Modeling Agent Based Services

Part II: Modeling Agent Services for Open Environments

Steven Willmott

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

http://www.lsi.upc.es/~webia/KEMLG

AGENTCITIES

Plan and Objectives

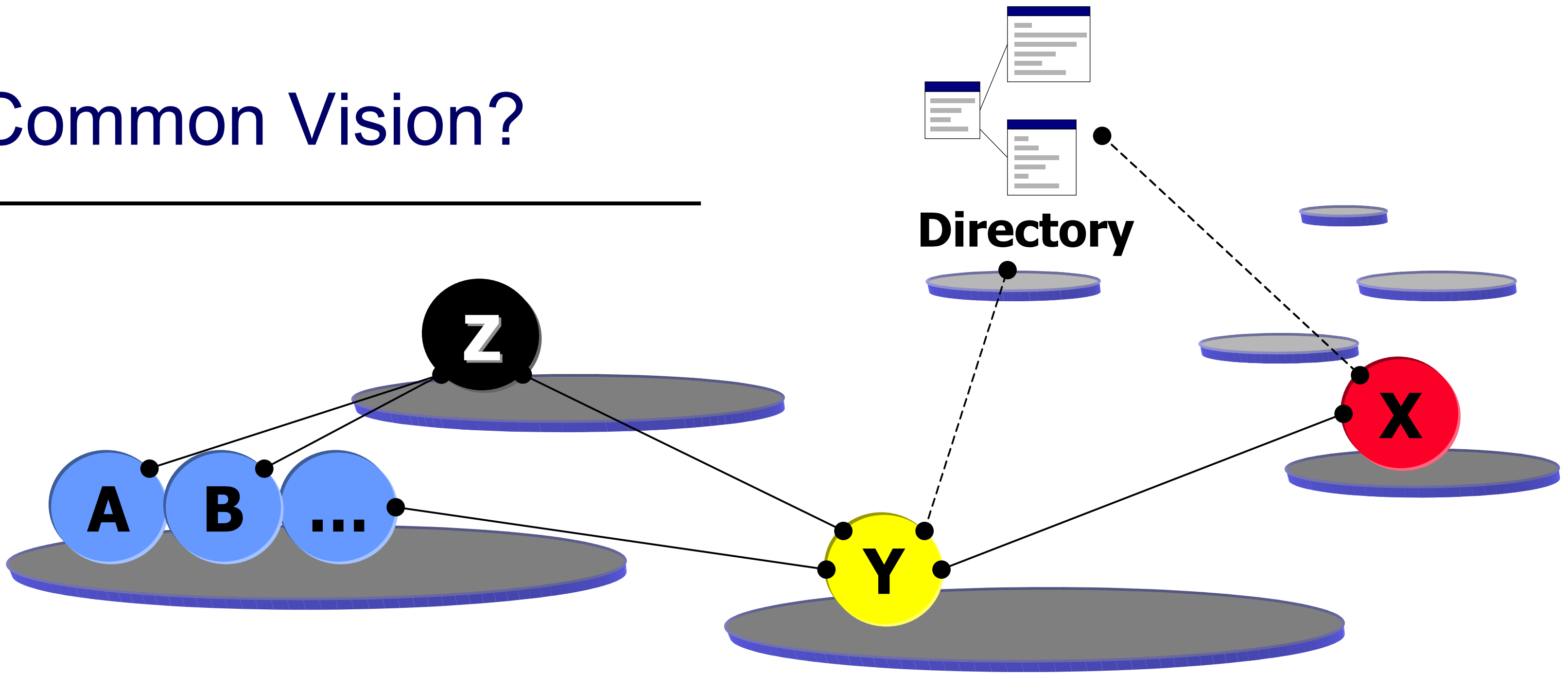
- Background
  - Services in Open Environments
  - Industry and Research trends
  - Agentcities
- Modeling an Agent Service
- Challenge Areas
  - Autonomy
  - Communication
  - Coordination
- Conclusions

Technology Trends for On-line Environments

- Web Services
  - *(UDDI, WSDL, WSFL, …)*
  - Discovery and usage of applications in public networks
- Semantic Web
  - *(XML/RDF, DAML+OIL, OWL, …)*
  - Formal description of Web Content
- Peer-to-Peer Networking
  - *(Gnutella, JXTA, Groove, …)*
  - Discovery, interactions, data sharing between end systems
- GRID Computing
  - *(Global Grid Forum)*
  - Massively parallel shared, worldwide computing resources
- E-Business Systems
  - *(ebXML, RosettaNET)*
  - Structured electronic Business-Business interactions
- "Web Programming"
  - *(Microsoft .NET / Sun ONE)*
  - Development kits for network enabled distributed application components

(some simplifications...)

Common Vision?



- A Web/Internet filled with “Smart” Automated Systems interacting with one another on behalf of the organisations
- Expect:
  - Automation: autonomous components
  - Meaningful interaction: formal semantics, flexible, provable meaning despite heterogeneity
  - Dynamic composition/coordination: automated contract creation, establishing first-time and long-term business relationships
  - Open: heterogeneous, large-scale, interconnected

Example Scenario

- Public Call for tender
  - Integrated survey of on-line information on headaches
- System X (belonging to company X) finds partners
  - Web indexing systems (Y[1 … N]), medical expert (Z) owned by other companies
- Agree strategy and actions
  - Terms and conditions, virtual company formation, contracts
- Collaborative Action
  - Over time, meeting constraints in the face of change, in success or failure

The “Web”

- The web is
  - Autonomous: Web sites individually owned and administered
  - Heterogeneous: Web sites on a vast range of topics
  - Open: anybody can launch a site, any site could be taken down
- Limited or No
  - Interaction (*between services*)
    - Beginning with web focused EAI tools (using XML for example)
  - Dynamic composition generally based on manual integration
    - Limited forms (RDF/RSS(?) based syndication for example)

Standard Middleware Environments

- Standard Applications using (e.g) CORBA, SOAP etc. generally involve:
  - Interaction: complex interactions specified as interfaces (very flexible if not very abstract)
  - Autonomy: objects involve could be made very opaque / generate their own events

- Limited or no:
  - Heterogeneity: generally focused on a particular application
  - Openess: generally know all objects or object types that will be involved
  - Dynamic composition: often only done static
    - Becoming more dynamic with (e.g.) Jini.

