MW4SOC: part of Middleware ‘07
“Ontology based algorithm modeling: obtaining adaptation for SOA environment”

Simone Grassi, Trinity College Dublin
Computer Science Department
Distributed Systems Group
(other authors: Stephen Barrett, Francesco Sordillo)
Objectives

- Software development life cycle slow and expensive
- In SOA we need to model specific sections of the systems
- Adaptable Web Services: client driven adaptations
- Distribute adaptation to heterogeneous systems
State of the art

- Automatic service adaptation techniques act on the composition (ex: BPEL)
- Software synthesis techniques model big section of the systems (ex: UML and MDA)
- Standard and work in progress provides additional technology but don’t approach the problem
- We didn’t find an effort in creating adaptable web services
Ontology to model algorithms

- We can add as much semantic information as needed
- It enables the use of reasoning on Algorithms to evaluate: complexity, compare algorithms, adapt using aspects
- Enable to abstract/concretize algorithms
- Extend the Ontology to obtaining a Domain Specific Language approach
Case Study: tax scheme

CSP level 1 (Government)
BP/WS Registry
BP Engine

SP (Region 1)
BP/WS Registry
BP Engine

Accountancy Application Server
Web Portal
Web Portal with Tax Services
Accountancy Application

SP (Region n)
BP/WS Registry
BP Engine
An algorithm

The individuals of an OWL ontology creates a syntax-tree of an algorithm

```
TaxAmountLayersWithAgeReduction (idP,a,P,S,E,y,yp,py,nla)
```

```
for i=1,size(P)
+= v1
Pi
=
v1 0
Ei
if
> a
- Ei Si
if
< a Si
0 - a Si
```
abstracted algorithm: AO style

With a more abstract specification we can describe how to adapt an algorithm.
Contribution

- Show how to obtain client driven web service adaptation
- Some degree of freedom in automating algorithms adoption and adaptation for heterogeneous systems
Experiments & Tests

A software synthesis engine has been built

An Algorithm to compare syntax trees

Working on: method to abstract-concretize algorithm models
Limitations

- Every different system/framework need the common Local Adaptation Engine to be partially modified
- Only a reduced part of the systems can be modified
- Evaluate the skills needed by users
Future work

- Build better software synthesis engine
- Extend to other systems/frameworks the capability to generate code (ex: EJB, Ruby on Rails)
- Extend the used Ontology
- Use a real case scenario for new tests
Thanks for your attention!