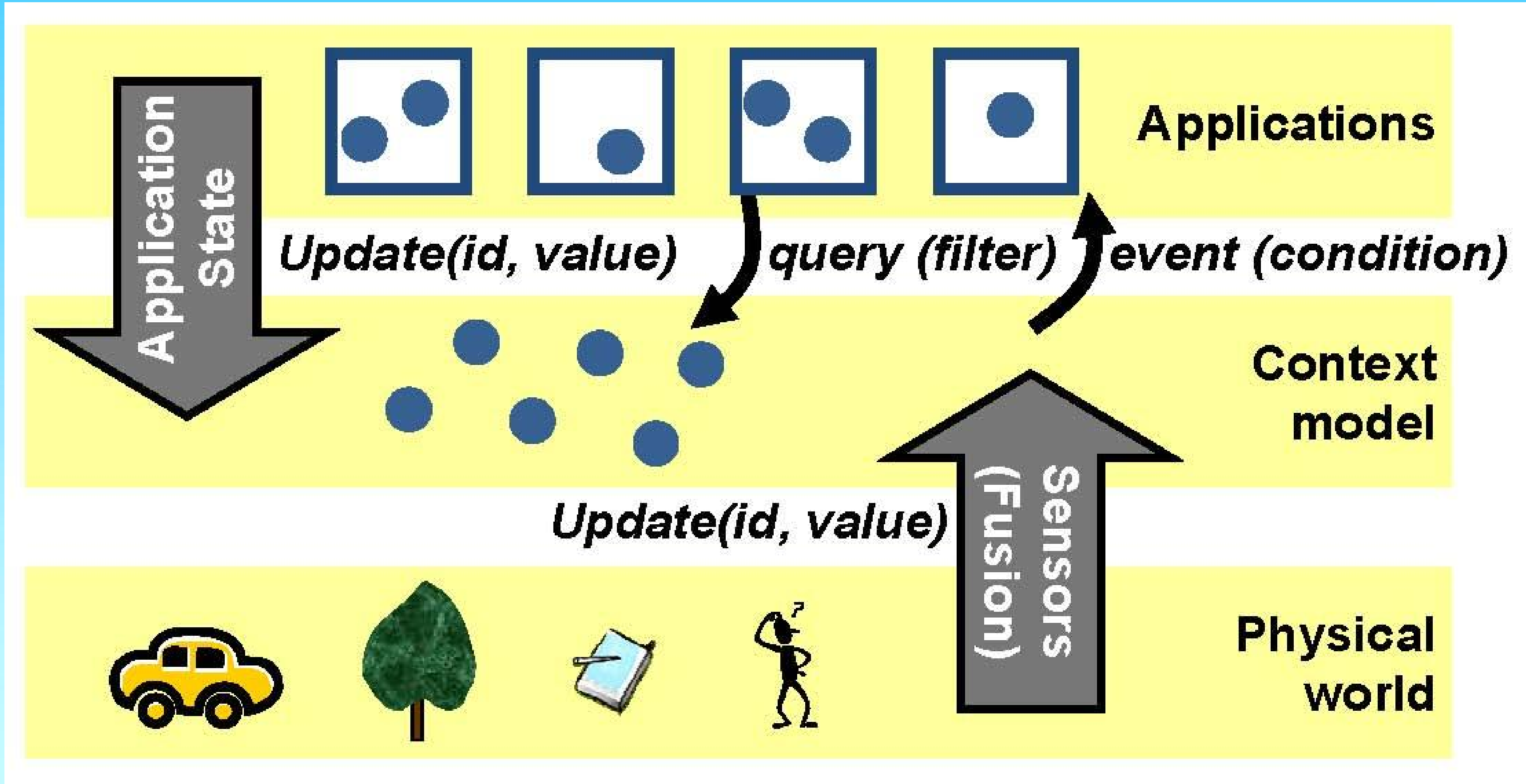
# Context Modeling

Some issues & techniques

# Context Models

- "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." (A. Dey)
- Some relevant questions:
  - How is context retrieved?
  - **How is context represented?**
  - How is context stored and managed?
  - How is context accessed by applications?
  - How is context shared between applications?

# A Generic Context Model




From Tutorial on Context Modeling, CoMoRea'10, by Bettini et al.

# Context Sources

- Context data could/should be grouped together by theirs sources:
  1. Sensors
  2. Applications
  3. Static environmental information
  4. (User, Application, Provider) Preferences

# Context Sources: 1 - Sensors

- Positioning systems (e.g., GPS, IPS- Indoor Positioning System)
- Temperature, acceleration, light, etc.
- Sensor Fusion
  - Light, more than n people, noise → room occupied
  - Dimmed light, more than n people, 1 talking → presentation

# Context Sources

2. Applications, e.g.:
   - state of services, specific information (e-notes, doorplates)
3. Static environmental information e.g.:
   - Global reference systems
     - Symbolic (world/USA/Berkeley …)
     - Geometric (N 50 11.8 E 009 10.2)
   - Topological information
     - Floor plans
     - Road network
     - City maps

# Context Sources: 4-Preferences

a. **User**
   - Affection
   - Restrictions – e.g., a visually impaired person → audio output
   - Cost, quality, …

b. Application (business logic) / Provider
   - Requirements on services, devices (displays, position information, …)

# Context Data: further characterization

- Static: context data which are created and never (or rarely) changed, e.g. road maps
- Dynamic: context data which are updated frequently, periodically, or on demand
- Sensed: context data obtained via sensors (reflecting state of the physical world)
- Application provided: context data provided by applications (mainly referring to virtual state w.r.t. the physical world)

