItinerary </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#FlightItineraryList"/>
    <profile:refersTo rdf:resource="&ba_process;#FlightItineraryList"/>
  </profile:ParameterDescription>
</profile:output>

<!-- The consequence of the reservation is that the traveler has a seat on the flight -->

<profile:effect>
  <profile:ParameterDescription rdf:ID="HaveFlight">
    <profile:parameterName> HaveFlight </profile:parameterName>
    <profile:restrictedTo rdf:resource="&concepts;#HaveFlightSeat"/>
    <profile:refersTo rdf:resource="&ba_process;#HaveFlightSeat"/>
  </profile:ParameterDescription>
</profile:effect>

</service:ServiceProfile>
</rdf:RDF>

7.2 BravoAir-Process.daml

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/daml-s/2001/10/Service">
  <!ENTITY process "http://www.daml.org/services/daml-s/2001/10/Process">
  <!ENTITY profile "http://www.daml.org/services/daml-s/2001/10/Profile">
  <!ENTITY ba_service "http://www.daml.org/services/daml-s/2001/10/BravoAir-Service.daml">
  <!ENTITY concepts "http://www.daml.ri.cmu.edu/ont/DAML-S/concepts.daml">
<ENTITY DEFAULT "http://www.daml.org/services/daml-s/2001/10/BravoAir-Process.daml">
</ENTITY>

<!--
This document uses entity types as a shorthand for URIs.
Download the source for a version with unexpanded entities.
-->

<rdf:RDF
   xmlns:rdf="&rdf;#
   xmlns:rdfs="&rdfs;#
   xmlns:daml="&daml;#
   xmlns:xsd="&xsd;#
   xmlns:service="&service;#
   xmlns:process="&process;#
   xmlns:profile="&profile;#
   xmlns="&DEFAULT;#">

<daml:Ontology about="">
   <daml:versionInfo>
$Id: BravoAir-Process.daml,v 1.3 2002/03/20 01:45:09 martin Exp $
   </daml:versionInfo>
   <daml:Ontology>
   
   <!-- Instance Definition of AlphaAir Reservation Agent Process Model -->

   <process:ProcessModel rdf:ID="BravoAir_ReservationAgent_ProcessModel"> 
   <service:topLevelProcess rdf:resource="#BravoAir_Process" />
   <service:isImplementedBy> 
   <service:Service rdf:resource="#ba_service;#BravoAir_ReservationAgent"/>
   </service:isImplementedBy>
   </process:ProcessModel>

   </daml:Ontology>
</daml:Ontology>

<!-- Instance Definition of the top level process -->

<rdfs:Class rdf:ID="BravoAir_Process">

36
<rdfs:subClassOf rdf:resource="&process;#Process" /
</rdfs:Class>

<!-- ########################################################################## -
<!-- Expand and Collapse relations for BravoAir_Process
   Based on the Congo.daml model
-->

<process:expand>
  <rdfs:Class rdf:about="#BravoAir_Process"></rdfs:Class>
  <rdfs:Class rdf:about="#ExpandedBravoAir_Process"></rdfs:Class>
</process:expand>

<!-- ########################################################################## -
<!-- ExpandBravoAir_Process (Composite)
   Top Level Description of the Process
-->

<rdfs:Class rdf:ID="ExpandBravoAir_Process">
  <rdfs:subClassOf rdf:resource="&process;#CompositeProcess" />
  <rdfs:subClassOf rdf:resource="&process;#Sequence" />
  <daml:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="&process;#components" />
      <daml:toClass>
        <daml:subClassOf>
          <daml:unionOf rdf:parseType="daml:collection">
            <rdfs:Class rdf:about="#GetDesiredFlightDetails"/>
            <rdfs:Class rdf:about="#SelectAvailableFlight"/>
            <rdfs:Class rdf:about="#BookFlight"/>
          </daml:unionOf>
        </daml:subClassOf>
      </daml:toClass>
    </daml:Restriction>
  </daml:subClassOf>
</rdfs:Class>

<!-- ########################################################################## -
<!-- BookFlight (Composite)
   Log into account and confirm reservation
-->