Peer-to-Peer Networks

- Classic P2P networks (Gnutella, JXTA) are normally
  - Autonomous: many nodes operated by individuals,
  - Open: nodes join and leave dynamically

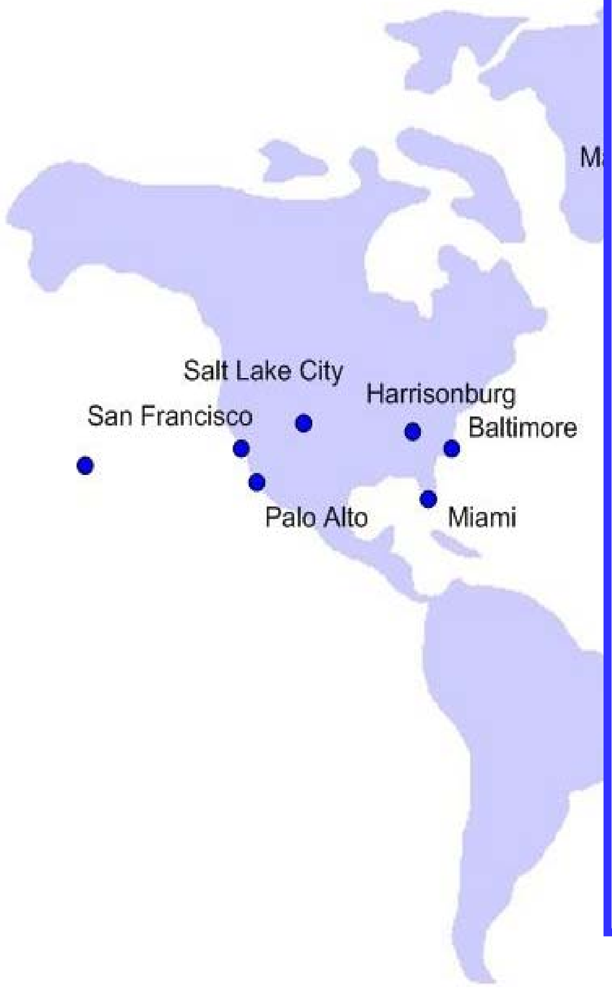
- Limited or no:
  - Interaction: restricted to an application specific API (napster, gnutella)
  - Heterogeneity: although content is often heterogeneous – most *networks* are often application specific
  - Dynamic composition: mostly limited to structured search in a particular domain

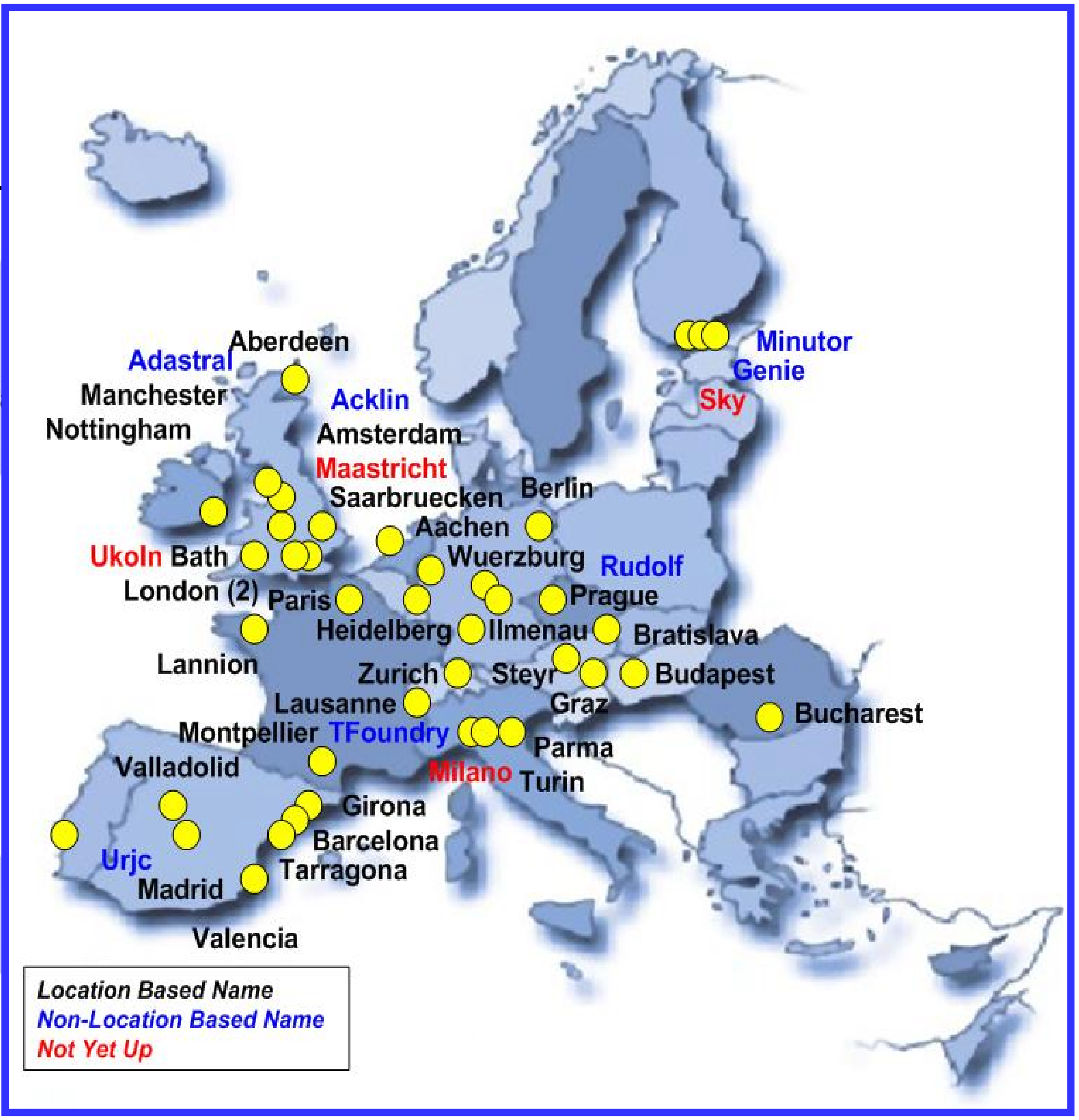
>>> Agentcities

Large-scale Open Testbed for Agent Based Services

- 100+ organizations involved worldwide
  - Participating in an open test environment
  - Long term deployment, evolution and integration of technologies
  - Key technology issues
    - *Service interaction / semantics*
    - *Service composition*
    - *Automating service components*
- Concrete terms most groups work on:
  - Particular technology trials
  - Particular application focus
- This is the "*messy end*" of the research spectrum

