DIANE - An Integration Approach to Automated Service Discovery, Matchmaking and Composition

Summary by: Ben Rockstroh

July 20, 2009

Goal

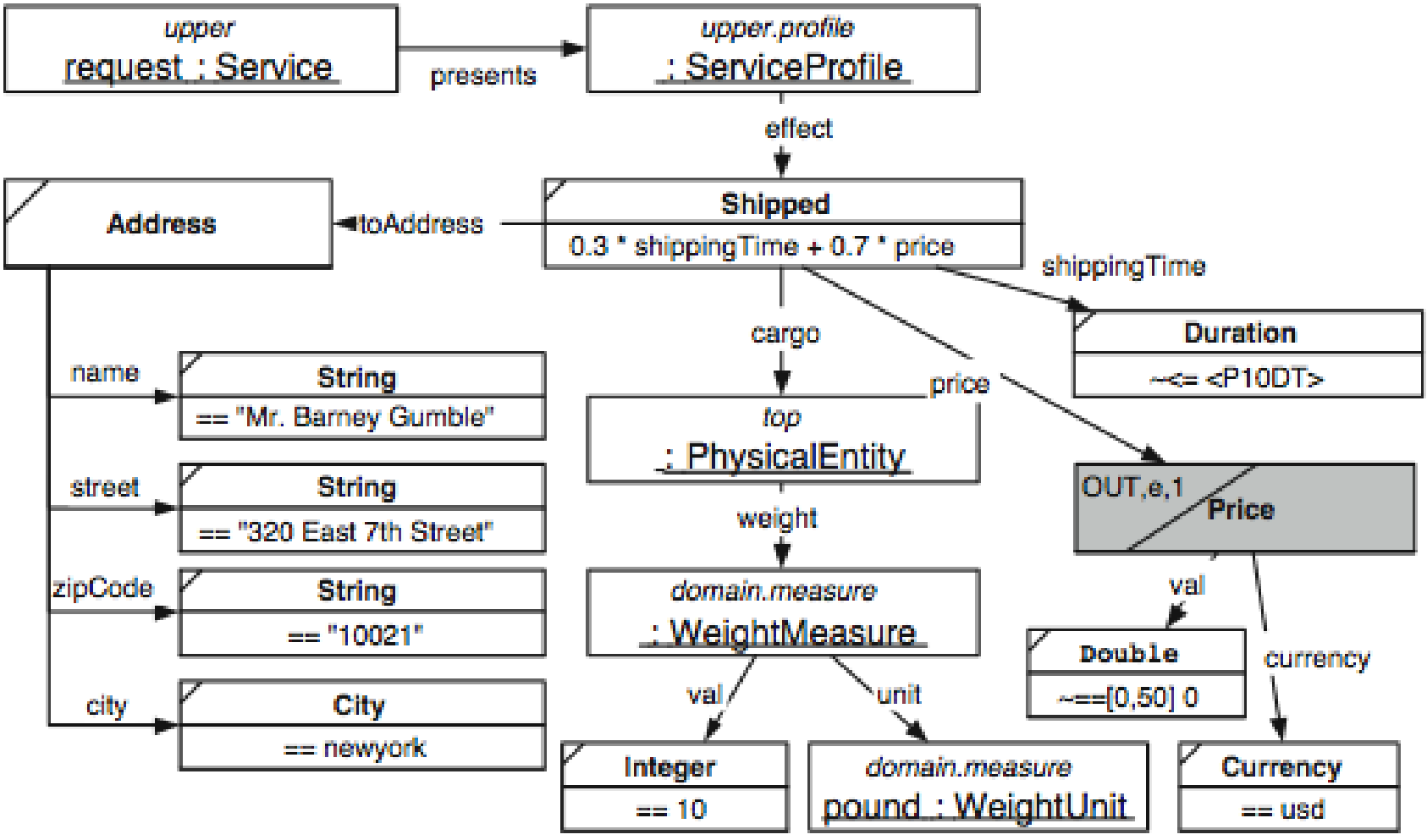
Develop an automated matcher that is able to compose services, provides fine-grained and precedes ranking among competing offers (single offers and automatically composed ones) and is able to automatically invoke the best offer.

DIANE Service Descriptions (DSD)

Based on a light weight custom ontology designed for service descriptions. Provides the ability to rank effects.

- Operational elements - Specify the world-altering affects that are desired from a service.
  e.g. Shipping a screw from France to Germany.
- Aggregating elements - Specify acceptable "extra" effects a service may have.
  e.g. I want a netbook, so I'll buy FIOS
- Selecting Elements - Used to specify elements that are to be selected from a service.
  e.g. I want book A from Amazon.
- Rating Elements - Used to rate desired affects.
  e.g. $(0.3 * \mathit{shippingTime} + 0.7 * \mathit{price})$ - price is more important than shipping.

