Overview and objectives

- The agent paradigm -- what is it and what’s important about it (IMHO)
- The centrality of communication for agents
- Some historical context and future directions
- Basic concepts of agent communication and their realization in FIPA
- Some examples of applications done at UMBC
What is a software agent?

- No consensus yet, but several key concepts are important to this emerging paradigm.
- A software agent is a program that assists people and acts on their behalf. People can delegate tasks and work to agents. An agent:
  - is an autonomous, goal-directed process
  - is situated in, is aware of, reacts to and adapts to its environment
  - cooperates with other agents (software and/or human) to accomplish its tasks

Agent Characteristic: Mobility?

A mobile agent is an executing program that migrates from machine to machine in a heterogeneous network under its own control.

- Examples: programs in Telescript, Agent-Tcl, Voyager, etc. and, to a limited degree, Java Applets.
- Note -- this definition implies some agent attributes, e.g. autonomy, persistence, ...
- Mobile agents offer some very interesting advantages as well as some disadvantages.
- This is an important technology for distributed systems but is largely orthogonal to other “agent” issues.
Agent Characteristic: Intelligence?

Q: What makes an agent an “intelligent agent”?  

A: The size of the price tag.  
More seriously…  
– The paradigm covers agents of varying degrees of intelligence  
– Intelligent agents will tend to  
  • know and apply more sophisticated domain knowledge  
  • recognizing underlying goals and intentions  
  • react to unexpected situations in a robust manner  
  • better NLP skills  
  • etc.  

Much of what we will be saying applies to agents of little or no intelligence.

Some key ideas

• Software agents offer a new paradigm for very large scale distributed heterogeneous applications.  
• The paradigm focuses on the interactions of autonomous, cooperating processes which can adapt to humans and other agents.  
• Mobility is an orthogonal characteristic which many, but not all, consider important.  
• Intelligence is always a desirable characteristic but is not strictly required by the paradigm.  
• The paradigm is still forming.
Agent Communication

- Cooperation requires communication.
- Agent-to-agent communication is key to realizing the potential of the agent paradigm, just as the development of human language was key to the development of human intelligence and societies.
- Agents use an Agent Communication Language or ACL to communicate information and knowledge.
  - Genesereth (CACM, 1992) defined a software agent as any system which uses an ACL to exchange information.
- Understanding a “common language” means:
  - understanding its vocabulary, i.e., the meaning of its tokens
  - knowing how to effectively use the vocabulary to perform tasks, achieve goals, effect one’s environment, etc.
- For ACLs we’re primarily concerned with the vocabulary

Some ACLs

- Is CORBA an ACL?
- Knowledge sharing approach
  - KQML, KIF, Ontologies
- FIPA
- Ad hoc languages
  - e.g., SGI’s OAA

Intentional Sharing
- Shared beliefs, plans, goals, and intentions

Knowledge Sharing
- Shared facts, rules, constraints, procedures and knowledge

Object Sharing
- Shared objects, procedure calls and data structures

e.g., CORBA, RPC, RMI

e.g., KQML, FIPA, KIF, Aglets

© Copyright 1998-99 Tim Finin
Agent Communication, at the theoretical level

- ACL have message types that are usually modeled after speech acts, which are understood in terms of an intentional-level description of an agent.
- An intentional description makes references to beliefs, desires, intentions (BDI) and other mental states.
- BDI frameworks have the power to describe an agents’ behavior, including communicative behavior.
- Describing behavior at this level is an important contribution of the agent-based approach.

Agents and agencies

- Groups of agents can form a team to cooperate and act as one super-agent.
- Opening up an agent we may find it useful to describe its internal architecture as a collection of sub-agents.
- What’s going on here? Is it agents all the way down?
- My take -- a group of agents which can be modeled as having collective or mutual “mental states” (e.g., beliefs, desires, intentions) and can take collective actions can be usefully described as an agent.
Dividing up the problem

Historical Note: Knowledge Sharing Effort

- Initiated by DARPA circa 1990
- Sponsored by DARPA, NSF, AFOSR, etc.
- Participation by dozens of researchers in academia and industry.
- Developing techniques, methodologies and software tools for knowledge sharing and knowledge reuse.
- Sharing and reuse can occur at design, implementation or execution time.
Knowledge Sharing Effort