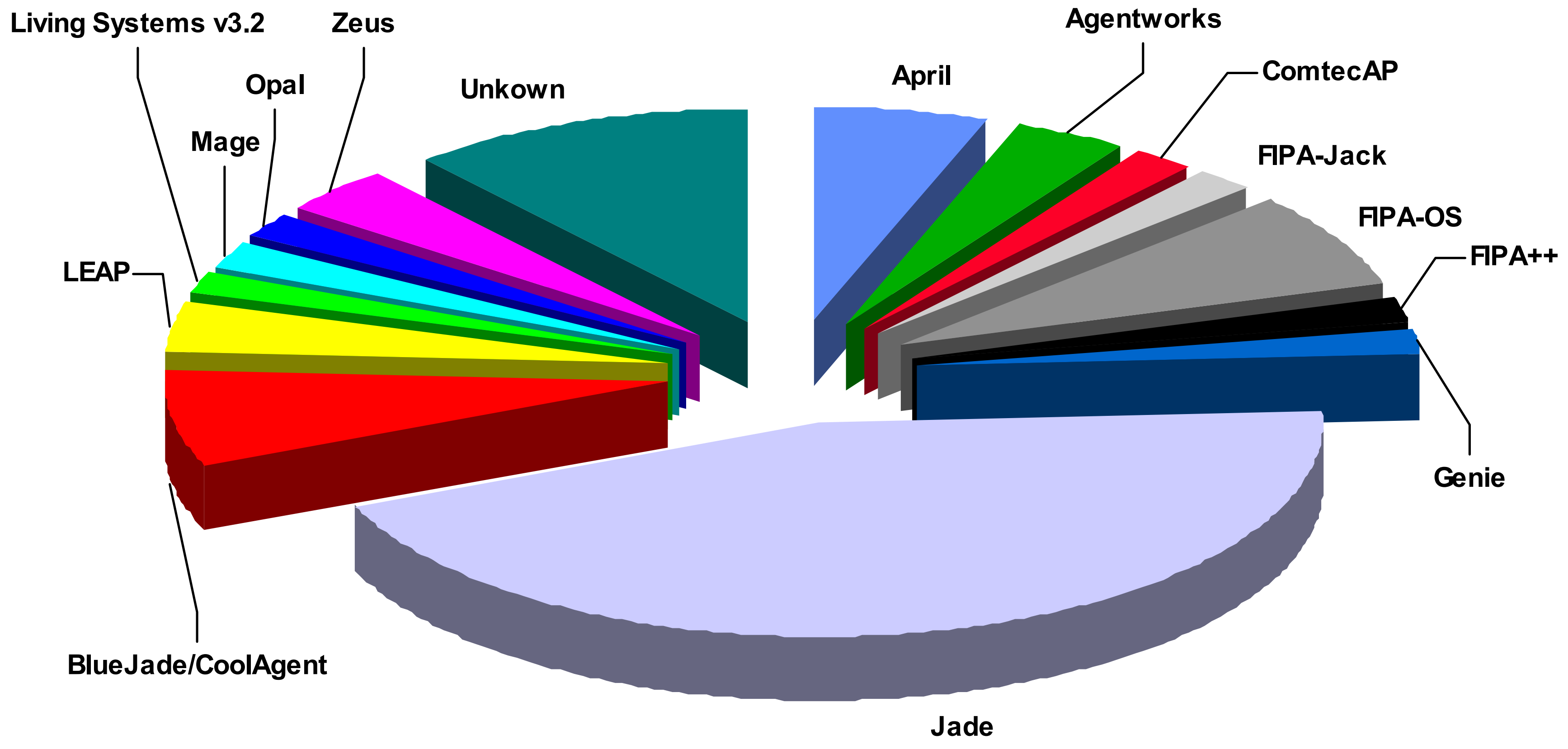
Network


Salt Lake City
San Francisco
Harrisonburg
Baltimore
Palo Alto
Miami
Ma


Adastral
Aberdeen
Manchester
Nottingham
Acklin
Amsterdam
Maastricht
Saarbruecken
Berlin
Aachen
Ukoln
Bath
Wuerzburg
Rudolf
London (2)
Paris
Prague
Heidelberg
Ilmenau
Bratislava
Lannion
Zurich
Steyr
Budapest
Lausanne
Graz
Bucharest
Montpellier
TFoundry
Parma
Valladolid
Milano
Turin
Girona
Barcelona
Urjc
Tarragona
Madrid
Valencia
Minutor
Genie
Sky
Location Based Name
Non-Location Based Name
Not Yet Up

FIPA Agent Platforms in The Network

- July 2002 (of 50 platforms)



What is the point of doing this?

- This is another tool in the tool box
- Stepping stone to mature systems
- Real open system problems
- Technology evolution rather than revolution?

Modeling an Agent Service

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

http://www.lsi.upc.es/~webia/KEMLG

AGENTCITIES

Deploying a Platform

- Machine
  - Any O/S
  - Outside the firewall (or with necessary access)
  - Permanent connection if possible
- FIPA Compliant Agent Platform
  - “Roll your own”
  - … or use one of the 20 or so available
- Install and Deploy the Platform
  - External Address (FIPA HTTP MTP)
- Register the platform
  - Create a group on http://www.agentcities.net
  - Register you platform data (address, name, …)
  - Activate the monitoring services

es

What’s needed? Communication Stack

Level	Description	Example
Conversation	Sequence of communicative acts related to a particular topic	Communicating about buying and eating an apple
Communicative Act	Communication about a piece of content	Requesting somebody to perform the action of…
Content Expression	Description of states of the world over objects	Expressing the action of eating an apple
Ontology	Description of objects in the domain	Meaning of “apple” and “eat”
Syntax	Representation of Content	HTML, JPG, SQL
Protocol	Data exchange protocol (ISO layer 7)	HTTP, GIIOP, SMTP
Transport	Physical transport and low level transport protocols (ISO layers 1-6)	Optical Fiber, TCP-IP etc.

Design Methodology

- Service Specification
  - What functionality do we want? What agents do we need?
- Design Sequence
  - Protocols: map into interaction sequences
  - Performatives: find out what performatives you need
  - Content Expressions: work out what content is needed
  - Ontology: build a domain model
- “Top down design”
  - Bottom up design might also be valid for very generic services – e.g. for an “Oracle” of generic knowledge

Modeling Background

Need for Ontology

- Ontology defines
  - The things in the world and their relationships
  - Conceptualisation, Vocabulary, Axiomatisation
- Need
  - The meaning of the things referred to in the domain (Vocabulary)
  - To know what possible values exist (Vocabulary)
  - To know about underlying model / relationships between items (Conceptualisation)
  - To know how to make inferences (Axioms)
- Required
  - To effectively interpret return values and parameter values
- Ontology defines the application domain and its boundaries.

Need for Structured Content Expression

- Relate ontology items to each other (e.g. return value or a parameter)
- public **String** myStrangeFunction(x,y,z)
  - Need to know the structure of the encoding for the String
  - Need to know how it links to the ontology
- Example String = “|1|207|#145678|lamp.bulb.filament|”
  - Arbitrary encoding
  - Unknown potential set of values (is “|1|y|3|….” allowed?)
- Content Languages structure content communication

Need for Performatives

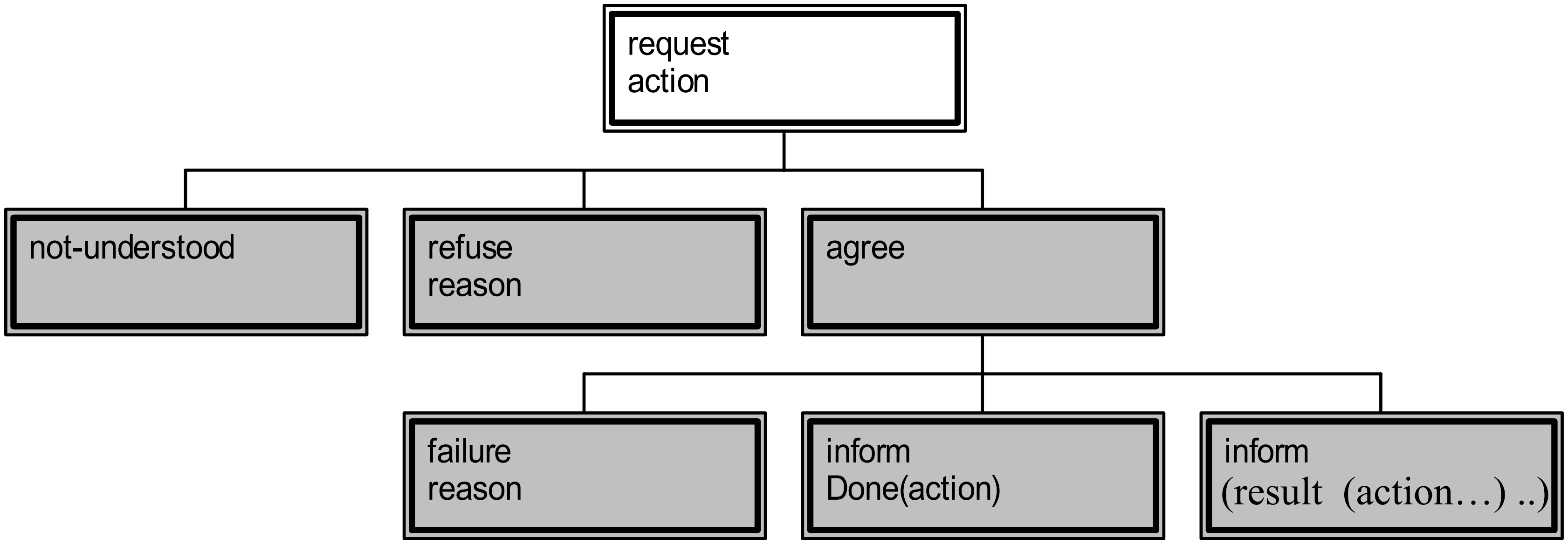
- Two models for APIs:
  1. APIs have one “communication act” with semantics “Do this” - easily understood but very general
  2. APIs have a complex infinity of “communication acts” – user defined and often with no semantics defined
- Understanding the Semantics of a method call (action)
  1. **Easy:** but the complexity is pushed to #2 anyway
  2. **Hard:** you may only have function names… (Myth: function names can be self explanatory…)
- Performatives pick a fixed action set with
  - Well defined / agreed semantics
  - Broad coverage of most applications