# Various Context Classifications

<table>
<tr><th></th><th>Location</th><th>Conditions</th><th>Infrastructure (Computing Environment)</th><th>Information on User</th><th>Social</th><th>User Activity</th><th>Time</th><th>Device Characteristics</th></tr>
<tr><td>[Benerecetti et al. '01]</td><td colspan="2">Physical Environment</td><td></td><td colspan="2">Cultural Context</td><td></td><td></td><td></td></tr>
<tr><td>[Schmidt et al. '99]</td><td colspan="3">Physical Environment</td><td colspan="3">Human Factor</td><td>X</td><td></td></tr>
<tr><td>[Lieberman and Selker'00]</td><td>User Environment</td><td>Physical Environment</td><td>X</td><td>User Environment</td><td></td><td></td><td>X</td><td></td></tr>
<tr><td>[Hull et al. '97]</td><td></td><td>Physical Environment</td><td></td><td>X</td><td></td><td></td><td></td><td>X</td></tr>
<tr><td>[Chalmers and Sloman’99]</td><td>X</td><td></td><td>X</td><td></td><td>X</td><td>X</td><td></td><td>X</td></tr>
<tr><td>[Lucas’01]</td><td colspan="2">Physical Environment</td><td colspan="5">Information context</td><td>X</td></tr>
<tr><td>[Schilit et al’94]</td><td colspan="2">Physical Environment</td><td>X</td><td colspan="2">User environment</td><td></td><td></td><td></td></tr>
<tr><td>[Abowd and Dey’99]</td><td>X</td><td></td><td></td><td>Identity</td><td></td><td>X</td><td>X</td><td>Identity</td></tr>
<tr><td>[Chen & Kotz’00]</td><td colspan="8">Active/Passive</td></tr>
</table>

From CoMoRea’04, Panel 1

# Evolution of Context Modeling Goal

- Early models mainly addressed the modeling of context with respect to one application or an application class
- Generic context models are more interesting since many applications can benefit from these, of course they are more complicated
- The objective of most current research is to develop uniform context models, representation and query languages as well as reasoning algorithms that facilitate context sharing and interoperability of applications

A Context Modeling Survey, T. Strang, C. Linnhoff-Popien, 2004

# Some *Context-awareness* Requirements

- Support for distributed context data
  - Users, Network operators, Service providers, …
- Interoperable context representation
  - Shared knowledge and standard formats
- Support for context dynamics
  - Intra session changes + user and provider policies
- Efficiency
  - Multiple requests have to be served in real time
- Support for reasoning

See publication on Care, Agostini et al.

# Some **Context-Modeling** Requirements

- Distributed composition
- Partial Validation (structure and instances)
- Richness and quality of information
- Incompleteness and ambiguity
- *Level of formality (same interpretation of the data exchanged → semantics)*
- *Applicability to existing environments*

A Context Modeling Survey, T. Strang, C. Linnhoff-Popien, 2004

# Modeling Approaches: Simple Listing

1. *Object-Oriented Models (e.g., TEA and GUIDE projects): benefits -> inheritance, encapsulation and reusability*
2. Key-Value Models
3. Graphical Models (e.g., *UML*, Object-Role Modeling extension)
4. Markup Scheme Models (CC/PP & its extensions)
5. Logic Based Models
6. Ontology Based Models (e.g., CONON, CoBra, SOUPA, SOCAM)
7. *(Hybrid Approaches*)

A Context Modeling Survey, T. Strang, C. Linnhoff-Popien, 2004

# Key-Value Models

- The model of **key-value pairs** is the most simple data structure for modeling contextual information
- The key-value modeling approach is frequently used in distributed service frameworks. In such frameworks, the services itself are usually described with a list of simple attributes in a key-value manner, and the employed service discovery procedure (e.g. SLP, Jini,... see [39]) operates an exact matching algorithm on these attributes
- Key-value pairs are:
  - easy to manage
  - lack capabilities for sophisticated structuring (also useful for enabling efficient context retrieval algorithms)

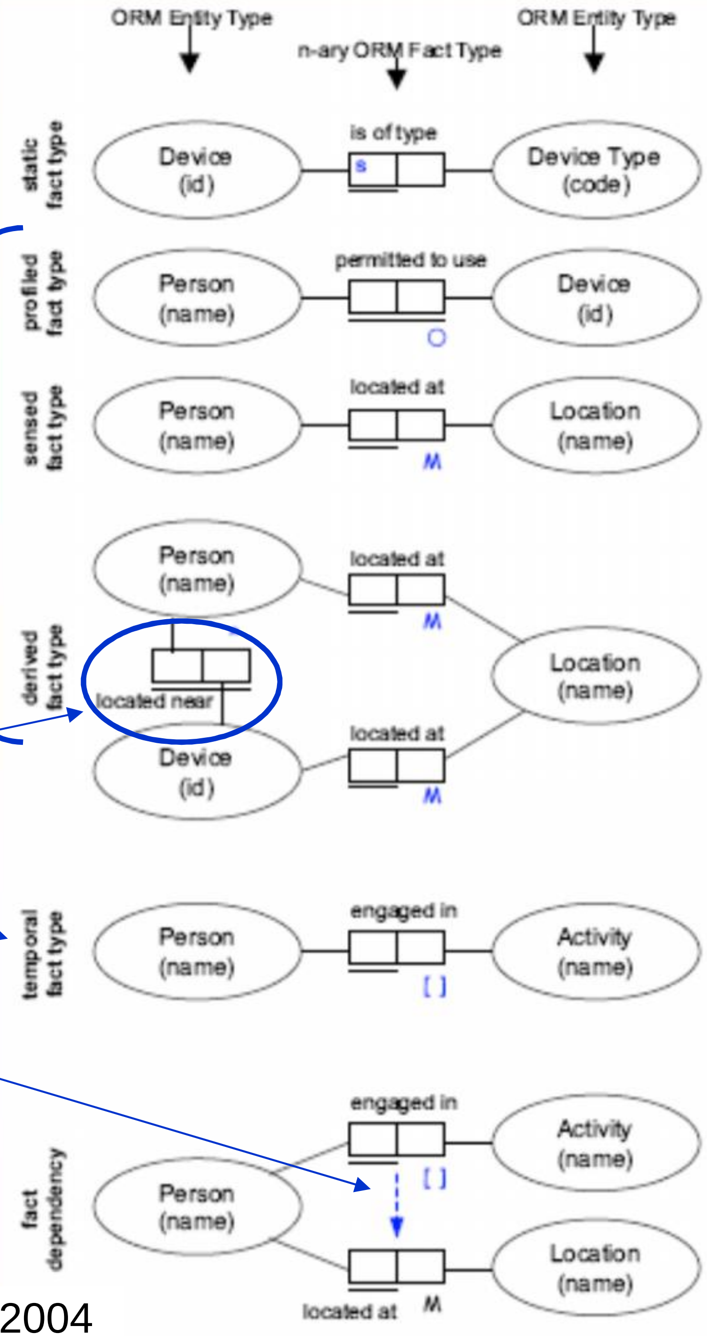
A Context Modeling Survey, T. Strang, C. Linnhoff-Popien, 2004