Simplified Shipping Request



Single Effect Matching

Since DSD requests descriptions are trees stemming from roots (ontology concepts) single effect matching becomes a simple graph matching problem.

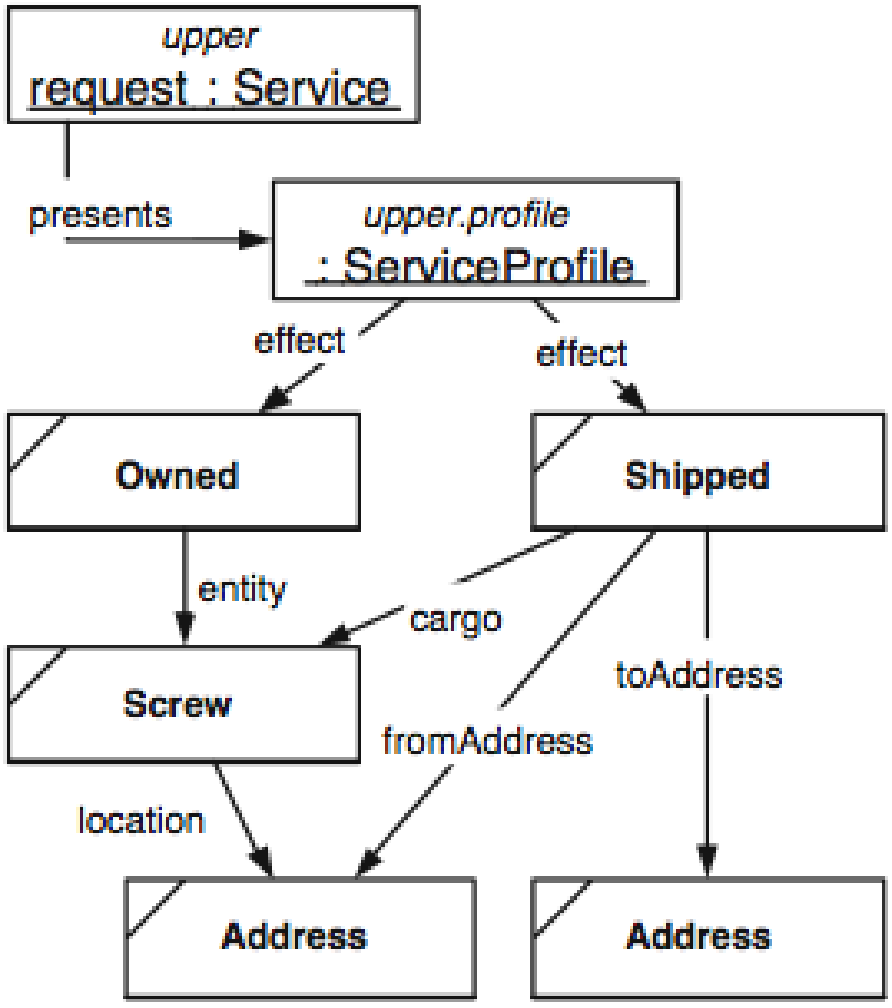
Problems Extending to Multiple Effects: Services containing multiple effects usually contain relations between the effects.

e.g. Ordering a screw and having it shipped to your location.

Service1 - Orders the screw at factory A.