- Knowledge sharing requires a communication which requires a common language
- We can divide a language into syntax, semantics, and pragmatics
- Some existing components that can be used independently or together:
  - KIF - knowledge interchange format (syntax)
  - Ontolingua - a language for defining sharable ontologies (semantics)
  - KQML - a high-level interaction language (pragmatics)

Knowledge Sharing Effort => FIPA

- Knowledge sharing requires a communication which requires a common language
- We can divide a language into syntax, semantics, and pragmatics
- Some existing components that can be used independently or together:
  - KIF - knowledge interchange format (syntax) => SL
  - Ontolingua - a language for defining sharable ontologies (semantics) => ACL
  - KQML - a high-level interaction language (pragmatics)
Knowledge Interchange Format

- KIF ~ First order logic set theory
- An interlingua for encoded declarative knowledge
  - Takes translation among $n$ systems from $O(n^2)$ to $O(n)$
- Common language for reusable knowledge
  - Implementation independent semantics
  - Highly expressive - can represent knowledge in typical application KBs.
  - Translatable - into and out of typical application languages
  - Human readable - good for publishing reference models and ontologies.
- Current specification at http://logic.stanford.edu/
- FIPA’s SL $\approx$ KIF + modal operators - default reasoning

Common Semantics

Shared Ontologies and Ontolingua

- **Ontology**: A common vocabulary and agreed upon meanings to describe a subject domain.
- Ontolingua is a language for building, publishing, and sharing ontologies.
  - A web-based interface to a browser/editor server.
  - Ontologies can be automatically translated into other content languages, including KIF, SL, LOOM, Prolog, etc.
  - The language includes primitives for combining ontologies.
Common Pragmatics
Knowledge Query and Manipulation Language

- KQML is a high-level, message-oriented, communication language and protocol for information exchange independent of content syntax and ontology.
- KQML is also independent of
  - transport mechanism, e.g., tcp/ip, email, corba, IIOP, ...
  - High level protocols, e.g., Contract Net, Auctions, …
- Each KQML message represents a single *speech act* (e.g., ask, tell, achieve, …) with an associated *semantics* and *protocol*.
- KQML includes primitive message types of particular interest to building interesting agent architectures (e.g., for mediators, sharing intentions, etc.)

KQML
Knowledge Query and Manipulation Language

- KQML is a high-level, message-oriented, communication language and protocol for information exchange independent of content syntax and ontology.
- KQML is independent of
  - the transport mechanism (e.g., tcp/ip, email, corba objects, IIOP, etc.)
  - Independent of content language (e.g., KIF, SQL, STEP, Prolog, etc.)
  - Independent of the ontology assumed by the content.
- KQML includes primitive message types of particular interest to building interesting agent architectures (e.g., for mediators, sharing intentions, etc.)
A KQML Message

Represents a single speech act or performative
ask, tell, reply, subscribe, achieve, monitor, ...

with an associated semantics and protocol

\[
\text{tell}(i, B_i \phi) = \text{fp}[B_i B_i \phi \land \neg B_i (B_i \phi \lor U_i B_i \phi)] \land \text{re}[B_i B_i \phi] \ldots
\]

and a list of attribute/value pairs

:sender        bhkAgent
:receiver      fininBot
:in-reply-to   id7.24.97.45391
:ontology      ecbk12
:language      Prolog
:content       “price(ISBN3294,24.95)”

KQML Reserved Parameter Keywords

:sender the actual sender of the performative
:receiver the actual receiver of the performative
:from the origin of the performative in :content when forward is used
:to the final destination of the performative in :content when forward is used
:in-reply-to the expected label in a response to a previous message (same as the :reply-with value of the previous message)
:reply-with the expected label in a response to the current message
:language the name of the representation language of the :content
:ontology the name of the ontology (e.g., set of term definitions) assumed in the :content parameter
:content the information about which the performative expresses an attitude
Facilitation Services

Facilitators are a class of agents who

• traffic in meta-knowledge about other agents.

• provide communication services such as:
  – message forwarding and broadcasting
  – resource discovery
  – matchmaking
  – content-based routing
  – meta-knowledge queries

• Performatives of special interest to facilitators are
  – advertise, broker, recruit, recommend, forward, broadcast, etc.

• Brokers are generally considered to focus on matchmaking

• Facilitators can be intelligent or not
  – Intelligent facilitators use domain knowledge in matching services needs and offers.
If KQML is so great, why do we need the FIPA ACL?