Need for Interaction Protocols

- String together several communicative acts into a useful sequence
- APIs :
  - Provide for *single method calls*
  - Call method (“doX()”), then get one return (“give me the result”)
  - Multiple steps are possible only if the object retains state and/or there is a synchronous connection
- In general we require
  - Extended Interactions with multiple steps
  - Asynchronous Interactions
  - Link with Semantics – (what does a certain sequence mean in total?)
- Protocols
  - Link a sequence of (inter)actions with a coherent aim / semantic description.

Box Diagram Descriptions

- Old Fipa standard description
  - New method is AUML (see specification XC00025)
  - Will use this box version in this course



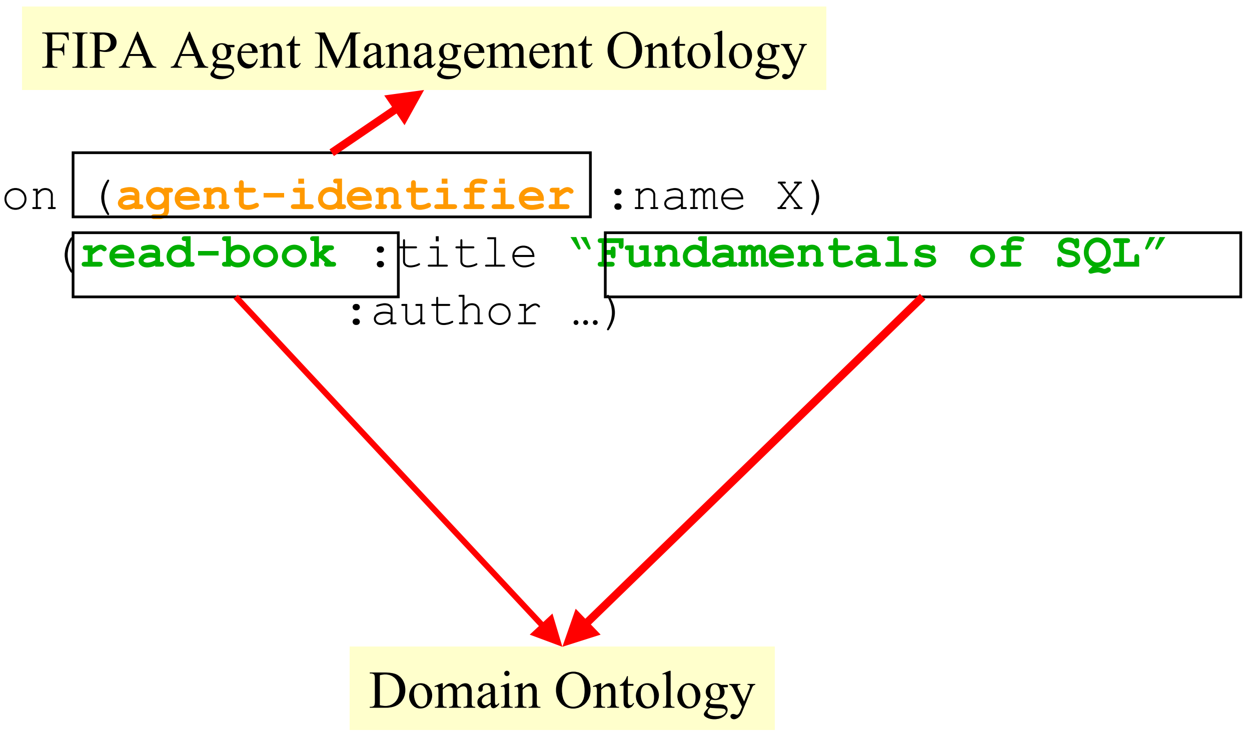
Link Interaction Protocols to Structured Information Exchange

- Different types of media use different models
  - E.g. TV – streaming, email – asynchronous message passing,
- Agent Communication: asynchronous message passing
  - Telephone/email metaphor is often useful
    - Q: “Hi, who won the American Presidency?”
    - Potential A: “It was Bush/Gore”
    - Potential A: “Unable to answer – please wait while we do a recount for you”…
- Phrases are atoms which can be mapped to individual messages

SL Relationship to Ontology

- Terminals in SL often bottom out into
  - Strings
  - Numerical constants
    - These need to be replaced by things in the ontology

- Example



```
((action (agent-identifier :name X)
         (read-book :title "Fundamentals of SQL"
                    :author …)
  )
)
```

Table Frame Representation

Frame	agent-identifier
Ontology	FIPA-Agent-Management

Parameter	**Description**	**Presence**	**Type**	**Res Values**

name	The symbolic name of the agent.	Mandatory	Word	`df@`*`hap`*
				`ams@hap`

addresses	A sequence of ordered transport addresses where the agent can be contacted. The order implies a preference relation of the agent to receive messages over that address.	Optional	Sequence of `URL`	
resolvers	A sequence of ordered AIDs where name resolution services for the agent can be contacted. The order in the sequence implies a preference in the list of resolvers.	Optional	Sequence of `agent-identifier`	

Example Agent-identifier

- Example
  - (agent-identifier

```
        :name bill.clinton
        :addresses (sequence http://www.whitehouse.gov, …)
        :resolvers (sequence (agent-identifier :name hillary.clinton
        …)
    )
```

- Green underlined items are named in the ontology
- *Orange italic* items are values (may or may not be in the ontology)
- Black items part of SL syntax

Modeling Steps

Service Specification

- Domain?
  - Hotel services
- What does the thing do?
  - Actors involved – potential agents (Hotel Agent, Personal Agent)
  - Services provided:
    1. (INFO) Information about the hotel
    2. (BOOKING) Room booking service
- Considerations
  - Interactions with other agents, larger scale application, flexibility and service usage etc.