# Object-Role Model extension (Henricksen et al.)

- Basic concept is the *fact*
- Modeling means identifying fact types and the roles that entity types play in these

Henricksen extention:

1. Fact types can be categorised as: *static* or *dynamic*
2. Dynamic facts can be distinguished in: *profiled*, *sensed* or *derived* types
3. Quality indicator to cover time-aspects
4. Fact dependencies represents a special type of relationship between facts; a change in one fact leads automatically to a change in another fact




A Context Modeling Survey, T. Strang, C. Linnhoff-Popien, 2004

Figure 3: Contextual Extended ORM

# Markup Scheme Models

- Common to all markup scheme modeling approaches is a hierarchical data structure consisting of **markup tags** with attributes and content. The content of the markup tags is usually recursively defined by other markup tags
- Typical context modeling approaches are **profiles** usually based upon a serialization of a derivative of *Standard Generic Markup Language (SGML)*, the superclass of all markup languages
- Some of them are defined as extension to the **Composite Capabilities / Preferences Profile (CC/PP)** and User Agent Profile Vocabulary *(UAProf in HTML, UAProf in RDF/S)* standards, which have the expressiveness reachable by RDF/S and a XML serialization. *(A W3C Recommendation for the specification both of device capabilities and user preferences; now* see *Open Mobile Alliance*); an example of validated profile
- These approaches usually extend the basic CC/PP and UAProf vocabulary to try to cover the higher dynamics and complexity of contextual information compared to static profiles

**Extended profile**

# Logic Based Models

- A logic defines the conditions on which a concluding expression or fact may be derived (namely reasoning or inferencing) from a set of other expressions or facts. To describe these conditions in a set of rules a formal system is applied
- The context is defined as facts, expressions and rules. Contextual information is added to, updated in and deleted from a logic based system in terms of facts or inferred from the rules in the system respectively
- All logic based models have a high degree of formality

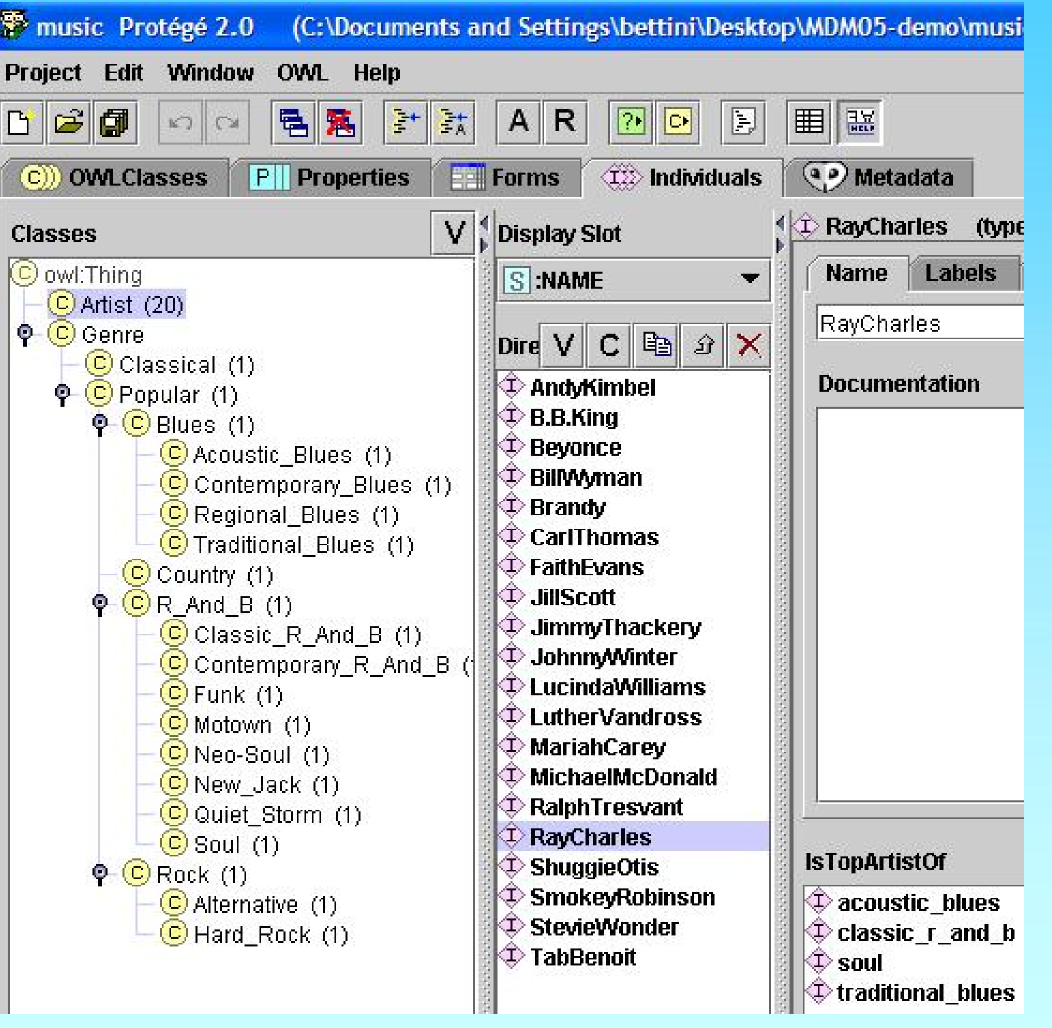
A Context Modeling Survey, T. Strang, C. Linnhoff-Popien, 2004

# Ontology Based Models

An ontology is the attempt to build a conceptual schema complete and rigorous for a specific domain; it is generally a hierarchical structure which contains the relevant entities, the relationships among them, the axioms and the constrains of the domain.