- There are two KQML specification documents and many dialects and “extended” versions of KQML plus many important concepts not yet addressed (e.g., security).
- The FIPA ACL has been specified along with other critical aspects of agent systems (e.g., agent management).
- The FIPA ACL has the support of a formal standardization process and organization.
- FIPA provides us with an opportunity to revisit and improve on many of the design decisions made in KQML
  - e.g., primitive CAs are defined in such a way as to allow them to be composed, creating new CAs

Major Features of FIPA ACL compared to KQML

- Management and facilitation primitives (register, broker, recruit, etc.) are not part of the ACL
- Primitives can be defined compositionally from “core” primitives
- Use of a powerful language to define agents’ states (Semantic Language, or SL)
- Semantics based on mental attitudes (belief, intention, etc.)
- The meaning of primitives is given in terms of Feasibility Preconditions (FPs) and Rational Effect (RE)
Ontologies

Overview

• What is an ontology?
• Tools for building, using and maintaining ontologies
• Existing ontologies of general interest
• FIPA's view on agents and ontologies
Common Semantics
Shared Ontologies and Ontolingua

**Ontology**: A common vocabulary and agreed upon meanings to describe a subject domain.

Ontology: A common vocabulary and agreed upon meanings to describe a subject domain.

- Vocabulary specification
- Domain theory
- Conceptual schema (for a data base)
- Class-subclass taxonomy
- Object schema

This is not a profoundly new idea …

Conceptual Schemas

A conceptual schema specifies the intended meaning of concepts used in a data base

**Data Base**: 

<table>
<thead>
<tr>
<th>stockNo</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>139</td>
<td>74.50</td>
</tr>
<tr>
<td>140</td>
<td>77.60</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

**Data Base Schema**: Table: price

*stockNo*: integer; *cost*: float

**Conceptual Schema**: 

price(x, y) =>

\[ \exists (x', y') [\text{auto_part}(x') \land \text{part_no}(x') = x \land \text{retail_price}(x', y', \text{Value-Inc}) \land \text{magnitude}(y', \text{US_digits}) = y] \]
Implicit vs. Explicit Ontologies

• Systems which communicate and work together must share an ontology.
• The shared ontology can be implicit or explicit.
• Implicit ontology are typically represented only by procedures
• Explicit ontologies are (ideally) given a declarative representation in a well defined knowledge representation language.

Conceptualizations, Vocabularies and Axiomitization

• Three important aspects to explicit ontologies
  – Conceptualization involves the underlying model of the domain in terms of objects, attributes and relations.
  – Vocabulary involves assigning symbols or terms to refer to those objects, attributes and relations.
  – Axiomitization involves encoding rules and constraints which capture significant aspects of the domain model.
• Two ontologies may
  – be based on different conceptualizations
  – be based on the same conceptualization but use different vocabularies
  – differ in how much they attempt to axiomitize the ontologies
Simple examples

Ontologies vs. KBs

Ontologies are distinguished from KBs not by their form, but by the role they play in representing knowledge:

– Consensus models for a domain
– Emphasis on properties that hold in all situations
– Emphasis on classes rather than instances
– Intended to support multiple tasks and methods
– Don’t change during problem solving and are suited for “compiling” into tools
– Need to satisfy a community of use
  • Emphasis on collaborative development
  • Emphasis on translation to multiple logical formalisms
– Useful for education
Ontology Library and Editing Tools

Ontolingua is a language for building, publishing, and sharing ontologies.

- A web-based interface to a browser/editor server at http://ontolingua.stanford.edu/ and mirror sites.
- Ontologies can be translated into a number of content languages, including KIF, LOOM, Prolog, CLIPS, etc.

Ontolingua - Language

- Ontolingua allows full KIF
  - 1st order logic with relation constants in domain of discourse
  - Extremely expressive
  - Too much for most users
  - Too much for most systems!
- Ontolingua provides an object-oriented projection
- Statements within the o-o sublanguage easy to make
  - But any statement is allowed
- Ontolingua separates representation from presentation

### Class Characterization-Oscilloscope

- Defined in Ontolingua
- Source path: /ontolingua/ontolingua
- Ontolingua allows for object-oriented projection
- Statements within the o-o sublanguage easy to make
- But any statement is allowed
- Ontolingua separates representation from presentation
Ontolingua - Library