Modeling Example

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

http://www.lsi.upc.es/~webia/KEMLG

AGENTCITIES

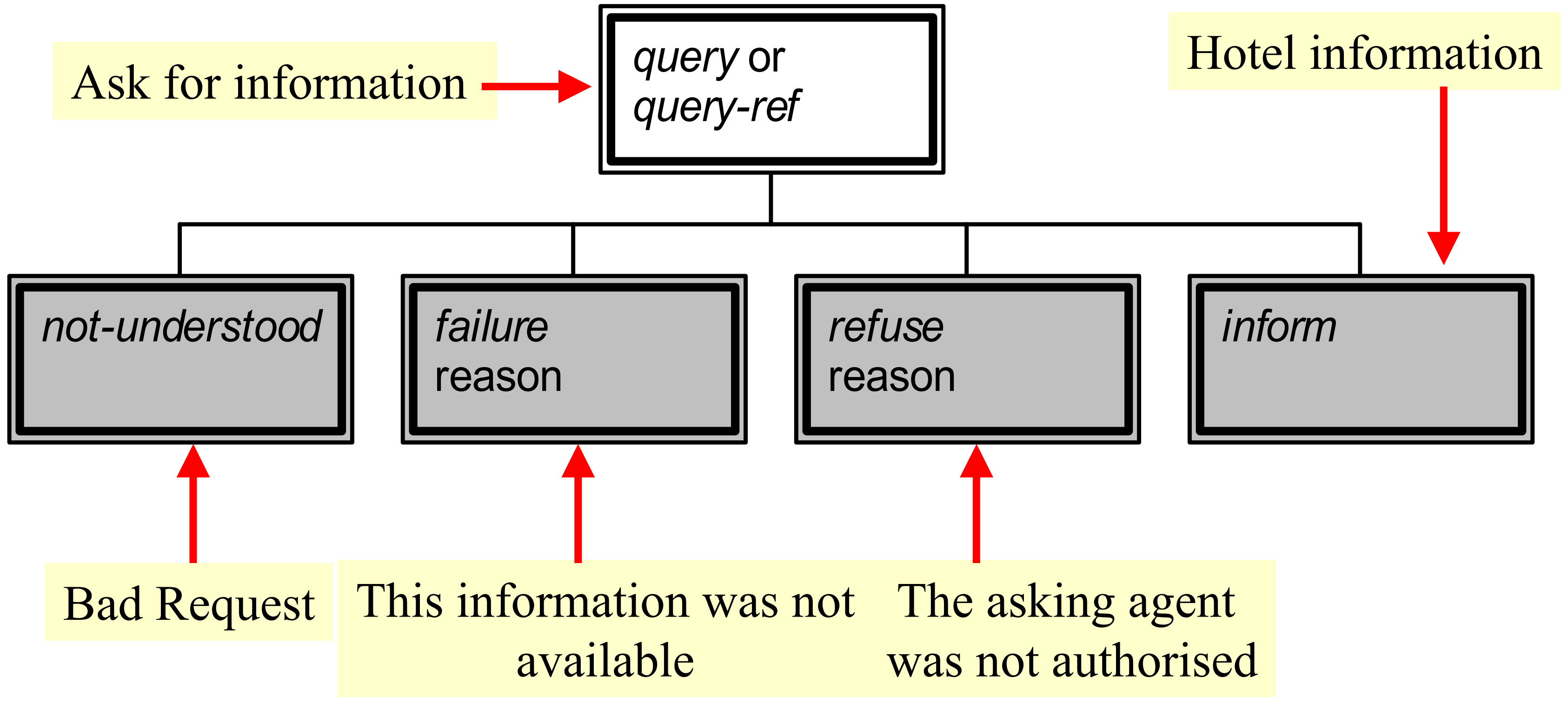
Map to Interaction Protocols

makeProtocols(service_spec)

Map to Protocols

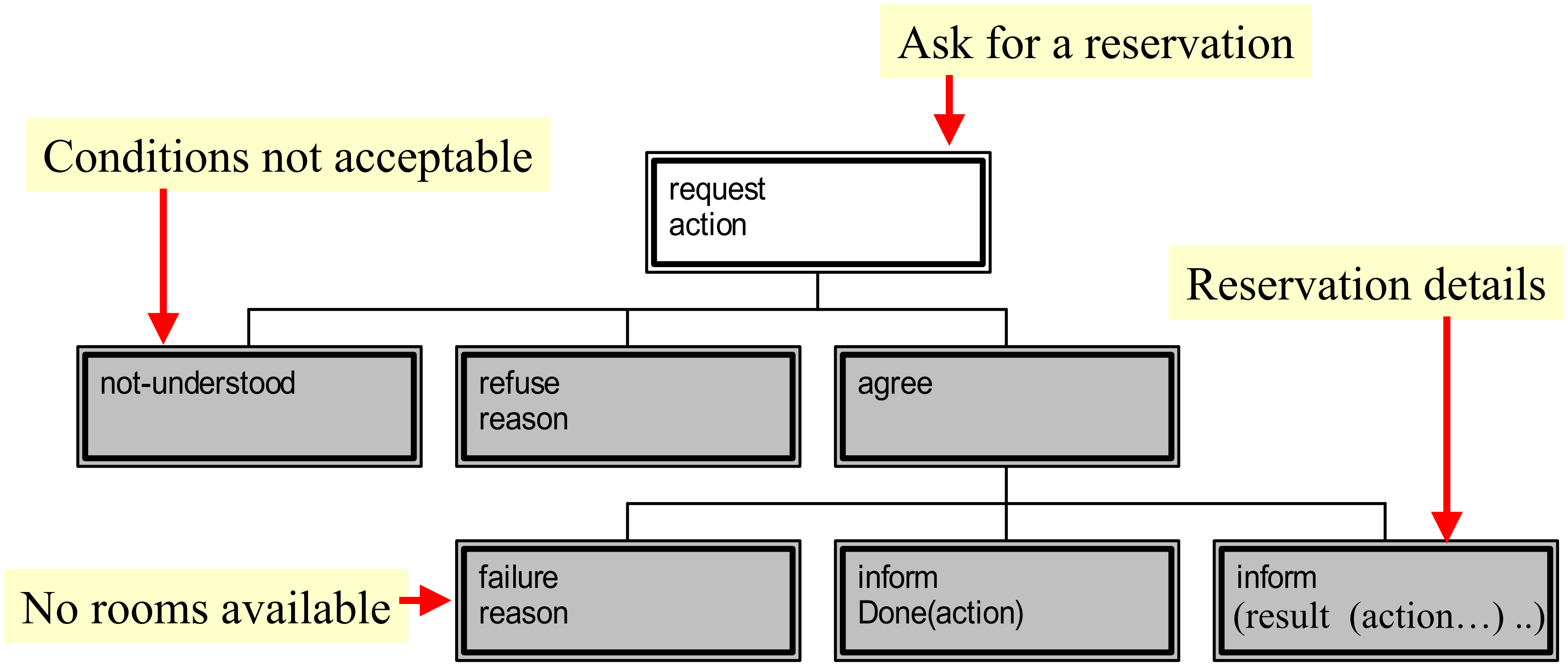
- Telephone conversations
  - "What facilities does your hotel have?"
    - "The hotel has…"
  - "I'd like to book a room for X people, arriving A, departing B"
    - "Ok, confirmed" OR "Sorry, no rooms available for this period"
- Suggests Protocols
  - FIPA-query
  - FIPA-request
- New Protocols
  - Reformulate the questions?
  - Maybe need new protocols… -> performatives

(INFO) Query Protocol



- Match the steps to hotel Information service (examples)…

(BOOKING) Request Protocol



- Match the steps to Hotel reservation action…

Generate Performatives…

getPerformatives(protocols, service_spec)

