Session 3: Agent communication and knowledge sharing

Behrouz Far
Schulich School of Engineering, University of Calgary
far@ucalgary.ca
http://www.enel.ucalgary.ca/People/far/
Course Curriculum

Overview of agent-based SE

Methodologies for agent-based analysis and design

Agent communication & knowledge sharing

Agent-based System Architecture & Organization

FIPA: Foundation for Intelligent Physical Agents

Principles of Object Technology

Other topics: Agent Interaction, Infrastructure, APIs, Performance metrics, Learning, Self-organizing systems etc.
Session 3: Agent communication and knowledge sharing

- Introduction to Ontology Engineering
- KQML as an Agent Communication Language
Ontology Engineering

Behrouz Far
Department of Electrical & Computer Engineering,
University of Calgary
(far@ucalgary.ca)
When a politician says “yes” he actually means “no” and when a young and beautiful lady says “no” she actually means “yes”!

[George Bernard Shaw]

It is all about “concepts”, “meanings” and their “relations”
Contents

- Basic concepts and definitions
- Ontology as an engineering artifact
- Constructing ontologies
- Ontology research
- Applications and tools
- Conclusion
Example of Symbol Level Communication

- Signal-level
- Symbol-level

Agent 1

Signal level:
Dynamic Message passing

Symbol level:
Knowledge sharing

Agent 2

Let’s go to a Latin American restaurant

OK. I know a very good Mexican restaurant.

far@ucalgary.ca
Example: Database Search

- Query restaurant database
  - Example:
    
    served_food

    | R_id | Cuisine   |
    |------|-----------|
    | 1    | American  |
    | 2    | Mexican   |
    | 14   | Portuguese|

    SELECT * FROM served_food
    WHERE cuisine = 'Latin American';

- Returns ‘nil’
Example: Database Search

- Now suppose that we have an ontology
- What will an ontology aware query return?

> Portuguese
> Mexican

How to perform Semantic-Based Information Retrieval?
Knowledge Level Communication

- What is needed to communicate?

Using the same **terminology** is essential!
Knowledge Level Communication

What is needed to communicate?

I love turkey!

I love turkey, too.

Assigning the same meaning to terminology is also essential!

far@ucalgary.ca
Knowledge Level Communication

- What is needed to communicate?

I love turkey!

I love turkey, too.

Having the same context is also essential!
Knowledge Level Communication

- What is needed to communicate?

I love turkey!

I love turkey, too.

Is this enough to communicate?

Shared:
Terminology
Meaning
Context

Shared:
Terminology
Meaning
Context
Knowledge Level Communication

- **Communication language**: a medium through which the attitudes regarding the content of the exchange are communicated
  - A representation language
  - A communication language

Is this enough to communicate now?
Knowledge Level Communication

- **Interaction protocol**: knowing with whom to talk and how to find them, knowing how to initiate and maintain an exchange.

- **Transport protocol**: actual transport mechanism used for the communication: HTTP, TCP, SOAP, etc.

Is this enough to communicate now?
Knowledge Level Communication

- Communication as a social consensus
  - Not only between two agents but a society of agents
Knowledge Level Communication

- Signal-level
- Symbol-level

Let’s go to a Latin American restaurant

Agent 1

Signal level:
Dynamic Message passing

Symbol level:
Knowledge sharing

Agent 2

By the way, what are we going to do at the restaurant?

In communication, ontology is not an option but a must; several ontologies may be needed.

far@ucalgary.ca
History: Experience Factory

(1) Someone decides how knowledge should be recorded

(2) Someone else actually records knowledge

(3) Finally a user can query and use knowledge following rigid rules
History: ARPA Knowledge Sharing

- Aims at defining a common language for message passing among software modules in terms of inputs and outputs

- **KQML** and **KIF** as languages for information exchange and knowledge level communication

- **Ontolingua**: an ontology description language and design tool
Ontology: Definition

- The American Heritage Dictionary defines ontology as the branch of metaphysics that deals with the nature of being.

- The term ontology has been adopted by the AI community to refer to a set of concepts or terms that can be used to describe some area of knowledge or build a representation of it.

- Recently, the term ontology is usually used in conjunction with knowledge sharing and reuse and it has been extensively used in knowledge management (KM) research.

far@ucalgary.ca
What is Ontology? (1)

Types of AI theories

- **Mechanism theories**
  - “Secrets of making machines intelligent”
  - *e.g.*, rule-based system, neural nets, fuzzy logic, unification, etc.)

- **Content theories**
  - A kind of theory of the domain on which the mechanism theory works
  - Need a representation language
Ontology: Definition /2

- Ontologies are *content theories* about the sorts of objects, properties of the objects and relations between objects.
- Ontologies provide potential terms of describing our knowledge about the domain.
- Predicate logic augmented with *type-of* relation (to induce class hierarchies) can potentially be used to represent ontologies.
Two Approaches

- Ontology as a representational vocabulary
  - Conceptual structure of terms should remain unchanged during translation

- Ontology as the body of knowledge describing a domain (commonsense domain in particular)
Ontology: Definition

- Philosophy viewpoint
  - Dealing with the nature of being or reality (Theory of Existence)

- AI viewpoint
  - An explicit representation of conceptualization (Gruber, 1992)
  - A theory of what entities can exist in the mind of a knowledgeable agent (Wielinga, 1993)
Ontology: Definition

- A concept is a unit of thoughts consisting of two parts, the extension and the intension.
- The extension covers all objects belonging to this concept and the intension comprises all features valid for all those objects.

