Coordination Technologies for Internet Agents

Franco Zambonelli,
Andrea Omicini,
Paolo Ciancarini

Contacts

- **Franco Zambonelli**
  DSI - University of Modena and Reggio Emilia
  franco.zambonelli@unimo.it
  http://www.dsi.unimo.it/Staff/st35/Zambonelli.idc

- **Andrea Omicini**
  LIA - DEIS - University of Bologna
  aomicini@deis.unibo.it
  http://www.lia.deis.unibo.it/Staff/AndreaOmicini

- **Paolo Ciancarini**
  DSI - University of Bologna
  ciancarini@cs.unibo.it
  http://www.cs.unibo.it/~cianca

Part 1: Outline

- Features of Internet agents
- Interactions in Internet agent applications
- The role of ACLs and CORBA
- Survey of several interaction models
  - direct interaction
  - meetings
  - blackboards
  - tuple spaces
- Impact in case study application

Internet Applications

- The Internet no longer used as a simple
  - communication infrastructure OR
  - information repository
- The Internet as a *global distributed computing system*
  - traditional distributed applications executed at a world-wide scale
  - new application areas
- Communication, computation and data management intertwined in several application areas
  - E-commerce
  - Information Retrieval
  - CSCW

Internet Agents

- Autonomous (*task-oriented*)
  - execute in *autonomy* to achieve a *goal*
- Network-aware (*location-aware*)
  - conscious about the *distribution* of the environment
- Mobile
  - can proactively roam through Internet sites (changing their *execution environment*)
  - mobile devices can be abstracted as mobile Internet agents
- Interactive
  - inter-agent interactions
  - agent-environment interactions

Autonomy

- Process-based and Object-based applications
  - global goal achieved via a *global control scheme* for the application entities
  - design by *delegation of control*
- Agent-based applications
  - sub-goals assigned to autonomous agents which try to achieve in autonomy their own goal
  - design by *delegation of responsibility*
- In the Internet autonomy is necessary since
  - global control at a world-wide scale cannot be implemented
  - unpredictability requires dynamic decisions
  - efficiency and reliability encourage decentralization

Network-Awareness

- Traditional approaches to distributed programming enforces transparency
  - abstraction of a single virtual computing system
- In the Internet it is unfeasible
  - to hide physical distribution
  - to build/abstract a single global virtual machine, due to decentralized management
- Network-awareness
  - application components are aware of a multiplicity of computing resources distributed over the network

Mobility

- In traditional distributed system
  - mobility managed by the run-time support transparently to applications
- In Internet applications
  - application components can migrate across computing environments
- Motivation
  - save of bandwidth (local access to data and services)
  - robustness (independency from connection flaws)
  - models mobile users and mobile devices
  - manageable abstraction makes application design simpler

Interactivity

- Multiagent systems (MAS)
  - a multiplicity of agents interact to achieve a common goal
  - inter-agent and agent-environment interactions
  - individual and social tasks towards the global MAS task
- Users
  - users might want to interact with agents (with MAS)
- External entities
  - foreign agents
  - Web services
  - CORBA applications
  - legacy systems

Enabling Interaction

- Agents must *interoperate* despite
  - heterogeneity of
    - source
    - representation of the domain
    - architecture, language, technology
    - attitude, BDI, knowledge
  - unpredictability of behaviour due to
    - pro-activity (agents can say “do”)
    - autonomy (agents can say “no”)
    - mobility (agents can say “go”)
- SW infrastructures to enable interaction
  - Middleware

Middleware

- quite a generic term …
  - protocols and SW for open, distributed, heterogeneous architectures (ISO-OSI layer 7)
  - examples: interapplication protocols, scripting languages, document-oriented frameworks, …
- ensuring global features of the interaction space
  - interoperability
  - security
  - transactionality
  - …

CORBA

- Common Object Request Broker Architecture
  - ORB
  - Distributed CORBA Object
  - Interface Definition Language
    - Interface Repository
  - CORBA services
    - Security, Transactions, Persistency, …
- Benefits
  - network transparency, self-describing system, static/dynamic method invocation, embedded security and transactions, separation between interface and implementation (legacy), …

Enabling Communication

- Agents exchange information
  - Enabling interaction is not enough
  - Communication is interaction *plus* information
- Communication issues
  - communication *language*
    - syntax, semantics, ontology
  - communication *acts* (speech acts)
    - performatives
- Agent Communication Languages (*ACL*)
- Mediators, Ontologies

Agent Communication Languages

- Speech-act languages
  - KQML *vs.* FIPA
- KQML
  - tenth of performatives + communication language (KIF, SQL, …)
  - Virtual Knowledge Base for beliefs and goals
  - pragmatics
- FIPA
  - 6 performatives + performative composition
  - formal semantics (from Arcol)
  - strong requirements on agents
- *Conversational* model for agent interaction