Ontologies for:

- Clear semantics of context concepts
- Knowledge sharing among involved entities
- Deriving further contextual information
- Consistency check of contextual data instances



# Context Modeling

## An Example: CARE

# CARE

- CARE is a middleware supporting different (generic) application domains
- CARE uses an hybrid approach to context modeling integrating CC/PP profiles, rules/policies, and ontologies

see CARE by Agostini, Bettini, Riboni et al. (http://webmind.dico.unimi.it/care/)

# Context-Awareness Requirements

A middleware for context-aware adaptation and personalization should:

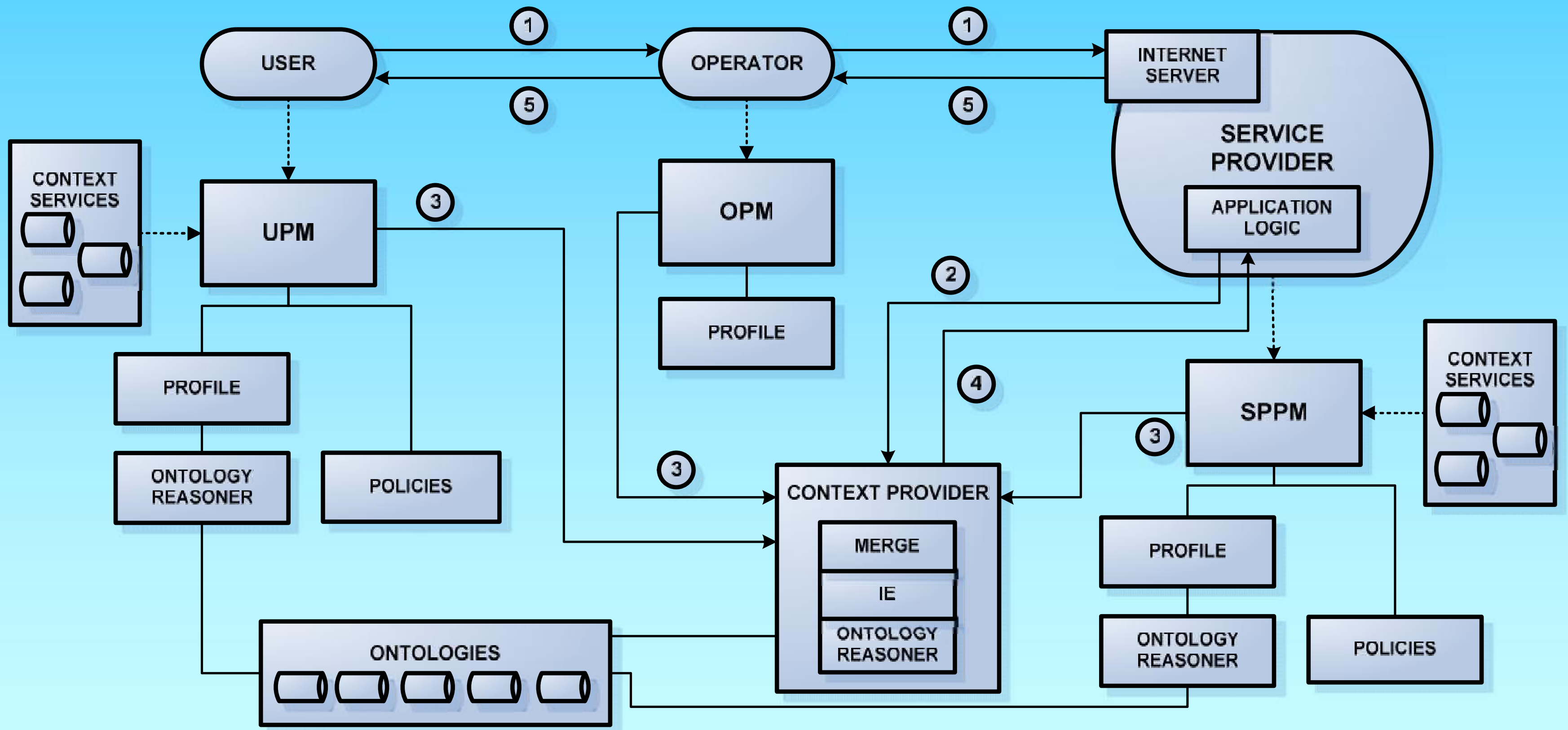
- Support **distributed** context data
- Allow **interoperability** of context representation
- Support context **dynamics**
- Handle "***complex***" context data (e.g., socio-cultural data)

... while maintaining **efficiency**

# Context Modeling: Bases

- Context data are "*any information that can be used to characterize the situation*" [Dey, '01] of a mobile user requesting a service
- Context data include spatio-temporal data, environmental conditions, technological infrastructure data, socio-cultural information
- Context Data are handled by different distributed entities (users, network operators, service providers)
- Context data of a single entity (profile) are partial and conflicting with other entities' profiles
- User and service provider can declare policies over profile data for adapting the service

# Architecture Overview



# Adaptation based on profiles and policies

- An extended notion of profile includes information about:
  - User personal data, device capabilities, network infrastructure, location, time...
- Profile information is distributed:
  - Users
  - Network operators
  - Service providers
- Both user and service provider can declare *policies* to adapt and personalize the service

# Profile representation

- Requirements:
  - Structure
  - Interoperability
  - Extensibility
  - Semantics

- Composite Capability/Preference Profiles (CC/PP)
  - A W3C effort for the specification both of device capabilities and user preferences
  - Based on RDF/XML
  - Profiles constructed as a two-level hierarchy:
    - Components
    - Attributes belonging to a single Component