List of Performatives Used

- Request
- Not-understood
- Refuse
- Agree
- Failure
- Inform
- Query
- Query-ref

$\longrightarrow$ Each with its own semantics and usage

Check the *Semantics*

- Each performative has
  - A natural language description
  - A corresponding formal logic description
- Do they fit with your application?
  - Do your agents react in the correct way to the performatives?
- Some of the semantics may seem unintuitive
  - This probably means you ought to be using another performative…

Check the *Content*

- Each performative has strict requirements on the content required for the message
  - Given in the “Message Content / Description” field
- Might be Basic Types
  - Objects (IRE), Actions, Propositions
- Might be Tuples
  - E.g. call-for-proposal: (<action>, <referential expression for a proposition>)

(INFO) Query-ref

- Description
  - The action of asking another agent for the object referred to by an referential expression
- Example
  - (query-ref

```
  :sender (agent-identifier :name i)
  :receiver (agent-identifier :name j)
  :content
     <a referential expression referring to some object which i
     wants to know about – e.g. a hotel>
  )
```

(INFO) Inform

- Description
  - The sender informs the receiver that a given proposition is true.
- Example
  - (inform

```
   :sender (agent-identifier :name i)
   :receiver (agent-identifier :name j)
   :content
      <a proposition stating that something is true or false – in
      this case that it is true that a given object (hotel description)
      corresponds to an object referenced in a previous query.>
   )
```

(BOOKING) Request

- Description
  - The sender requests the receiver to perform some action.
- Example
  - (request

```
:sender (agent-identifier :name i)
:receiver (agent-identifier :name j)
:content
   <an action expression of the action i wishes a particular
   agent to carry out (not necessarily j) – in our case – the act
   of reserving a hotel room>
)
```

Generate Content Expressions…

findContentExpressions(performatives, service_spec)

Different Content Types

- As we already saw – there are three main types of content:
  - Actions: E.g. – booking a room
  - Objects (IREs): E.g. – a hotel description
  - Propositions: E.g. – a statement about room availability or about the fact that a given object corresponds to a query

- Different types have different syntax structures and allowed components

Object: A Hotel Description

- Example
  - (<hotel description key word>
    <attribute-value pair>
    <attribute-value pair>
    …)
- Needs to be in a referential expression such as:
  - (iota ?x <some wff which generates the object…>)
  - (any ?x <some wff which generates the object…>)
  - (all ?x <some wff which generates the object…>)

Well formed formula

Proposition: Information about an Object

- Example
  - ((= <the object reference we used in the hotel query>
    <the object found in the database which matches the query – a hotel description of some sort>
    ))
  - The "=" is a binary operator between terms which states their equality
  - With this kind of statement we can respond to a query

Action: Booking a Room

- Example
  - ((action (agent-identifier :name <name of hotel agent>)
    <some function which expresses the room booking
    – and also (presumably) some conditions on it>
    ))

- The action in this case asks an agent to execute a function – which must be defined in the ontology

Proposition: Statement about action outcome

- Example
  - ((result (action <agent which carried out the action> <action expression>)
    <predicate describing the outcome - potentially with some arguments which are to provide additional information>
    ))
- Or…
  - ((done (action <agent which carried out the action> <the action that was requested>)))
- Note this a statement which says
  - The predicate must indicate the truth/falsity of something – in this case the fact that the room booking was carried out.

Generate Domain Ontology

generateOntology(content, service_spec, domain_model)

Objects in the Domain

- The content expressions guide what we might need to about in the domain
- Ontology must express
  - **Conceptualisation**: underlying model
    - given by the frame representation
    - Relationships basically "part-of" hierarchy only
  - **Vocabulary**: objects, propositions, functions
  - **Axiomatisation**:
    - in this case limited/none (only on the part-of hierarchy)
    - no strong basis for inference
- In practice
  - Work on the Vocabulary…

Three types of Entity in the Vocabulary

- Objects
  - Refer to concrete objects in the world
  - Can be components of other objects (“part-of”)
- Propositions
  - Functions which are evaluated to logical truth/falsity only
- Functions
  - Functions which can evaluate to any value
  - Used loosely here since they often result in “side effects”

Hotel_description Object

Frame	hotel-description			
Ontology	Hotel-ontology			
Slot	**Description**	**Presence**	**Type**	**Res Values**
name	The name of the hotel.	Mandatory	Word	
description	A text description of the hotel	Optional	String	
Number-of-stars	The quality of the hotel – expressed in a number of stars	Optional	Integer between 0 and 5.	0,1,2,3,4,5 .

`is_hotel_description` Predicate

Predicate	is-hotel-description
Description	The predicate has two arguments: 1) an object, 2) a set of constraints. The predicate evaluates to true IFF: 1. The object corresponds to a hotel description. 2. The constraints are expressed in the second argument take the form “attribute-name = attribute-value”. 3. The attribute values of the hotel are such that they meet the constraints expressed in the second argument
Arguments	Object, Set of String.
Result	Boolean

is_hotel_description

- Example 1.
  - (is-hotel-description
    (hotel-description :name “Beau-Rivage Palace”
    :number-of-stars 5)
    (set “number-of-stars = 5”)
)

    Object
  - Evaluates to *true*.

    Variable
- Example 2.
  - (is-hotel-description ?x (set “number-of-stars = 4”))
  - Evaluates to true IFF there exits a known object corresponding to a 4* hotel.

`Make-booking` Function

Function	make-booking
Description	The function has three arguments: 1) a number of people, 2) an arrival date, 3) a departure date. The function has two potential outcomes: 1. A physical booking according to the criteria given in the arguments is made and an object describing the booking is returned. 2. No booking is made and a failure is announces (Note the conditions for 1. May depend on many factors – availability, subjective decision of the agent executing the function etc.) The physical real-world booking is essentially a “side effect” of the function.
Arguments	Integer, DateTime, DateTime.
Result	A `room-booking-description*` object / null (in case of failure)

* Not given in these slides

Final Message Examples

Message Exchanges for Hotel Service

(INFO) Query for Information

- Example
  - (query-ref

```
:sender (agent-identifier :name i)
:receiver (agent-identifier :name j)
:content
  ( (iota ?x (is-hotel-description ?x
                                  (set “name = Beau Rivage Palace”)
                  )
      )
    )
  )
```

  - Asking for a description of a hotel called the “Beau Rivage Palace”

(INFO) Response to Query

- Example
  - (inform

```
 :sender (agent-identifier :name i)
 :receiver (agent-identifier :name j)
 :content
   (  ( = (iota ?x (is-hotel-description ?x
                                          (set "name = Beau Rivage Palace")))
        (hotel-description :name "Beau Rivage Palace"
                                   :number-of-stars 5)
      )
   )
 )
```

The ?x evaluated to this Object

  - The ?x evaluated to the object describing the BR Palace with 5 stars.