- Library of modules supports reuse
- Authors assemble a new ontology
  - Assembly defines a general graph
  - Cycles are allowed (sports and medicine)
- Authors may augment definitions
  - But you can never say less!
  - Different authors may make incompatible extensions

 Ontolingua - Usage

- Ontolingua is (one of) the most widely used knowledge development environments
  - Available since 1/94 at http://ontolingua.stanford.edu
  - Over 4500 total users, 1200 current users, 300 active users
  - Over 4,200,000 user commands executed
  - Recently averaging over 7000 commands per day
  - Over 800 ontologies stored on the KSL server
  - Mirror sites in Spain, Netherlands, UMBC, and corporate sites
- Applications include
  - Enterprise modeling, electronic commerce, engineering, ribosomal structure modeling, workflow modeling, molecular biology, cross-disciplinary design and simulation, drug interactions, medical vocabularies, software design reuse, standards development
Open Knowledge Base Connectivity

- OKBC is another software tool for building and accessing ontologies
  - OKBC is to KBs what ODBC is to Databases
  - A standard API to Knowledge Representation Systems (KRS)
- OKBC
  - Specifies a protocol for KRS interoperation
  - Supports a client-server model for interaction
  - Provides an object-oriented view of a KRS
  - Supports wide variation in underlying KRS
- Adopted as KRS interoperation protocol within DARPA High Performance Knowledge Base (HPKB) program
- OKBC drivers available for Loom, Ontolingua, Ocelot, ATP, JavaKB, TupleKB, ...
- OKBC applications include GKB (SRI), Jot (KSL), Ontolingua (KSL), Riboweb (SMI), Protégé (SMI), Hike (SAIC), ...
- http://ai.sri.com/~okbc/

Big Ontologies

- There are several large, general ontologies that are freely available.
- Some examples are:
  - Cyc - Original general purpose ontology
  - WordNet - a large, on-line lexical reference system
  - World Fact Book -- 5Meg of KIF sentences!
  - UMLS - NLM’s Unified Medical Language System
- See http://www.cs.utexas.edu/users/mfkb/related.html for more
- We anticipate the development of ontologies to support ecommerce
  - see www.ontology.org
  - probably in XML
Ontologies in the FIPA Reference Model

Ontology Agents provide ontology related services (e.g., translation) to FIPA agents

Ontology Agent services

- Help a FIPA agent in selecting a shared (sub) ontology for communication,
- Create and update an ontology, or only some terms of an ontology.
- Translate expressions between different ontologies (different names with same meanings),
- Respond to query for relationships between terms or between ontologies,
- Discovery of public ontologies in order to access them.
The FIPA knowledge model

- FIPA specifies **fipa-meta-ontology** as the ontology used to talk about ontologies.
- This is largely based on the OKBC model developed by Stanford and SRI
  - See http://ai.sri.com/~okbc
- As its name suggests:
  - OKBC is to KBs what ODBC is to Databases
  - A standard API to Knowledge Representation Systems (KRS)
- Generic OKBC clients (e.g., browsers, editors) exist as well as OKBC drivers for a number of knowledge representation systems.

Relationships between ontologies

- **fipa-meta-ontology** also includes:
  
  (ontol-relationship ?O1 ?O2 ?level)

- to describe translation services, where ?level can be:
  - **Extension** - O1 extends the ontology O2
  - **Identical** - O1 and O2 are identical
  - **Equivalent** - O1 and O2 are equivalent
  - **Strongly-translatable** - every term in O1 is translatable to a term in O2 without loss of information
  - **Weakly-translatable** - some terms in O1 are translatable to terms in O2 with some loss of information
  - **Approx-translatable** - translating terms from O1 to O2 may introduce some inconsistencies
Ontology Conclusions

• Shared ontologies are essential for agent communication and knowledge sharing
• Ontology tools and standards are important
  – Ontolingua and OKBC are good examples
  – XML and RDF may be a next step
• Some large general ontologies are available
  – Cyc, WFB, Wordnet, …
• For more information…
  – http://www.ontology.org/
  – Ontology mailing list: send mail to majordomo@cs.umbc.edu with “info ontology” in message body for information.