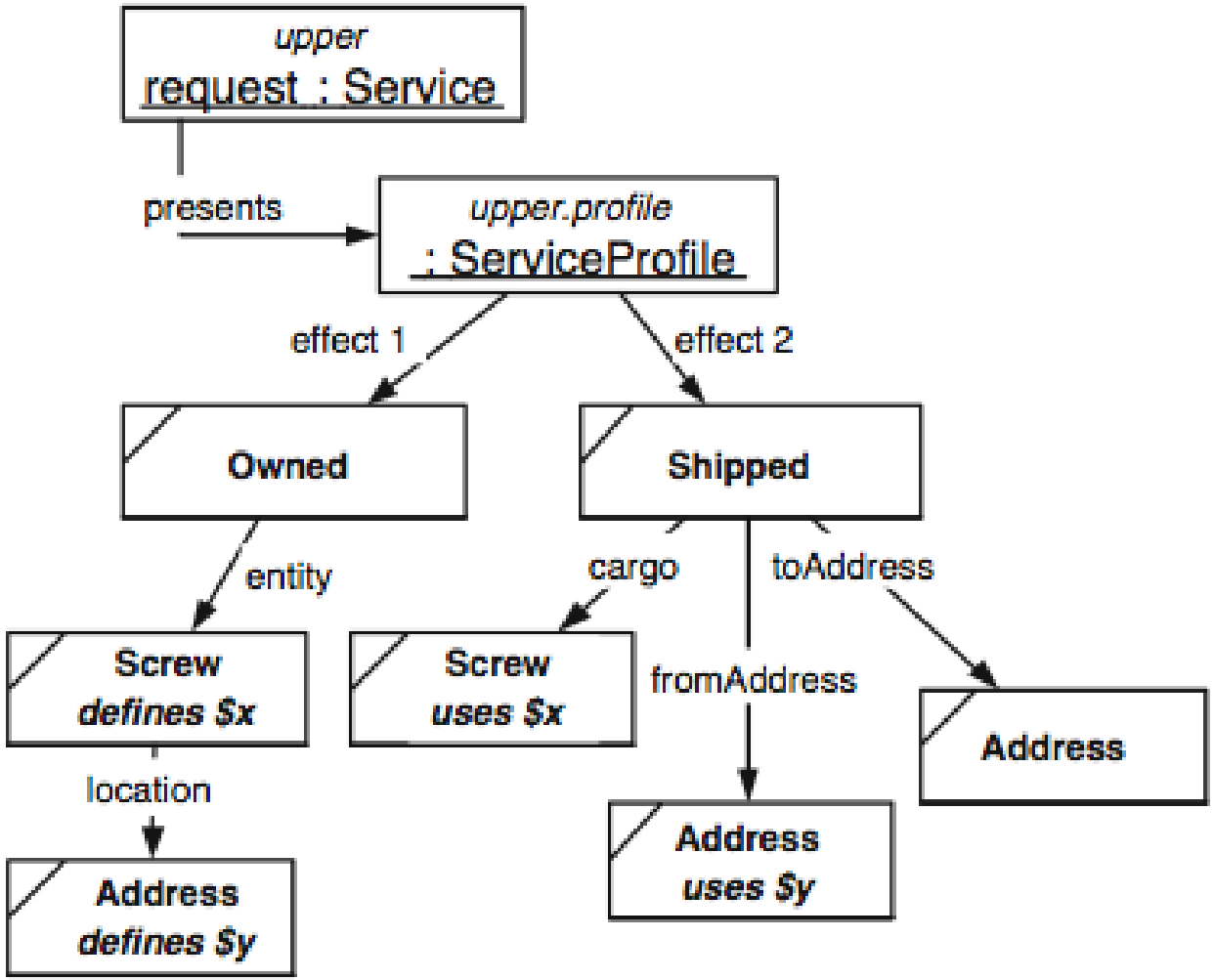
Service2 - Needs to ship the screw from factory A to your location.

Intuitive DSD Request for Multiple Effects



Value Propagation

The Idea: I care more about finding a service that does A then I do B. This allows the user to select what services the algorithm should look at first avoiding an exponential combination of



services.

Matching Multiple Effects

- Plug-in Matching
  - Check all offers for ability to fulfill a subset of the requested effects.
- Computing Compositions
  - Compute all possible compositions (coverages) so that every effect is covered by an offer exactly once.
  - Prepare for value matching - compute cut on the parameters involved.
  - Restrict concrete parameters to those involved in cut.
    i.e. Ignore parameters that are not needed.

Matching Multiple Effects Continued

- Final Result Computation
  - Optimally configure each offer with respect to user defined *rating elements*.
  - Compute matching values for coverage.
    e.g. *Service1* has a calculated value of 0.7.
    Service2 and *Service3* combine to provide the same service as *Service1* and have calculated values of 0.9 and 0.7 respectively. Thus the resulting service has a calculated value of 0.72 and would be more desirable.

Analysis

Component	Complexity
Prematch	$O(n)$
Plug-In Match	$O(\sigma_1 \cdot n \cdot m)$
Computing Compositions	$O((\sigma_1\sigma_2 \cdot n)^{const})$
Result Computation	$O(\sigma_3 \cdot (\sigma_1\sigma_2 n)^{const} \cdot m)$

Table 1: Overview: Complexity of the components

n	Overall number of offered services
m	Mean number of different configurations of an offer effect
σ_1	Selectivity of the pre-matcher, $\sigma_1 \ll 1$
σ_2	Selectivity of the plug-in-matcher, $\sigma_2 < 1$
σ_3	Selectivity of the composition-process, $\sigma_3 < 1$

Table 2: Meaning of the variables.

http://hnsp.inf-bb.uni-jena.de/DIANE/en/

Scalability

Concerned with 3 parameters

- Size of decryption (number of requested effects)
  Considered a constant since effects need to be related.
  Thought to be no more than 10.
- Number of available offers.
  Pre-matching performs linear search for appropriate offers.
- Number of ways to configure and offer.
  This can be in the hundreds of thousands though is reduced drastically by *Value Propagation*.