Citrous=[[color={}, [rind={thick}], [pulp={juicy}], [climate={warm}], [size={small, medium}]]

<table>
<thead>
<tr>
<th>O/F</th>
<th>color</th>
<th>rind</th>
<th>pulp</th>
<th>climate</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. apple</td>
<td>yellow</td>
<td>thin</td>
<td>dry</td>
<td>moderate</td>
<td>small</td>
</tr>
<tr>
<td>2. grapefruit</td>
<td>yellow</td>
<td>thick</td>
<td>juicy</td>
<td>warm</td>
<td>medium</td>
</tr>
<tr>
<td>3. lemon</td>
<td>yellow</td>
<td>thick</td>
<td>juicy</td>
<td>warm</td>
<td>small</td>
</tr>
<tr>
<td>4. orange</td>
<td>orange</td>
<td>thick</td>
<td>juicy</td>
<td>warm</td>
<td>small</td>
</tr>
</tbody>
</table>
Ontology: Definition /6

- Knowledge Base viewpoint
  - A system of primitive vocabulary/concepts and their relationships used for building artificial systems (Mizoguchi, 1993)

No overall and generally accepted definition of ontology exists!
Ontology in KM

- In knowledge management applications, ontologies are typically used for three purposes:
  - (1) to **support knowledge visualization**, where ontologies are inspected in order to create new knowledge by analysis and recombination of existing knowledge.
  - (2) to **support knowledge search, retrieval and personalization**, where ontologies are used to improve search and retrieval of information.
  - (3) to serve as the basis for **reasoning and reuse**, where some degree of formality of ontologies allows partial automation of problem solving and reasoning.
Ontology: Properties

- Ontologies have to deal with
  - **Heterogeneity in space**: ontology as networks of meaning
    - Tools for dealing with conflicting definitions and strong support in interweaving local theories are essential in order to make this technology workable and scalable
  - **Development in time**: learning and adapting ontologies
    - Having protocols for the process of evolving ontologies is a real challenge. Evolving over time is an essential requirement for ontologies
Why Ontology? (1)

- **Knowledge sharing problem:**
  - Different systems use different concepts and terms for describing domains. This makes it difficult to take knowledge out of one system and use it in another.
  - Ontologies may be used in an integration task to describe the semantics of the information sources and to make the content explicit.
  - Developing **reusable ontologies** that facilitates sharing and reuse is an important goal of ontology research.
  - Developing **tools** supporting building, merging and translating between ontologies is another goal.
Why Ontology? (2)

- Explicit ontology supports
  - Shared understanding among agents
  - Interoperability between tools
  - Systems engineering
  - Reusability
  - Defining the vocabulary and concepts
  - Plays the role of Mathematics in Physics (Theory of content)
    - Defines the necessary and sufficient concepts and “methods”
    - Common knowledge theory
What is it about?

Ontology is about

- Concepts
- Meaning
- Pragmatics
Ontology: Semantics

It is about meaning ...
Ontology as Artifact (1)

- Today, ontology has grown beyond philosophy and now has strong connections to information technology (e.g., Object Oriented systems, WWW and AI)

- Thus, research on ontology in AI and information systems has had to produce pragmatically useful proposals for how to construct and maintain ontologies
Ontology as Artifact

- A pragmatic view:
  An ontology is an artifact that must be constructed (i.e., designed artifact)

- Many artifacts have strong ontological flavour, such as
  - Glossary, taxonomy, dictionary, thesaurus
  - Knowledge bases, database schema
  - Conceptual maps, templates
Taxonomy := Segementation, classification and ordering of elements into a classification system according to the relationships among them; only one relationship: IS-A
Thesaurus

- Terminology for specific domain
- Graph with primitives, 2 additional relationships (similar, synonym)
- Originated from bibliography
Topics (nodes), relationships and occurrences (to documents)
Typically for navigation- and visualisation
Conceptual Graph: Definition

- Conceptual Graph (CG) is a knowledge representation system developed by John Sowa.
- CG is made of concept nodes and relation nodes
  - concepts nodes represent entities, attributes, states, and events
  - relations nodes show how the concepts are interconnected

![Diagram of Conceptual Graphs]

Cat: Michel on Mat
Cat: {*}@2 on Mat
Ontology

Person

Student

Reseacher

PhD Student

Topic

Semantics

F-Logic

Document

knows

writes

is_a

described_in

is_a

is_a

instance_of

Tel

Affiliation

Hamdy

403 210 5475

ISS

Ontology evolution

Rules

T described_in D

D is_about T

P writes D

D is_about T

P knows T

far@ucalgary.ca
A Real Example

Software measurement ontology

A measurement goal has its specific entity, attribute, perspective, purpose, with an environment description.
Imagine if
Every document on the Web contained metadata in a machine readable form as to who created it, What it is about, and some of the key document features.
Semantic Web Vision

- Bernes-Lee’s view of what we should be able to do with the help of data and engines on the Web.