(BOOKING) Request

- Example
  - (request

```
:sender (agent-identifier :name i)
:receiver (agent-identifier :name j)
:content
   ((action (agent-identifier :name j)
          (make-booking 5 26-11-2000 29-11-2000)
    )
   )
)
```

Note: date format would need to be agreed upon

(BOOKING) Response to Request

- Example
  - (inform

```
:sender (agent-identifier :name i)
:receiver (agent-identifier :name j)
:content
   ((result (action … )
            (booking-complete
              (agent-identifier i)
              (room-booking-description
                         5
                         26-11-2000
                         29-11-2000 ))
   )
)
```

Predicate confirming the booking

Note: code example uses (done …)

Service Modelling Tips

Extra slides…

Distribute Complexity over Layers

-
  ```
  (query-ref
    :sender …i
    :receiver …j
    :content
      (iota ?x (is_car
                  :color red
                  :make ford
                )
      )
   )
  ```

-
  ```
  (request
    :sender …i
    :receiver …j
    :content
     (action j
      (send-information
        (iota ?x (is_car
                    :color red
                    :make ford
                  )
        )
   ))
  ```

Could be defined in any
Way the programmer likes

Distribute Complexity over Layers

- In the two examples on the previous page - the two versions are potentially equivalent
  - In the right hand version – semantics are redefined in the ontology
  - In principle: we could have a performative for everything or put everything in the ontology and just use
- In principle we could push semantics UP:
  1. Have a performative for everything
  2. Have very limited content language
- Or push semantics DOWN
  1. Use only “request”
  2. Define the semantics every precise request requires in the Ontology

Focus on Application Characteristics

- What is happening at the *application level*?
  - Try to abstract away implementation details
- Make ontologies generic
  - Concentrate on general descriptions of the world which feed into your application
  - Leads to greater re-use
- Focus on the goals of the interactions between your agents
  - Not on the details of how your particular mechanism works

Use Real Data Sources

- World (web) full of information
  - Ontologies and service models exist for many domains
  - When working in a domain identify the industry standards body or group which steers consensus
- Data sources
  - International and business organisations
  - Domain leading web sites
  - ebXML, BizTalk, UDDI, Jini community -> all producing, storing XML based examples.

Well not Always! ☺

Work Top Down

- Keeps you focused on your application
  - Restrict your view (the generally intelligent agent is a while away…)
  - Build simple models first
  - Select the correct granularity (is a hotel an agent or should it be a hotel chain?)
  - Match the real world in granularity (hotels are independent but don't often have their own websites…)
- Smaller granularity
  - More work
  - More potential re-use in other services

Be Rigorous

- Formal models are important
  - Worse than useless if they are not adhered to…
- Formal models are almost never adequate
  - If you need to deviate from them
    - Do so in a principled way
    - Document how and why you deviated
    - Feed them back to the community using the framework
- Be rigorous in your descriptions
  - Precise specifications,
  - Formal grammars,
  - Correct use of the agreed semantics

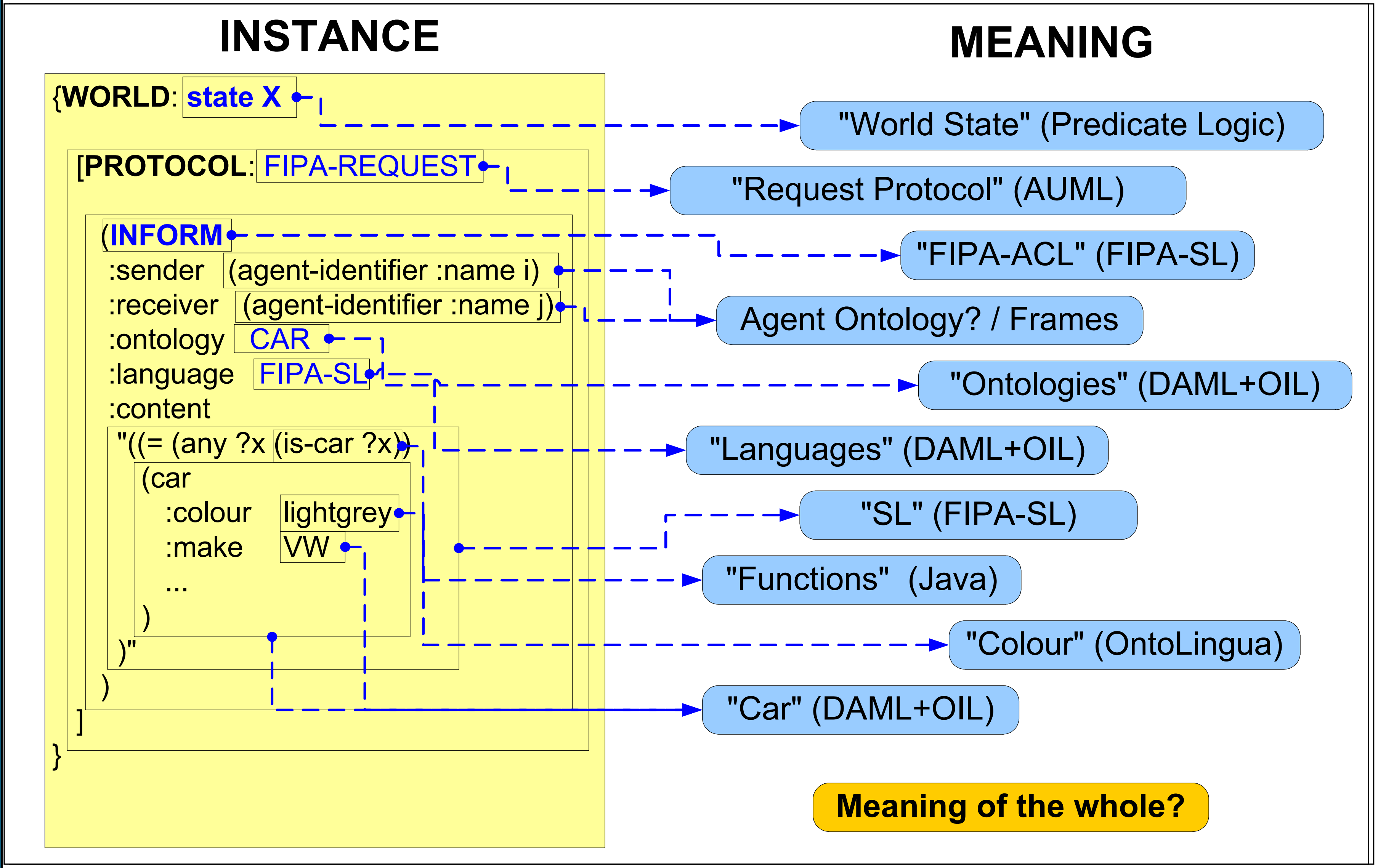
>>> Challenge Areas

So why is all this useful?

I: Communication

- Communication is
  - Two or more systems interacting (signaling to one another*)
- “Useful” communication
  - Unambiguous shared meaning of interaction (*)
  - Creates shared understanding between systems
  - Potentially changing the state of the world (an “action”)
- Implicit v’s Explicit semantics
  - Explicit: formal shared description of meaning
  - Implicit semantics: coded into the end systems
  - Explicit semantics are essential for *flexible interactions*, *dynamic worlds*, *open systems* – they underpin reasoning about communication

I: Using the Semantics?



I: Steps to Reach for Communication

- Step one:
  - Formalisms to describe (off line) the meanings of constrained classes interactions (and hence determine their consequences) incorporating formalisms from all levels – supporting system design.
  - Systems have very fixed interaction patterns

?

- Step two:
  - Formalisms can be used generatively in narrow ranges to enable flexibility to creep into interactions

?

- Step three:
  - Agents can reason about the meaning (and impact) of arbitrary messages in the subset of formalisms it is familiar with.

?

- Step four:
  - Universal languages for arbitrary system-system interaction.

?