# Merging profile data

- Different entities can provide different values for the same profile attribute

***User :***

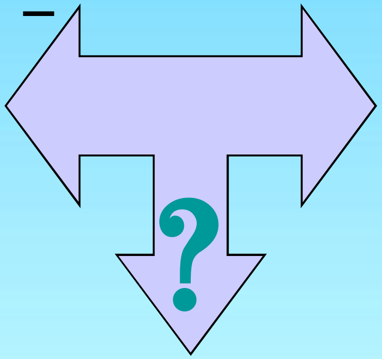
*AvailableBandwidth =*

***56 kbps***

***Service Provider :***

*AvailableBandwidth =*

***12 kbps***



Profile Resolution Directive

***AvailableBandwidth = <Service Provider, User>***

**AvailableBandwidth = 12 kbps**

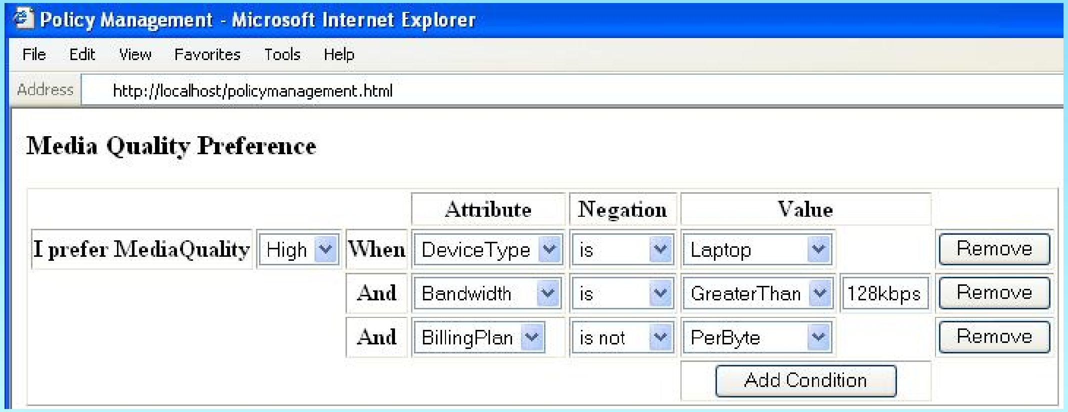
# Policy representation

- Requirements:
  - Expressiveness
  - Rule chaining
  - Efficiency
  - Conflicts handling
- A policy rule is composed by
  - A set of conditions on profile data that determine a new value for a profile attribute when satisfied

# Specification of policies

*“When I am using a laptop, the bandwidth is **higher than** 128kbps, and the billing plan **is not** per byte, I want to receive high-resolution multimedia content”*

User interface:




Policy language (general logic programs): [cycle detection]

*MediaQuality(High) ← DeviceType(Laptop), Bandwidth(X), >(X,128kbps), not BillingPlan(perByte)*.

# Policy conflicts

- **At most a single ground atom for each predicate must be present in the program model**
- **An example of two conflicting policies (due to the actual facts)**

rule or policy

*MediaQuality(High) ← DeviceType(Laptop), Bandwidth(X), X>128kbps, not BillingPlan(perByte)*

*MediaQuality(Low) ← BatteryLevel(Low)*

*DeviceType(Laptop) ←*

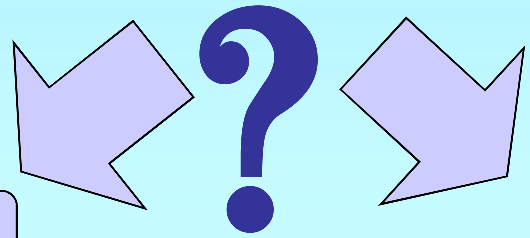
predicate

*BillingPlan(FlatRate) ←*

*Bandwidth(256kbps) ←*

*BatteryLevel(Low) ←*

fact



*MediaQuality('High')* | *MediaQuality('Low')*

# Conflict resolution strategies

Example:

MediaQuality = < UPM, SPPM, OPM > (profile resolution directive)

| *conflicts with* | **UPM RULE** | **SPPM RULE** |
|---|---|---|
| **UPM RULE** | *Explicit priority* | *UPM rule* |
| **SPPM RULE** | *UPM rule* | *Explicit priority* |
| **UPM FACT** | *UPM rule* | *UPM fact* |
| **SPPM FACT** | *UPM rule* | *SPPM rule* |
| **OPM* FACT** | *UPM rule* | *SPPM rule* |

*OPM non definisce regole

# Explicit Priority: Language extensions

- We extended general logic programs with *labels* and *priorities*
- The expression R1 > R2 states that rule *R*1 has higher priority than rule *R*2
- A second parameter is added to predicates
  - Predicate(Value, **Weight**)
  - Example:
    - Rule "*MediaQuality(Low,**2**) ← DeviceType(CellPhone, X)*" has weight 2

# Policy conflict resolution: example

***R1:*** *MediaQuality(High) ← DeviceType(Laptop), Bandwidth(X), X>128kbps, not BillingPlan(perByte)*

***R2:*** *MediaQuality(Low) ← BatteryLevel(Low)*

*DeviceType(Laptop) ←*

*BillingPlan(FlatRate) ←*

*Bandwidth(256kbps) ←*

*BatteryLevel(Low) ←*

***R2 > R1***

↓

*MediaQuality(High,* ***0****) ← DeviceType(Laptop, _ ), Bandwidth(X, _ ), X>128kbps, not BillingPlan(perByte, _ ) ,* ***not MediaQuality( _ , J), J>0***

*MediaQuality(Low,* ***1****) ← BatteryLevel(Low, _ ),* ***not MediaQuality( _ , J), J>1***