  - Short-term:
    - Being able to reply on an email via telephone call
    - Meaningful browsing support

  - Mid-term:
    - Planning appointments with colleagues by integrating diaries
    - Context-aware applications
    - Giving restrictions for a trip and getting the schedule and the booking

  - Long-term:
    - Information exchange between different devices
    - Oral communication with the Semantic Web
    - Lawn assistant
Ontology vs. Semantic Web

- Ontology scope goes beyond Semantic Web

- Example of queries that cannot be handled by semantic web:
  - “find all students in Canada who are of East Asian origin and drive a Mercedes Benz”!
Ontology vs. Knowledge Base

Some basic questions

- Is the knowledge representation agreed upon?
- Is the used terminology clearly defined?
- Is the used terminology expressive enough?
- Is the knowledge easily reusable?
- Is it stable (for a given domain or task)?
- Is it the starting point for a group of new project?
Ontology vs. Knowledge Base

- Is ontology a knowledge base?

Rule based:

A \rightarrow B
B & C \rightarrow D
C & E \rightarrow F

Frame based:

concept

slot

method

Rule based

Frame based
Ontology vs. KB: Differences

- An ontology provides the basic structure around which a knowledge base can be built.
- An ontology provides a set of concepts and terms for describing some domain and captures the intrinsic conceptual structure of the domain and does not change with the change of environment.
- This is done via Ontological Analysis.
- Then, a knowledge base uses those terms to represent what is true about some real or hypothetical world.
Ontological Analysis (1)

- Ontological analysis clarifies the structure of knowledge.
- Given a domain, its ontology forms the heart of any system of knowledge representation for that domain.
- Without ontologies, or the conceptualizations that underlie knowledge, there cannot be a vocabulary for representing knowledge.
- Thus, the first step in devising an effective knowledge-representation system, and vocabulary, is to perform an effective ontological analysis of the field, or domain.
- Weak analyses lead to incoherent knowledge bases.
Ontological Analysis (2)

- A knowledge representation language, and consequently, a knowledge base, is built based on the ontological analysis

By

- associating terms with the concepts and relations in the ontology; and
- devising a syntax for encoding knowledge in terms of the concepts and relations
Ontological Analysis (3)

Analogical examples:

- Designing database systems
- Designing software systems
- Both require system (requirement) analysis and design before system implementation

So do need knowledge-based systems!
Example (1)

- A set of conceptualizations, and representative terms, for the *electronic devices* domain
- The resulting ontology would likely include domain-specific terms such as *circuit*, *component*, *transistors* and *diodes*; general terms such as *functions*, *causal processes*, and *modes*; and terms that describe behavior such as *cut-off*, *saturation current*, *voltage* and *power*
- But it would not contain assertions that a particular transistor had some defects, although a knowledge base might
Example (2)

- We can use the terms provided by the domain ontology to assert specific propositions about a domain or a situation in a domain.

- Example:

  circuit_3 has transistor_2 as a component;
  circuit_3 is an instance of the concept circuit;
  transistor_2 is an instance of the concept transistor;
  transistor_2 is currently in the cut-off mode;
  etc.
Ontology vs. OO (1)

- Building an ontology appears similar to building an Object Oriented software.
- Objects, their attributes, and their operations mirror aspects of the domain that are relevant to the application.
- OO systems representing a useful analysis of a domain can often be reused for a different application program.
Ontology vs. OO (2)

- So what are the differences?
  - Classes and objects in ontologies are about and must reflect the structure of the world; OO objects may not.
  - The representational repertoire of objects, relations, states, events, and processes does not say anything about which classes of these entities exist. The modeler of the domains makes these commitments.
  - Building blocks of OO are objects; building blocks of ontology are objects and relations.
Ontological Analysis : Benefits

- We can share this knowledge representation language with others who have similar needs for knowledge representation in that domain.
- Thereby eliminating the need for replicating the knowledge analysis process.
- Shared ontologies can thus form the basis for domain specific knowledge representation languages.
Constructing ontologies is an ongoing research.

Ontologies range in abstraction, from very general terms that form the foundation for knowledge representation in all domains, to terms that are restricted to specific knowledge domains.

- e.g., *space*, *time*, *parts*, and *subparts* are terms that apply to all domains; *malfunction* applies to engineering or biological domains; and *transistor* applies only to electronic circuits.
Constructing Ontology (2)

- Even in cases where a task might seem to be quite domain-specific, knowledge representation might call for an ontology that describes knowledge at higher levels of generality.
- e.g., solving problems in the domain of turbines might require knowledge expressed using domain-general terms such as flows and causality.
- Such general-level descriptive terms are called the upper ontology or top-level ontology.
Levels of Ontology (1)

- **Level 1**
  - Defining the concepts and the hierarchy
    - e.g., Yahoo Ontology

- **Level 2**
  - Unambiguously defining the meaning of concepts
  - Adding relationship definition
    - Case of many existing ontologies

- **Level 3**
  - Task ontology
As we move from an ontology top to lower taxonomic levels, commitments specific to domains and phenomena appear.