Software Agent Research at UMBC
UMBC agent research

- Funding from NIST, DARPA, NSA, IBM, Fujitsu
- Foci:
  - Agent communication languages
  - Scalable Information filtering and retrieval
  - Mobile agent frameworks and infrastructure
  - Data mining
  - Applications to several problem domains
    - enterprise integration
    - distributed information retrieval
    - network management
    - Electronic commerce

Jackal

A Communications Infrastructure for Java-based Multi-agent Systems

- a Java package facilitating the use of the KQML and FIPA ACL
- Presents a simple yet powerful API.
- Situates messages within conversational context.
- Blackboard provides flexible interface to message traffic.
- Agent behavior specified by Java threads, Jess rules, (Prolog code)
- Jackal ≈ FIPA platform and migrating toward FIPA compliance
- http://jackal.cs.umbc.edu/
Jackal Architecture: Overview

Contribution Component:
Distribute messages to various components, as per requests.

Conversation Component:
Filter messages through individual contexts.

API

Agent

Agent Components/Threads

Conversation-based Protocols

- Allow more intuitive and convenient method for handling messages in context.
  - Through conversation composition, scale to varying levels of granularity.
  - Provide conversation management independent of agent implementation.
  - Facilitate communication through conversation sharing.

- Conversation in jackal
  - Currently modeled as DFA
  - Each conversation managed by separate thread.
  - Maintain context through local data store, accessible to agent.
  - Declarative specification.
Current Conversation Specification

// Conversation Template
(conversation
  (name kqml-ask-one)
  (author "R. Scott Cost")
  (date "3/4/98")
  (start-state START)
  (accepting-states STOP)
  (transitions
    (arc (label ask-one) (from START) (to Asked) (match "(ask-one)")
    (arc (label tell)    (from Asked) (to STOP)  (match ")(tell)")
    (arc (label deny)    (from Asked) (to STOP)  (match "(deny)")
    (arc (label untell)  (from Asked) (to STOP)  (match ")(untell)")
    (arc (label sorry)   (from Asked) (to STOP)  (match "(sorry)")
    (arc (label error)   (from Asked) (to STOP)  (match "(error)"))))

Using Colored Petri Net Specifications

- Colored Petri Nets (CPNs) offer a well defined and expressive formal model for specifying agent conversations and behavior.
- Formally equivalent to regular Petri nets.
- Supports modeling of concurrency and resource sharing.
- Widely used and supported by a large body of literature
- Simple enough for general use and practical implementation.
- Intuitive graphical representation
Colored Petri Net (CPN)

- A CPN model consists of a set of modules (pages) which each contains a network of places, transitions and arcs.

- The modules interact with each other through a set of well-defined interfaces, much like modules or subroutines in a programming language.

- This CPN describes a distributed database.

Properties of CPNs

- The CPN formalism has a formal, mathematical representation with a well-defined syntax and semantics.

- CPN models can be made with or without explicit reference to time.
  - untimed CPN models are used to validate the functional/logical correctness of a system, while
  - timed CPN models are used to evaluate performance

- CP-nets can be simulated interactively or automatically.

- CP-nets also offers more formal verification methods, known as state space analysis and invariant analysis. In this way it is possible to prove certain behavioral properties.
Jackal Message Distributor

- Provides a simple blackboard through which the agent logic access agent services and threads.
- Communication focal point for agent processes.
- Processes can check or wait for messages of a specified type.
- Messages are bundled with context information.

- Match on:
  - Message form/content
  - Priority
  - Others

- Specify:
  - Number of matches
  - Delete or write protect matches
  - Lifetime of a request

Typical Components of a Multi-agent System

- Client Agents
- White Pages Service (Agent Name Server)
- Yellow Pages Service (Broker)
- Resource Management (Control Agent)
- Resource Providers (Libraries)
- General Service Providers
KQML Naming Scheme (KNS)

- A DNS-like scheme for agent naming
- Protocols for dynamic group formation and disbanding.
- Transparent maintenance of a distributed, persistent identity for agents.
- Facilities for ‘no-fault’ access to agents and basic agent information.