*DeviceType(Laptop,* ***0****) ←* ***not DeviceType( _ , J), J>0***

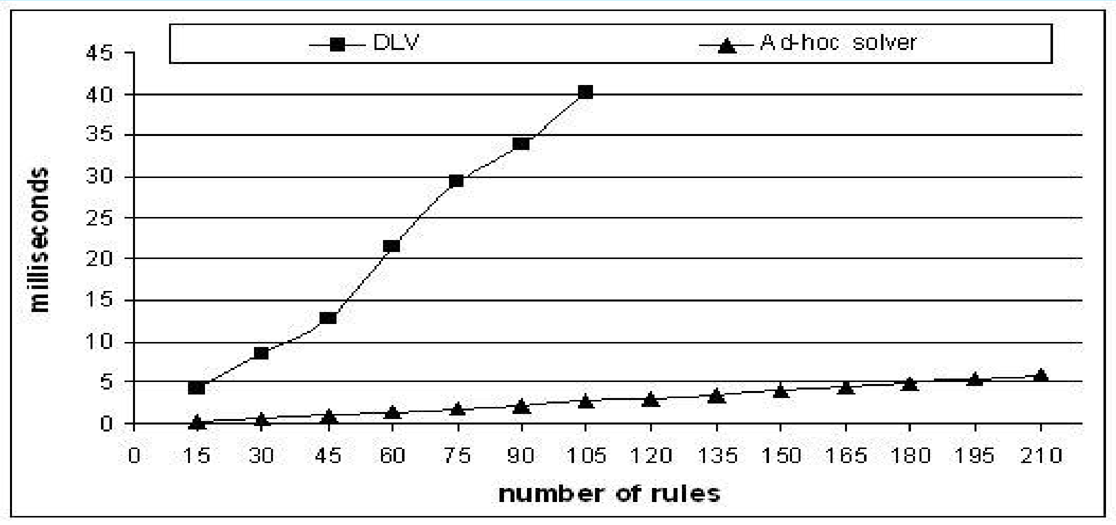
*BatteryLevel(Low,* ***0****) ←* ***not BatteryLevel( _ , J), J>0***

# Formal properties

- The *weight algorithm* ensures that
  - no pair of rules exist having the same head predicate symbol and the same weight
  - rules having the same head predicate but higher weight have higher priority, according to our conflict resolution strategy, over those with lower weights
- The transformations preserve *acyclicity*
- An evaluation algorithm is devised that is *linear* in the number of rules

# Performance evaluation

- Our policies can be evaluated by standard solvers (e.g. Mandarax, DLV)
- We have developed an ad-hoc evaluator in order to improve the performance, exploiting the characteristic features of our language



# Context Modeling

- We distinguish two categories of profile data which need different representations:
  - Shallow profile data (e.g., device capabilities, network characteristics)
  - Non-shallow profile data (e.g., socio-cultural information, user interests)

# Modeling Shallow Profile Data

- “Simple” data are modeled by attribute/value pairs adopting the CC/PP language
- CC/PP (Composite Capability/Preference Profiles) profiles:
  - RDF graphs composed by sets of *components* containing *attributes* with associated values
  - Having a strict two-level hierarchy (components-attributes)
  - Component/attributes names and their allowed values are defined in CC/PP vocabularies as RDF Schemas
  - Semantics expressed in natural language in the <rdfs:comment> resource

CC/PP proved unsuitable for modeling data belonging to complex domains

# An Excerpt of a CC/PP profile

```
<RDF xmlns:expro="http://webmind.dico.unimi.it/expro/extendedProfile-1.0.xml#"
        xmlns:uaprof="http://www.wapforum.org/UAPROF/ccppschema-19991014#">
<ccpp:component>
 <Description about="http://www.wapforum.org/UAPROF/ccppschema-19991014#">
  <type resource="http://www.wapforum.org/UAPROF/ccppschema-19991014#BrowserUA"/>
  <uaprof:AvailableStorageMemory>
    37.45
  </uaprof:AvailableStorageMemory>
 </Description>
</ccpp:component>
<ccpp:component>
 <Description about="http://webmind.dico.unimi.it/expro/extendedProfile-1.0.xml#">
   <type resource="http://webmind.dico.unimi.it/expro/extendedProfile-1.0.xml#Activity"/>
  <expro:currentAction>
    Chatting
   </expro:currentAction>
 </Description>
</ccpp:component>
</RDF>
```