For modeling objects on earth, we can make certain commitments.

For example, animals, minerals, and plants are subcategories of objects; has-life(x) and contains-carbon(x) are object properties; and can-eat(x, y) is a possible relation between any two objects.

These commitments are specific to objects and phenomena in this domain.
Top-level Ontology Problems

- Different systems suggest different notations

Diagram:

- CYC:
  - Thing
  - Individual object
  - Intangible
  - Represented

- Wordnet:
  - Thing
  - Living
  - Nonliving

- GUM:
  - Um-Thing
  - Configuration
  - Element
  - Sequence

- Sowa’s:
  - Thing
  - Concrete
  - Process
  - Object
  - Abstract
General Agreements

- In spite of differences, partial agreement exists:
  - There are objects in the world
  - Objects have properties/attributes that can take values
  - Objects can exist in various relations with each other
  - Properties and relations can change over time
  - There are events that occur at different time instants
  - There are processes in which objects participate and that occur over time
  - The world and its objects can be in different states
  - Events can cause other events or states as effects
  - Objects can have parts
Domain Dependency

- There is no sharp division between domain-independent and domain-specific ontologies.
- E.g., the terms object, physical object, device, engine, and diesel engine all describe objects, but in an order of increasing domain specificity.
- Also, terms for relations between objects can span a range of specificity, such as connected, electrically-connected, and soldered-to.
How task-dependent (subjective) are ontologies?

Presumably, the kinds of things that actually exist do not depend on our goals. In that sense, ontologies are not task-dependent.

On the other hand, what aspects of reality are chosen for encoding in an ontology does depend on the task.

For example, in the domain of fruits, we would focus on particular aspects of reality if we were developing the ontology for the selection of pesticides; we would focus on other aspects if we were developing an ontology to help chefs select fruits for cooking.
Task Dependency (2)

- Practically, an ontology is unlikely to cover all possible potential uses
- In that sense, both an ontology for a domain and a knowledge base written using that ontology are likely to be more *task oriented*, i.e., appropriate for certain uses than others and unlikely to be sharable across widely divergent tasks
Interaction Problem

- Representing knowledge for the propose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem!
  
  [Bylander and Chandrasekaran, 1988] [Heijst]

- The interaction problem decreases at the knowledge level.
  It is possible to build at least understandable (if not reusable) knowledge bases by defining a formal ontology intended as a theory of the \textit{a priori} forms and nature of objects.
  [Guarino]
Potential Applications

- Semantic integration
- Ontology driven knowledge management
- Agent mediated knowledge management
- Data and knowledge translation
- Business process modeling
- Internet information processing: Search
Semantic Integration /1

- Different software systems may use individualized conceptualizations of a certain domain.
- To achieve ontology-based semantic integration, two software systems must find a way to share the semantics of the terms in their ontologies, this can be done in several possible directions:
  - **Using a single centralized global ontology:** a single centralized global ontology is defined for the application domain and all agents or computer programs in communication use terms from this ontology (e.g., CYC project, Experience Factory).
Semantic Integration /2

- **Merging source ontologies into a unified ontology**: ontologies defined on a common domain by different applications have lots of overlap, therefore merging the source ontologies into one unified ontology before agent interactions is a way to fulfill semantic integration.

- **Mappings (or matches) between two ontologies**: instead of trying to merge two source ontologies, finding a set of mapping rules between them.

- **Runtime ontology resolution**: Software agents may be heterogeneous and it is impractical to restrict all agents to use a single ontology or to have ontology merging, matching, and translation services available prior to the deployment of the agent system. A better way is to resolve semantic differences when they arise during run-time interaction.
1) Single ontology approaches

All information sources have the same view of a domain; impractical; hard to maintain.
2) Hybrid ontology approaches

Ontologies are built from a global shared vocabulary.
3) Multiple ontologies approaches

Local ontology

Ontologies differ in their vocabulary. No need for ontology commitment.

DB1

DB2

DB3

Practical; requires sophisticated sharing mechanisms.
Ontology Based Knowledge Management Tool - Implementation
Architecture of Layered KM

Peer1

Semantic

Syntactic

Lexical

Encoding

Peer2

Semantic

Syntactic

Lexical

Encoding

EXCHANGE CHANNELS

Query Phrase

<Semantic: XXX>Query Words</Semantic: XXX>

<Syntactic: YYY><Semantic: XXX> Query Phrase </Semantic: XXX></Syntactic: YYY>

Lexical: ZZZ><Syntactic: YYY><Semantic: XXX> Query Words </Semantic: XXX></Syntactic: YYY></Lexical>

<Encoding: English Unicode><Lexical: ZZZ>

<Syntactic: YYY><Semantic: XXX> Query Phrase </Semantic: XXX></Syntactic: YYY></Lexical></Encoding>
Ontology Learning & Semantic Search

1. Send query
   - Encode query phrase
   - Number of concepts contained in query phrase

2. Semantic search
   - Annotate documents
   - Disambiguity
   - Return documents

3. Send sample documents back
   - Document specificity

4. Concept learning
   - Identify newly learned concept
   - Cumulative concepts
Ontology Driven KM

User
Knowledge Engineer

Sharing
Searching
Visualizing
Browsing

Ontology Editing

Ontology Middleware and Reasoning

Ontology Repository

Annotations
Documents

Semi-Structured Data
Unstructured Data

Ontology
Middleware
and
Reasoning
Agent Mediated KM

- Agent Mediated Knowledge Management (AMKM) has been an active research area in the software agent research community in recent years. It uses agent concepts to analyze and model organizations and their knowledge needs and to provide a reusable architecture to build KM systems.