I: But what we have is already useful

- We are boiling down to what we need
- There is now little argument over the levels
- Humans can have a stab at interpreting the meaning given all the formalisms
- The levels might hold the key to reasoning (by abstraction)
- It is difficult to exploit domain specific traits – people seek generic languages
- Domain specific ontologies do not necessarily help
  - It is the expressiveness of the ontology language which hurts
  - At least there is an ontology language convergence

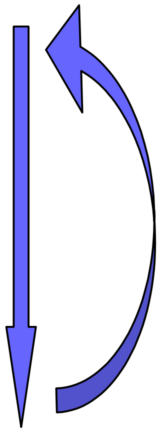
II: Coordination

- What are the *implications* of communicating
  - Agreements?
  - Commitments?
  - Contracts?
- How can a system achieve its goals in the world
  - On its own?
  - By coordinating with others?
- Interactions should generate a set of commitments between parties

1. Guiding System Behavior
   - Discovery
   - Reasoning
   - Coordination (composition of services)
   - Delegation of Authority
2. Creating environments (or “institutions”) for “interactions with consequences”
   - Semantics in context
   - Enforcement
   - Identity, Trust
3. Understanding system dynamics

II: Generic Coordination Cycle

- Cycle
  1. *Problem Identification*
  2. *Service Discovery*
  3. *Team Formation*
  4. *Plan Formation*
  5. *Joint Action*



- Seen in a wide range of literature
  - Levels distinct but interdependent
  - Many techniques for different levels

- Relevance very widespread

III: Where are we technically?

- Research
  - Negotiation
  - Markets
  - Contract Nets
  - Organisation Structures
  - Social Rules
  - …
- Theories and prototypes
- Significant number of useful example applications
- Focus
  - Generic solutions
  - Very flexible agent behaviour

- Industry
- Focus
  - Rigid coordination patterns
  - Proven failsafe interactions / transactions
- But rapid *progress* and *standardization*:
  - ebXML
  - rosettaNET
  - WSFL/XLANG/…
  - UDDI

Steps to Reach for Coordination

- Step one:
  - Systems are able to establish fixed template contracts with one another on demand with predetermined business partners on behalf of an organization
  - Covered by a high level “human level” agreement

Now

- Step two:
  - Fixed agreements with previously unknown trading partners (no high level agreement in force), multiple systems in the same agreement on behalf of the organization

?

- Step three:
  - Agreements in a specified flexible range with previously unknown trading partners on behalf of the organization

?

- Step four:
  - Any agreement with anybody – on behalf of itself.

?

III: Automation

- Software Engineering and Control Problems
  - Environments containing systems with different owners
  - Actions with consequences
- Legal / Social Problems
  - Who is the software acting on behalf of?
  - Consequences of failures / errors
  - If we don't apply human laws – what will we apply?
  - European laws prohibit automating actions which may endanger the life of a human being

III: Where are we technically?

- Integration with Enterprise Java Bean Systems
  - Stability, Fault tolerance, …
  - Philosophical debate
- More powerful reasoning tools
  - JESS, JAM, Planners, …
  - Backend onto:
    - Contraint solvers
    - Planners
    - Schedulers

III: Proactive EJBs?

- EJB/Agent models
  - EJB model prohibits threads (e.g. Entity bean is reactive at best)
  - Agents must be "autonomous" – generally means
    - Has its own goals and is "proactive" in achieving them
    - Many people interpret this to mean agents must have their own thread of control

- What does **proactive** really mean?
- Why does the EJB model disallow threads?

III: Example One

- Example I: Security Agent
  - Receives 1000 authorization requests a minute
  - Goal: only allow in agents with a valid security ID
- Reactive: on demand service
- Proactive: ?
- Autonomous: ?

III: Example Two

- Example II: Group Communication Moderator
  - Connected to a group communication system
  - Goal: prevent abuse (spamming, offensive content etc.)
  - Agent Monitors the environment and builds up models of bad users over time – then when a threshold is reached expels them
- Reactive: last message takes it over the threshold
- Proactive: ?
- Autonomous: ?

Automation

- Step One:
  - Predetermined actions, Y/N evaluations using rules/triggers
  - All actions explicitly authorized ?
- Step Two:
  - Goal setting, action pattern selection, minimal reasoning, action suggestion
  - Actions authorized by special control processes ?
- Step Three:
  - Goal setting – reasoning/planning ?
  - Automatically takes actions within constraints
- Step Four:
  - Autonomous action – establishing ?

>>> Lessons & Challenges

Agentcities Lessons

- Think holistic but not homogenous
  - Example: Use of DAML-OIL with FIPA-SL or KIF is non-trivial.
  - Example: FIPA-ACL is the standard for Agent Communication Language – places significant constraints on the content.

- Autonomy
  - Interactions between automated agents quickly get out of hand - service testing often caused log/buffer/DB explosions simply due to 24/7 automated testing
  - Partly a platform robustness problem – partly (critically) an application level problem

Agentcities Lessons

- Communication
  - A *very* large proportion of effort goes into modeling
  - It is still very difficult to find suitable ontology resources, let alone have agents automatically discover them.
  - Nobody wants to give you their “whole” ontology – need to be able to work with partial/incremental data – granularity problem.
- Coordination
  - Most coordination mechanisms require careful casting into semantic frameworks
  - Service descriptions have the same granularity problem as ontologies

Agentcities Lessons

- Worry about security
  - HACKFORCE.Agentcities.Net attacks.
  - Powers of the languages involved are a strength and a weakness.
  - Integration of agent tools with industrial strength application servers.

- Service deployment is hard … but worth it ☺
  - The devil is in the details
  - The problems aren't necessarily what you think they are
  - New on line environments will be messy ecosystems

>>> Conclusions

Example Services

Hotel Service

Restaurant Review

Restaurant Booking

Theatre Recommender

GIS Service

Restaurant Finder

Auction House

Transport Info Service

Ontology Service

Trade House

Agent Directory

Payment Service

Security Service

Platform Directory

SMEAccess Service

Service Directory

Three Challenge Areas

- **Network Architecture**
  - Entities in the world
  - Mappings to concrete technologies
  - Services
- **Coordination Framework**
  - Abstract coordination cycle
  - Mappings to agreed technologies
  - Institutions / Norms / Organisations
  - Services
- **Communication Framework**
  - Abstract communication
  - Mappings to concrete technologies
  - Services

Questions

?

Para más información contactar con:

Steven Willmott (steve@lsi.upc.es)

Resources

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

http://www.lsi.upc.es/~webia/KEMLG

AGENTCITIES

Web sites

- Agentcities
  - http://www.agentcities.org/
  - http://www.agentcities.net/
- FIPA
  - http://www.fipa.org
- Agents
  - http://www.agentlink.org/

Agent Platforms

- Well Known Open Source Platforms
  - April Agent Platform: http://sourceforge.net/projects/networkagent
  - Comtec AP: http://ias.comtec.co.jp/ap/
  - FIPA-OS: http://fipa-os.sourceforge.net/
  - JADE: http://sharon.cselt.it/projects/jade/
  - LEAP: http://leap.crm-paris.com/
  - ZEUS: http://www.btexact.com/projects/agents/zeus/
- Fuller list of tools
  - http://www.agentcities.org/Resources/

Other Technologies

- Semantic Web
  - http://www.w3c.org/
  - http://www.semanticweb.org
  - http://www.daml.org
- Web Services
  - http://www.w3c.org/
  - http://www.ws-I.org/