**Shallow Attribute**

**Non-Shallow Attribute**

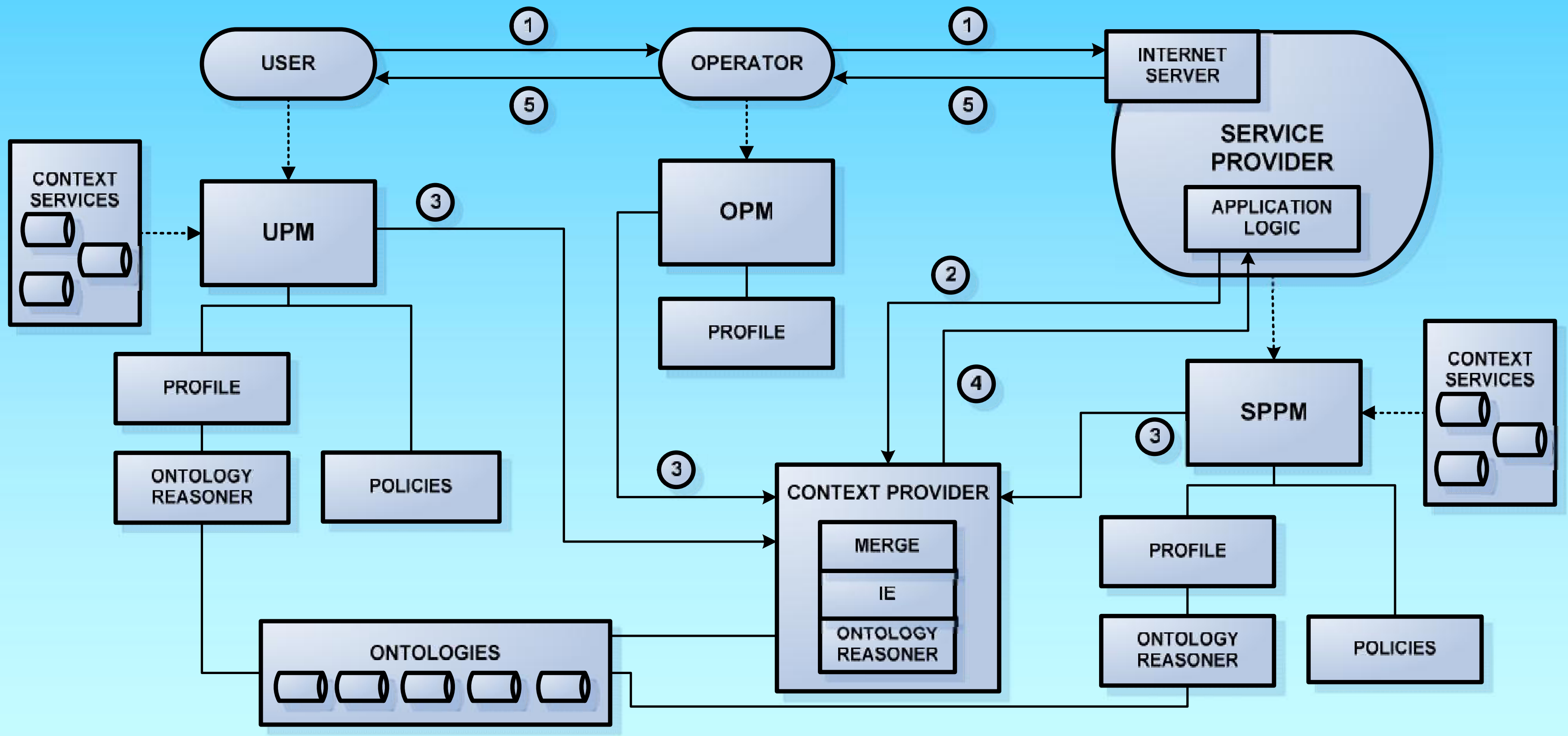
# Modeling Non-Shallow Profile Data

- “Complex” data are represented by means of both private and shared ontologies, in OWL-DL, to allow:
  - Knowledge sharing among involved entities (e.g., user interests)
  - Consistency check of contextual data instances
  - Reasoning to derive additional contextual data (e.g., specific activity of the user); in this case, private data of an entity can be exploited too
- To maintain interoperability ontology-based profile data are mapped into CC/PP attributes

# Ontological Reasoning Features

- Off-Line ontological reasoning:
  - Local to a single Profile Manager of an entity
  - Made before the user requests a service
  - Fired by local activation rules (e.g., change on a profile attribute)
  - Possibly using private data and/or private ontologies
- On-Demand ontological reasoning:
  - Local to the Context Provider module
  - Made after building the aggregated profile (i.e., at service request time)
  - Made when **crucial** ontology-based attributes have no value
  - The Service Provider, carefully, decides which attributes are crucial for a specific service
  - Using the whole aggregated profile
- Ontological and rule-based reasoning are made separately

# Architecture Overview



# Off-Line Reasoning: An Example

- Alina, a Unimi employee, is a user of an adaptive messaging service
- “When I’m involved in a work meeting I don’t wont to receive phone calls; please, set my state to busy”

**Alina’s policy**

AvailabilityState(Busy)←currentActivity(InternalWorkMeeting).

- The messaging service redirects calls to Alina’s answering machine when she is busy

# Off-Line Reasoning: An Example

Alina is in her office with a colleague, an entry in her calendar specifies the meeting... all phone calls are redirect to her answering machine