- AMKM is different from peer-to-peer KM in the sense that
  1. agents can learn and adapt themselves to changes in environment; and
  2. agents can get involved in complex interactions.

- These two points come from the fact that agents are autonomous and social entities.
Test Configuration

Remote Peers

Network Hub

Local Peer

far@ucalgary.ca
Another peer comes online

******** Discovery Event ********
Local pipe: Pipename:JxtaPeer-1866920075
Adding NEW Available Peers: Pipename:JxtaPeer556926133

******** Discovery Event ********
Local pipe: Pipename:JxtaPeer-1866920075
Available peer: Pipename:JxtaPeer556926133

******** Discovery Event ********
Local pipe: Pipename:JxtaPeer-1866920075
Available peer: Pipename:JxtaPeer556926133

******** Discovery Event ********
Local pipe: Pipename:JxtaPeer-1866920075
Available peer: Pipename:JxtaPeer556926133
Query local index & send results to the requesting peer
Query Process (1)
Enter Search String
Query Process (2)
Connect, Send Query, Receive

Code snippet:
```
1
[[ 1 ]]
Attempting to establish a connection to Pipename
Connected to peer: Pipename: JxtaPeer: 55692:5133
Query.connectOK(): sending query
Sending :test
Sending :ENDMESSAGE
Query.sendDone()
2
3
Number of local results for 'contents: test'? 7
4
5
Query.receive[] : true
----- Message Name : query_result
- Message Contents: [801610944]
```
Cycle completed
Another Implementation
Design of Prototype System

- Control
- Concept Learner
- Concept Manager
- Register Handler
- GUI
- User
- Doc Repository
- Concept Repository
- Yellow Pages
Realization - UIMA

- Unstructured Information Management (UIM) applications are software systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user.

- UIMA (Unstructured Information Management Architecture) is a framework and SDK for developing such applications such as semantic search.
UIMA Architecture

Any UIMA-Compliant Readers ➔ Connect, Read & Segment Sources ➔ Analyze Content Assign Task-Relevant Semantics ➔ Index or Process Results ➔ End-User Application Interfaces

UI MA: Pluggable Framework, User-defined Workflows
CAS: Common UIMA Data Representation & Interchange
Aligned with OMG & W3C standards (i.e., XMI, SOAP)

Query Interface(s) ➔ Relevant Knowledge ➔ Query Services ➔ Relational Database

Text, Chat, Email, Speech, Video

Original figure from IBM
Semantic Search Using UIMA

Collection Processing Engine

Collection of Text Docs

File Reader

Analysis Aggregator

Analysis Engines

Semantic Search Indexer

XCas Writer

CAS Consumers

Local File System

Semantic Search Index

Original figure from IBM
Implementation of Annotator

- Development Platform: UIMA
- Annotator Role is developed upon it
- Main Concepts of UIMA:
  - AE (Analysis Engine): Basic Building Block of UIMA
  - Type System: Mechanism to describe entity system
Workflow of Document Annotator

1. Create annotation type system
   - Type System:
     - Concept Hierarchy:
       - Concept 1 \((f_1, f_2, \ldots)\)
       - ...

2. Build Annotation Engine
   - AE:
     - Pre-defined processing logics

3. Annotate documents

4. Reply query with annotated documents

\((\text{concept1}, \ldots, \text{keyword1}, \text{keyword2}, \ldots)\)
Our Solution

Semantic Query
"<AgentMethodology>Prometheus</AgentMethodology>"
Study Case
First Round of Search

Keywords: prometheus

Concepts: 

Candidate Documents

03 Prometheus A Methodology for Developing Intelligent Agents-a 11 Greek Mythology
Second Round of Search

![Image of a search application interface with keywords "prometheus" and concepts "agent". A candidate document is highlighted: "03 Prometheus: A Methodology for Developing Intelligent Agents."
Ontology Refinement
Ontology Life-cycle

- Creating
- Evolving
- Maintaining
- Deploying
- Populating
- Validating

YOU
Ontology: Tools

- Ontology editing tools are needed.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Capabilities</th>
<th>Vendor</th>
<th>Standards Compliance and General Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protege-2000</td>
<td>Create concept hierarchies, create instances, view in several formats. Single user tool.</td>
<td>Stanford KSL</td>
<td>Open Source (Mozilla); plug-in architecture. Supports RDF, DAML+OIL, OWL.</td>
</tr>
<tr>
<td>OilEd</td>
<td>Create concept hierarchies, create instances, analyze semantic consistency (according to DL). Single user tool.</td>
<td>U of Manchester</td>
<td>RDF, DAML+OIL support. From the creators of OIL. Free download, integrates with DL reasoner.</td>
</tr>
</tbody>
</table>
## Ontology: Tools /2