Funder: National Institute of Standards and Technology / Advanced Technology Program
- Technologies for the Integration of Manufacturing Applications (TIMA)
  - ~ $45M over six years in two ATP projects

Goal: Plug and Play framework of business objectives and integration-enabling tools allowing a suite of solutions that can be implemented “out-of-the-box” at small and midsized manufacturing and process sites including MES, ERP, Finite Scheduling, and Capacity Analysis/Decision Support

Objectives: interoperability, configurability, adaptability, extensibility, plug and play.
Manufacturing Enterprise Integration

- Integration of planning and execution is imperative for agile manufacturing
  - parts delivery is delayed by the part supplier
  - a preferred customer asks to move ahead a delivery
  - machine breaks down on shop floor
- This involves collaboration among business applications and managers
- Business applications are legacy systems
  - not intended to talk to each other (no API, no means of communication)
  - developed over long period of time (expensive to change)
  - many decision steps are not covered (white space between applications)
- Multi-agent system (MAS) approach
  - flexible and dynamic communication among applications
  - plug-and-play
  - interface agents to interact with people
  - other agents to fill the white space between business applications

CIIMPLEX Architecture

- MQ Series
- Gateway Agent
- CIIMPLEX Agent Communication Infrastructure
- ANS
- BA
- Monitoring Agent
- Data mining Agent
- CIIMPLEX Analysis Agent
- Scenario Coordination Agent
- Java-Enabled Web Browser

Copyright 1998-99 Tim Finin
Negotiation among agents in the Supply Chain

Goals

• Support automated or semi-automated negotiation among applications in a supply-chain
• Develop an approach that can integrate with existing business practices and procedures
• Develop an approach that uses standards and technologies likely to be acceptable to the business community

Specific Approach

1. Use (modified) FIPA ACL primitives for negotiation
   – Important contribution is the set of primitives and their semantics
2. Use XML, extended with KIF, as the content language
   – KIF-based extensions allow the use of constraints and business rules
3. Introduce the notion of adjustable autonomy into agent-based supply chain negotiation
   – Use of “decision rules” to decide how to respond augmented
   – with “authorization rules” which decide if the action should be reviewed for authorization and by whom.
1 Negotiation primitives

- Based on the FIPA ACL with extensions
- Basic negotiation primitives:
  - **cfp**: call for proposals
  - **propose**: propose (or counter-propose) an action
  - **accept-proposal**: accept a proposal
  - **reject-proposal**: reject a proposal (with optional reason)
- Other ACL primitives useful in negotiation
  - inform, query, request, not_understood, refuse, ...
  - advertise, subscribe, broker, register, ...
- Specific negotiation protocols are defined using these primitives
  - e.g., Iterated-contract-net, English-auction, etc.

Examples of negotiation primitives

```xml
(cfp
 :from "http://umbc.edu/~finin/self"
 :language KIF/XML
 :ontology http://ec.ibm.com/ec42
 :content "<proposal>
 <salesContract>
 <buyer> "http://umbc.edu/~finin/self" </buyer>
 <price unit=usd> ?Price </price>
 <goods> … </goods>
 </salesContract>
 <constraints>
 <kif> <rel name=less>
 <arg ?Price> <arg 5000>
 </rel>
 </kif>
 </constraints>
</proposal>"
 :protocol bestAndFinal03
 :reply-with cfp3245a.11.06.99 )

(propose
 :to "http://umbc.edu/~finin/self"
 :from "http://compusa.com/self"
 :language KIF/XML
 :ontology http://ec.ibm.com/ec42
 :content "<proposal>
 <salesContract>
 <buyer> "http://umbc.edu/~finin/self" </buyer>
 <price unit=usd> 4500 </price>
 <goods> … </goods>
 </salesContract>
 <constraints>
 <kif> <rel name=less>
 <arg ?Price> <arg 5000>
 </rel>
 </kif>
 </constraints>
</proposal>"
 :in-reply-to cfp3245a.11.06.99
 :protocol bestAndFinal03
 :reply-with offer4762579.11.06.736125409 )
```
Defining negotiation protocols

- Different protocols can be defined using the communicative primitives
  - contract-net
  - iterated contract-net
  - English auction
  - etc
- Most protocols can be defined with a simple deterministic finite-state automata (DFA) formalism. More complicated ones will require CPNs.
- Negotiations can be augmented by “side conversations” composed of queries, informs, etc.