<rdfs:Class rdf:ID="BookFlight">
  <rdfs:subClassOf rdf:resource="#CompositeProcess" />
  <rdfs:subClassOf rdf:resource="&process;#Sequence" />
  <daml:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#process;#components" />
    </daml:toClass>
  </daml:subClassOf>
</rdfs:Class>
<daml:subClassOf>
  <daml:unionOf rdf:parseType="daml:collection">
    <rdfs:Class rdfs:about="#LogIn"/>
    <rdfs:Class rdfs:about="#ConfirmReservation"/>
  </daml:unionOf>
  <rdfs:subClassOf rdf:resource="#ConfirmReservation"/>
  <rdfs:Class rdfs:about="#ConfirmReservation"/>
  <rdfs:subClassOf rdf:resource="#LogIn"/>
</daml:subClassOf>
</daml:Restriction>
</rdfs:Class>
</daml:toClass>
</daml:subClassOf>
</daml:toClass>
</daml:subClassOf>

<!-- No specification of inputs or outputs are generated here -->

<!-- ################################################################-->
<!-- GetDesiredFlightDetails (ATOMIC) Get details such as airports, prefered time, roundtrip etc -->

<rdfs:Class rdf:ID="GetDesiredFlightDetails">
  <rdfs:subClassOf rdf:resource="&process;#AtomicProcess"/>
</rdfs:Class>

<!-- GetDesiredFlightDetails IOPEs -->

<rdf:Property rdf:ID="departureAirport_In">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="&#GetDesiredFlightDetails"/>
  <rdfs:range rdf:resource="&concepts;#Airport"/>
</rdf:Property>

<rdf:Property rdf:ID="arrivalAirport_In">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="&#GetDesiredFlightDetails"/>
  <rdfs:range rdf:resource="&concepts;#Airport"/>
</rdf:Property>

<rdf:Property rdf:ID="outboudDate_In">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="&#GetDesiredFlightDetails"/>
  <rdfs:range rdf:resource="&concepts;#FlightDate"/>
</rdf:Property>

<rdf:Property rdf:ID="inboundDate_In">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="&#GetDesiredFlightDetails"/>
  <rdfs:range rdf:resource="&concepts;#FlightDate"/>
</rdf:Property>

<rdf:Property rdf:ID="roundTrip_In">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="&#GetDesiredFlightDetails"/>
  <rdfs:range rdf:resource="&concepts;#RoundTrip"/>
</rdf:Property>
<!-- SelectAvailableFlight (ATOMIC) -->
    Get users preferred flight choice from available itineraries
    -->
</rdfs:Class>

<!-- SelectAvailableFlight IOPEs -->

</rdf:Property>

</rdf:Property>

</rdfs:Class>

<!-- LogIn (ATOMIC) -->
    Get user details
    -->
</rdfs:Class>

<!-- LogIn IOPE -->

</rdf:Property>

</rdf:Property>

</rdfs:Class>
8 Conclusion

- DAML-S is more complex than WSDL.
- DAMLS-S gives a lot more details about how a process is composed of other processes, what sequence they must execute, etc.
• It’s processes are akin to AI-planning operators.

• Just-in-time service composition will be much more likely if services are described using DAML-S. Unfortunately that will require extra effort on the programmer’s part (WSDL can be generated automatically).

• DAML-S is sits righ between web-services (RPCs over HTTP) and the Semantic Web vision.

Notes

1 http://www.daml.org/services/daml-s/2001/10/
2 http://jmvidal.cse.sc.edu/library/paoluccio03a.pdf
3 http://www.daml.org/services/daml-s/2001/10/Service.daml
4 http://www.daml.org/services/daml-s/2001/10/Profile.daml
5 http://www.daml.org/services/daml-s/2001/10/Process.daml
6 http://www.daml.org/services/daml-s/2001/10/Time.daml
7 http://www.daml.org/services/daml-s/2001/10/Resource.daml

This talk is available at http://jmvidal.cse.sc.edu/talks/damls

Copyright © 2004 Jose M Vidal. All rights reserved.