<table>
<thead>
<tr>
<th>Tool</th>
<th>Capabilities</th>
<th>Vendor</th>
<th>Standards Compliance and General Comments</th>
</tr>
</thead>
</table>
### Ontology: Tools 1/3

<table>
<thead>
<tr>
<th>Tool</th>
<th>Capabilities</th>
<th>Vendor</th>
<th>Standards Compliance and General Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Factory Workbench</td>
<td>Collaborative authoring environment. Originally designed for very large medical ontologies. Has a Java beans API and optional Application Generators for semantic indexing, automatic coding, and information extraction.</td>
<td>Language and Computing</td>
<td>Supports RDF(S); DAML+OIL/OWL. Some support for population and maintenance.</td>
</tr>
<tr>
<td>K-Infinity</td>
<td>Collaborative authoring environment.</td>
<td>Intelligent Views</td>
<td>Modularized tools supporting all stages of lifecycle. Supports RDF and Topic Maps.</td>
</tr>
</tbody>
</table>
Some Projects

- **CLEPE** - Task ontology description language  
  (Osaka University - Mizoguchi Laboratory)
- **WorldNet**  
  (Princeton University - Cognitive Science Laboratory)
- **SENSUS** - Machine translation and text summarization
- **Jena 2 Ontology API**  
  ([http://jena.sourceforge.net/ontology/](http://jena.sourceforge.net/ontology/))
- **OWL Web Ontology Language**  
  ([http://www.w3.org/TR/owl-ref/](http://www.w3.org/TR/owl-ref/))
Some Tools & News

- The Ontolingua Server: A Tool for Collaborative Ontology Construction
- Stanford KSL Network Services

Check regularly:
http://www.cs.umbc.edu/kse/ontology/
http://www.w3.org/
Research Topics

- How to learn and/or override concepts and properties?
- How to “mine” concepts and properties?
- How to “build” ontologies, check and debug them?
Conclusion

- Distributed multiagent system development relies on knowledge bases, but those knowledge bases are built with little sharing or reuse.
- In the future, intelligent systems developers will have libraries of ontologies. Rather than building from scratch, they will assemble knowledge bases from components drawn from the libraries.
- This should greatly decrease development time while improving the robustness and reliability of the resulting knowledge bases.
want a break?
KQML as an Agent Communication Language

Tim Finin     Yannis Labrou     James Mayfield
Computer Science Department
University of Maryland Baltimore County
Baltimore MD USA
Abstract

- This work describes the design of and experimentation with the Knowledge Query and Manipulation Language (KQML), a language and protocol for exchanging information and knowledge.

- This work is part of the ARPA Knowledge Sharing Effort which is aimed at developing techniques and methodology for building large-scale knowledge bases which are sharable and reusable.

- KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents.

- KQML provides a basic architecture for knowledge sharing through a special class of agent called communication facilitators which coordinate the interactions of other agents.
The computational environment in today’s business systems is characterized by being highly distributed, heterogeneous, dynamic, and comprising a large number of autonomous nodes. An information system operating in such environment must handle several emerging problems:

- The predominant architecture on the Internet, the client-server model, is too restrictive.
- Several forms of heterogeneity need to be handled, e.g., different platforms, different data formats, the capabilities of different information services, and the implementation technologies employed.
- Many software technologies such as applied natural language processing, knowledge based reasoning, advanced information retrieval, speech processing, etc., have matured to the point of being ready to participate in and contribute to the bigger picture of the information systems.
A community of intelligent agents can address each of the problems mentioned.

When we describe these agents as intelligent, we refer to their ability to: communicate with each other using an expressive communication language; work together cooperatively to accomplish complex goals; act on their own initiative; and use local information and knowledge to manage local resources and handle requests from peer agents.

Knowledge Query and Manipulation Language (KQML) is a language that is designed to support interactions among intelligent software agents. It was developed by the ARPA supported Knowledge Sharing Effort.
Ability to Interact

- It is doubtful that any discussion about agents will result in a consensus on the definition of an agent or of agency. From personal assistants and smart interfaces to autonomous, intelligent entities and to information retrieval systems, anything might qualify as an agent.

- But, despite these different viewpoints, everyone would agree that the ability of interaction and interoperation is a necessity.
A Message Contents

Message contents:

- I want a jaguar!
- I love turkey!
- 市川市
Interaction: The Problem

- How might meaningful, constructive and intelligent interaction among software agents be dealt with?

- The problem for humans requires more than the knowledge of a common language, e.g., English; it also requires a common (mutual) understanding of the terms used in a given context.