2 An XML-based content

- We are exploring the use of an XML-based content language
- XML will be the language of the web
  - XML will rapidly become the dominant “content” encoding used on the web for ecommerce and other applications.
  - Businesses (and their agents) will continue to interact by exchanging documents (POs, invoices, catalogues, etc) but encoded in XML.
- XML supports the required extensions
  - We envision extensions to encode rules, constraints and agent-agent negotiation.
Laptop description seen “by eye”

Laptop Computer

IBM Thinkpad
560X
233 Mhz
32 Mb
4 Gb
4.1 pounds
$3200

HTML laptop description

<TITLE>Laptop Computer</TITLE>
<BODY>
<UL>
<L1>IBM Thinkpad 560X
<img src="560.gif">
<L1>233 Mhz
<L1>32 Mb
<L1>4 Gb
<L1>4.1 pounds
<L1>$3200
</UL>
</BODY>
XML Laptop Description

```xml
<COMPUTER TYPE="LAPTOP">
  <MANUFACTURER>IBM</MANUFACTURER>
  <LINE>Thinkpad</LINE>
  <MODEL>560X</MODEL>
  <SPEED>233</SPEED>
  <MEMORY>32</MEMORY>
  <DISK>4</DISK>
  <WEIGHT>4.1</WEIGHT>
  <PRICE>3200</PRICE>
</COMPUTER>
```

CARROT: Cooperating Agent-based Routing and Retrieval of Text

- An example of a mediated agent-based information retrieval architecture developed at UMBC.
- Agents provide access to different corpora, using existing IR engines.
- Agents share metadata with Broker agents, which route queries and new documents to the “right” place(s).
- Two enabling technologies:
  - KQML agent communication language
  - N-gram processing
Broker and back-end agents

**Carrot Broker**
- Interacts with local IR/DB engines to access data
- No interference with existing applications
- Generates metadata, to be shared with one or more brokers
- Metadata for a set of documents is a compressed n-gram profile

**Back-end agent**
- Collects metadata from back-end agents
- Uses Telltale to manage these metadata corpora
- Otherwise similar to back-end agents
- Can be organized in hierarchies
VR based visualization of retrieval

• The only way to comprehend a large corpus or result set is through visualization.
• SFA, for example, provides
  – Real-time, interactive stereo viewing of results of information retrieval engine
  – Each document returned is rendered as a glyph (icon)
  – Document properties mapped to 3D location, shape, color, transparency, and texture.
  – Spatialization of complex relationships and comprehensible display of multiple variables

VR Approach

• Immersive
  – Isolates the user from environment
  – Expensive

• Minimally-immersive
  – Access to environment
  – Collaboration possible
  – Low cost
  – Two hands give proprioception

• Uses Two 3D Trackers with Buttons
  – User manipulates 3D scene with trackers
  – Each hand has a distinct role -- left sets up context and right performs fine manipulation
Visualizing a document space

Mappings

X: similarity to “federal reserve bank”
Y: similarity to “commodity prices”
Z: similarity to “foreign exchange rate of the dollar”

Shape: similarity to “coup attempt against Noreiga” with cube as lowest and cone as highest

Color: age of document with blue as the oldest and yellow as the newest

Transparency:

Texture:

Conclusions and Prospects
Some key ideas

- Software agents offer a new paradigm for very large scale distributed heterogeneous applications.
- The paradigm focuses on the interactions of autonomous, cooperating processes which can adapt to humans and other agents.
- Agent Communication Languages are a key enabling technology
  - Mobility is an orthogonal characteristic which many, but not all, consider central.
  - Intelligence is always a desirable characteristic but is not strictly required by the paradigm.
- The paradigm is still forming and ACLs will continue to evolve.

Prospects

- FIPA’s ACL is likely to be the next iteration of a widely used standard ACL.
- It’s not clear how ACLs will participate in the rapidly evolving world of Internet languages and protocols
  - The ACL “territory” may be overtaken by efforts from a programming language (e.g., Java, Jini), another interoperability language (e.g., CORBA) or Web technology (e.g., XML).
  - The Agent community is a small fish compared to, e.g., the Java community. What will Microsoft do?
- Many are experimenting with XML for agent communication
  - XML is a good way to represent structured information (e.g., ACL messages, KIF-like content) that is easy to use and understand by all agents, both human and software
  - We’ve developed DTDs and style sheets for FIPA ACL and KIF
  - XML is not a silver bullet.
For More Information

- General information on software agents
  - http://www.cs.umbc.edu/agents
- FIPA
  - http://www.fipa.org/
- KQML
  - http://www.cs.umbc.edu/kqml
- KIF
  - http://www.cs.umbc.edu/kif
- Ontologies
  - http://www.cs.umbc.edu/ontology/
- Agent Communication Languages
  - http://www.cs.umbc.edu/acl/