UnimiInternalMeeting $\equiv$ Activity $\sqcap$

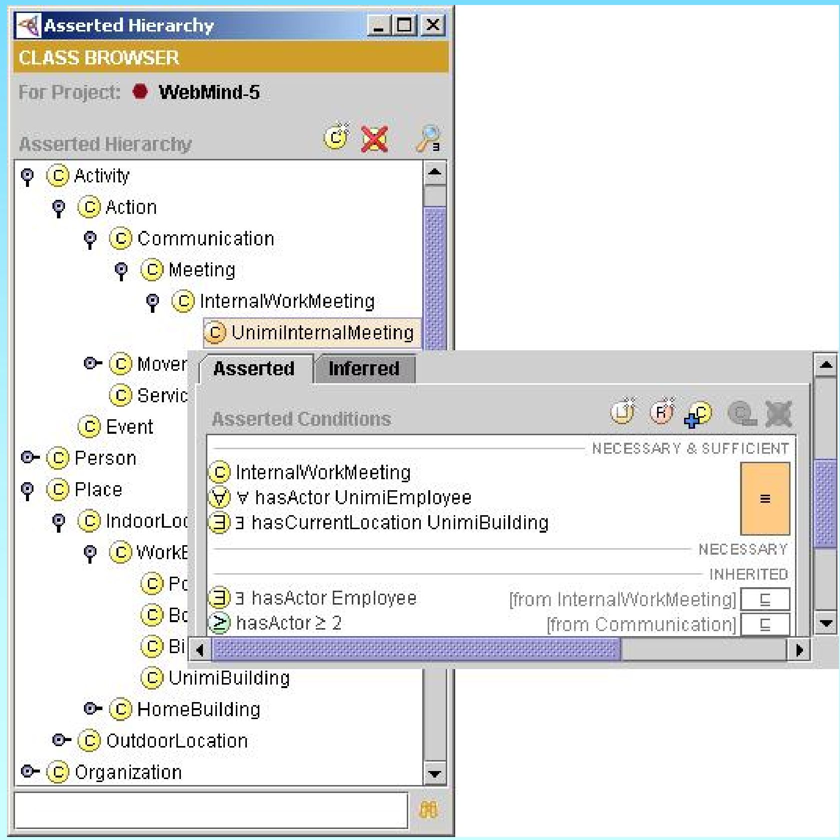
$\geq 2$ Actor $\sqcap$

$\forall$Actor.UnimiEmployee $\sqcap$

$\exists$ Location.UnimiLocation

UnimiEmployee $\equiv$ Employee $\sqcap$

$\exists$ Employer.{unimi}



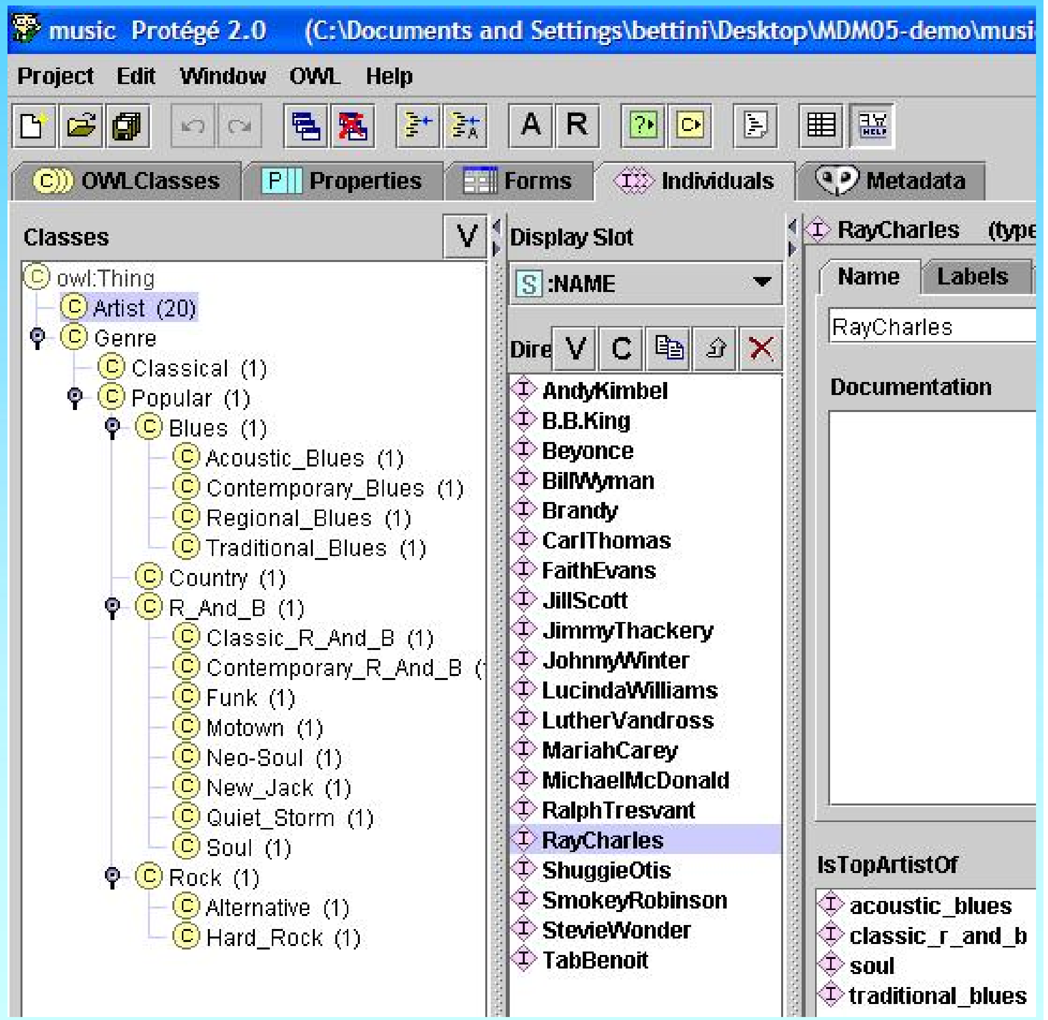
# On-demand Reasoning: An Example

- John is a user of a location-based recommendation service, selecting and ordering POIs and events according to user's interest and location

- John submits a query, but his integrated profile lacks his preferred music genres

# On-demand Reasoning: An Example

- However, the profile contains his preferred artists
- The Context Provider infers that John likes R&B music and updates the profile



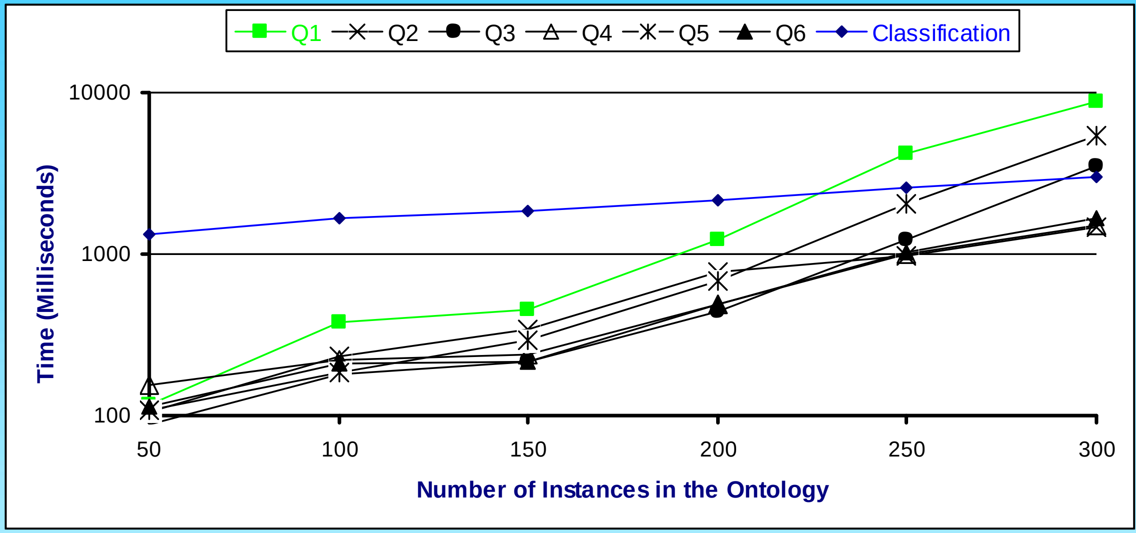
# On-demand Reasoning: An Example

- The service, exploiting the profile, is now able to select and order the music items appropriately

# A Prototype Using CARE: POIsmart

- A context-aware web service delivering POIsmarts
- POIsmarts:
  - Convergence of bookmarks and navigational Points of Interests
- Adaptation based on:
  - User location
  - User interests
  - Device capabilities
  - Device status
  - Available bandwidth
  - …

# Some Experimental Evaluation




- Tests made on our OWL-DL ontology modeling (part of) socio-cultural context (≅ 150 classes & relations) using Racer on a 2-processor Xeon, 2.4 GHz, 1.5 GB RAM
- Evaluation of policies, on-demand only, is not included; it is linear in the number of rules

# Some Conclusions

- Off-line reasoning on dedicated servers seems a practicable solution
- On-demand reasoning introduces both an appreciable delay on service provisioning and scalability problems
- OWL-DL shown some expressiveness limitation