- **Example:** A physicist’s understanding of velocity is not the same as that of a F-1 driver, and if the two want to talk about “fast cars” they have to speak a “common language.”
Knowledgeable Agent

Components related to sharing and communication

Other components

Representation

Ontologies

Knowledge Base

Communication

Transport Protocol

Communication Language

Interaction Protocol

Planning

Modelling

Non-Knowledge

Reasoning
1. Mutual Understanding

- **Mutual understanding** of what is represented may be divided into two subproblems:
  - **Translation** from one representation language to another (or from one family of representation languages to another).
  - **Sharing** the semantic content of the represented knowledge among different applications.

- Translation alone is not sufficient because each knowledge base holds *implicit assumptions* about the meaning of what is represented.

- If two applications are to understand each other’s knowledge, such assumptions must also be shared, i.e., the semantic content of the various tokens must be preserved.
2. Communication

Communication may be divided into three subproblems:

- Interaction protocol
- Communication language
- Transport protocol
2. Communication (contd.)

- **Interaction protocol** refers to the high level strategy pursued by the software agent that governs its interaction with other agents. Such a protocol can range from negotiation schemes and game theory protocols, etc.
  - **Example:** “every time you do not know something, find someone who knows and ask.”

- **Communication language** is the medium through which the attitudes regarding the content of the exchange are communicated. It is the communication language that suggests whether the content of the communication is an assertion, a request or some form of query.

- **Transport protocol** is the actual transport mechanism used for the communication, such as TCP, SMTP, HTTP, etc.
To address many of the difficulties of communication among intelligent agents, we must give them a *common language*. In linguistic terms, this means that they must share a *common syntax, semantics* and *pragmatics*.

**Syntax:** Currently, there is no universally accepted language in which to represent information and queries.
2. Communication (contd.)

- **Semantics:** Assuming the use of a common language, it is still necessary for communicating agents to share a framework of knowledge in order to interpret the messages they exchange, i.e., shared semantics or shared ontology.

- **Pragmatics:** Pragmatics among computer processes includes
  - 1) knowing with whom to talk and how to find them; and
  - 2) knowing how to initiate and maintain an exchange.
How to Interact?

- What is needed to interact and interoperate?
  - A representation language
    - KIF
  - A communication language
    - KQML

Sharing content requires more than a representation language and a communication language.

- Sharing the content
  - Ontology
Representation Language: KIF

- Knowledge Interchange Format (KIF) is a language for the syntactic aspects of representations for knowledge sharing.
- KIF can be used to support translation from one content language to another, or as a common content language between two agents which use different native representation languages.
- There were two intentions behind the development of KIF:
  - Creation of a lingua franca for the development of intelligent applications, with an emphasis on interoperation.
  - Creation of a common interchange format so that with the use of “translators” one could translate from language A to KIF and from KIF to language B instead of translating from A to B.
KQML was conceived as both a message format and a message-handling protocol to support run-time knowledge sharing among agents.

The basic features of KQML are:

- KQML messages are opaque to the content they carry. KQML messages do not merely communicate sentences in some language, but they rather communicate an attitude about the content (assertion, request, query).

- The language’s primitives are called performatives. Performatives define the permissible actions (operations) that agents may attempt in communicating with each other.

- An environment of KQML speaking agents may be enriched with special agents, called facilitators, that provide such functions as: association of physical addresses with symbolic names; registration of databases and/or services offered and sought by agents; and communication services (forwarding, brokering etc.). To use a metaphor, facilitators act as efficient secretaries for the agents in their domain.
Sharing Contents: Ontology

- Sharing content requires more than a representation language (i.e., KIF) and a communication language (i.e., KQML).
- Every knowledge-based system relies on some conceptualization of the world (objects, qualities, distinctions and relationships that matter for performing some task) that is embodied in concepts, etc., in a formal representation scheme.
- **Common ontologies** address the problem of what must be held in common among communicating agents.
- A common ontology refers to an explicit specification of the ontological commitments of a set of programs. An agent commits to an ontology if its observable actions are consistent with the definitions in the ontology.
Knowledge Interchange Format (KIF)
KIF is a version of first order predicate calculus with extensions to support non-monotonic reasoning and definitions.

The language description includes both a specification for its syntax and its semantics.

Example: expression of simple data in KIF

The sentences shown below encode 3-tuples in a personnel database (arguments stand for employee ID number, department assignment and salary, respectively):

- (salary 015-46-3946 ENEL 72,000)
- (salary 026-40-9152 CS 36,000)
Example: More complicated expression

The following sentence states that one chip is larger than another:

\[(? (* \text{(width chip1)} \text{(length chip1)}) (* \text{(width chip2)} \text{(length chip2)}))\]
KIF /3

- KIF includes logical operators to encode logical information, such as negation, disjunction, rules, quantified formulas and other operators for encoding knowledge.
- KIF can also be used to describe procedures, i.e., to programs or scripts for agents to follow. Given the syntax of KIF, such programs resemble Lisp.
- The semantics of the KIF core (KIF without rules and definitions) is similar to that of first order logic.
Knowledge Query Manipulation Language (KQML)
KQML

- KQML is a language and a set of protocols that support computer programs in identifying, connecting with and exchanging information with other programs.

- KQML is divided into three layers:
  - Content layer
  - Communication layer
  - Message layer
1. Content Layer

- The content layer bears the actual content of the message, in the program's own representation language.
- KQML can carry any representation language, including languages expressed as ASCII strings and those expressed using a binary notation.
- Every KQML implementation ignores the content portion of the message, except to determine where it ends.
2. Communication Layer

- The communication layer encodes a set of features to the message which describe the lower level communication parameters, such as the identity of the sender and recipient, and a unique identifier associated with the communication.
3. Message Layer

- Message layer is used to encode a message that one application would like to transmit to another. It determines the kinds of interactions one can have with a KQML-speaking agent.

- The prime function of the message layer is to identify the protocol to be used to deliver the message and to supply a performative which the sender attaches to the content (such as that it is an assertion, a query, a command, or any of a set of known performatives).
3. Message Layer (contd.)

- Since the content is opaque to KQML, this layer also includes optional features which describe the content language, the ontology it assumes, and some type of description of the content, such as a descriptor naming a topic within the ontology.

- These features make it possible for KQML implementations to analyze, route and properly deliver messages even though their content is inaccessible.
KQML: Syntax

- Syntax of KQML is based on a balanced parenthesis list (similar to Common Lisp). The initial element of the list is the performative (= permissible actions or operations); the remaining elements are the performative’s arguments as keyword/value pairs.

- **Example:**

```
(ask-one
    :sender joe
    :content (PRICE IBM ?price)
    :receiver stock-server
    :reply-with ibm-stock
    :language LPROLOG
    :ontology NYSE-TICKS)
```
KQML: Syntax (contd.)

- Other performatives:
  - `tell`
  - `ask-all` (query for multiple reply)
  - `stream-all` (convert a set of answers to a set of replies)
  - `standby` (expects a KQML language content and it requests that the agent receiving the request take the stream of messages and hold them and release them one at a time)
  - `subscribe` (requests all future changes to the answer to the query)
  - etc.
KQML: Syntax (contd.)

- **KQML reserved performatives:**
  - **Basic query:** evaluate, ask-if, ask-about, ask-one, ask-all
  - **Multi-response (query):** stream-about, stream-all
  - **Response:** reply, sorry
  - **Generic informational:** tell, achieve, cancel, untell, unachieve
  - **Generator:** standby, ready, next, rest, discard, generator
  - **Capability-definition:** advertise, subscribe, monitor, import, export
  - **Networking:** register, unregister, forward, broadcast, route
Communication Protocols

- There are a variety of interprocess information exchange protocols in KQML. In the simplest, one agent acts as a client and sends a query to another agent acting as a server and then waits for a reply, as is shown between agents A and B. The server’s reply might consist of a single answer or a collection or set of answers.
Another case, shown between agents A and C, the server’s reply is not the complete answer but a handle which allows agent A to ask for the components of the reply, one at a time. A common example of this exchange occurs when a client queries a relational database which produces a sequence of instantiations in response.
Communication Protocols

Another case, shown between agents A and D, client subscribes to a server’s output and an indefinite number of asynchronous replies arrive at irregular intervals. The client does not know when each reply message will be arriving and may be busy performing some other task when they do.
Facilitators

- A facilitator is an agent that performs various useful communication services.

- **Example:** maintaining a registry of service names, forwarding messages to named services, routing messages based on content, providing “matchmaking” between information providers and clients, and providing mediation and translation services.

- A main function of facilitator agents is to help other agents find appropriate clients and servers.
Example: Facilitators

- Consider a case where an agent A would like to know the truth of a sentence X, and agent B may have X in its knowledge-base, and a facilitator agent F is available.

- If A is aware that it is appropriate to send a query about X to B, then it can use a simple point to point protocol and send the query directly to B.
Example: Facilitators (contd.)

- What if A is not aware of what agents are available, or who may have X in is knowledge-bases, or how to contact those of whom it is aware?

- A possible scenario is that A uses the subscribe performative to request that facilitator F monitor for the truth of X. If B subsequently informs F that it believes X to be true, then F can in turn inform A.
A asks F to find an agent that can process an ask(X) performative. B independently informs F that it is willing to accept performatives matching ask(X). Once F has both of these messages, it sends B the query, gets a response and forwards it to A.
Example: Facilitators (contd.)

- A uses a slightly different performative to inform F of its interest in knowing the truth of X. The recruit performative asks the recipient to find an agent that is willing to receive and process an embedded performative. That agent’s response is then to be directly sent to the initiating agent.
A asks F to “recommend” an agent to whom it would be appropriate to send the performative ask(X). Once F learns that B is willing to accept ask(X) performatives, it replies to A with the name of agent B. A is then free to initiate a dialogue with B to answer this and similar queries.
Conclusions

- The agent communication and representation languages (KQML and KIF) and common ontologies approach together can address the fundamental interaction and interoperation problems.
- They together include:
  - Translation between representations.
  - Sharing the semantic (and often pragmatic) content of the knowledge that is represented.
  - Communicating attitudes about the shared knowledge.
Critics

- The formalism is based on common or shared ontology assumption, that may not be valid.
- Learning curve of the languages: the syntax is not compatible with the other common programming languages.
- Several newer variations are proposed.
- Several features have been later included in the FIPA standards.