Enabling Mobility

- Infrastructures for mobile agent support
- Modelling/enabling the interaction between an agent and an execution environment
  - trusting/entering/living in/leaving an environment
- Examples of systems
  - Ara, Aglets, Ambit, D'Agent, ffMain, MOLE, PageSpace, Mars, SOMA, TuCSoN, …
  - different interaction issues addressed
- We focus on *systems*

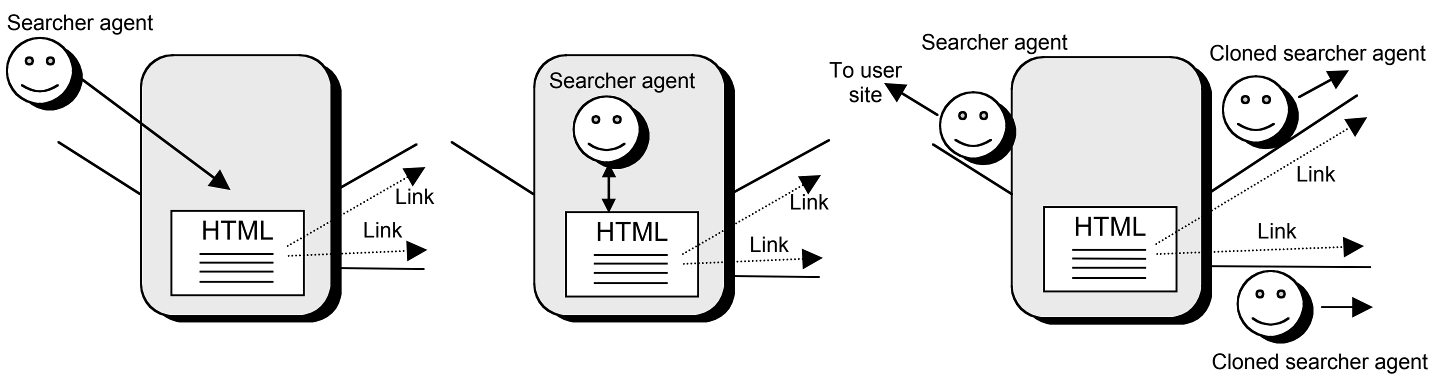
Interaction Models: a Taxonomy

- Temporal uncoupling
  - no activity synchronization
  - no time co-presence
- Spatial/Name uncoupling
  - no mutual knowledge (*e.g.* name identifier)

Spatial \ Temporal	Coupled	Uncoupled
Coupled	Direct *Aglets* *D'Agents*	Blackboard-Based *Ambit* *ffMain*
Uncoupled	Meeting-Oriented *Ara* *MOLE*	Linda-like *PageSpace* *TuCSoN* *MARS*

Case Study: Distributed Information Retrieval

- Agents reach remote sites to access and analyze HTML pages
- Cloning to follow remote links (possibly interesting) ⇨ dynamic tree of mobile agents
- After a visit, agents come back with the URLs of interests (*e.g.* matching a keyword)



Interaction in the Case Study

- With the local execution environment:
  – retrieve info on local HTML files
- Facts:
  – require access control
  – heterogeneity of execution environments
- Inter-agent
  – avoid duplicated visits on the same sites (due to cross-references in HTML pages)
- Facts
  – agents are dynamically created
  – cannot foresee their number and location

Case Study: Sample Code in Aglets *(simplified)*

```
Public class ExplorerAgent extends Aglet {
boolean _clone = false;      boolean visited = false;      String keyword = null;
URL destination = null;      URL home_URL = "atp://my.site" Vector new_destinations;
….
public void onCreation(Object args) {
    destination = (URL)((Object[]) args)[0]; keyword = (String)((Object[]) args)[1];
    dispatch(destination);}
….
public void run()
{   // CHECKS IF ALREADY VISITED:        INTER-AGENT COORDINATION
    if(visited) dispatch(home_URL);  // the agent has nothing to do on this site
    // RETRIEVES LOCAL HTML FILES:       AGENT-ENVIRONMENT COORDINATION
    // search for the keyword in the HTML files and for other interesting sites
    // for instance, is a Vector of references to HTML files has been retrieved:
            If (HTMLFiles.search_keyword(keyword)) …..
            new_destinations = this.extractUniqueURLs(HTMLFiles); …..
    // create clones and send them to the computed destinations
    for(int i=0; i < destinations.size(); i++)   {
            clone();
            if(_clone) // the  _clone variable can be set to true in the OnClone() method
                    dispacth((URL)new_destinations.elementAt(i);} // goes to a new destination
    dispatch(home_URL); // the original agent comes back home
}
```

Direct Interaction Models

- Peer-to-peer agent communication (message-passing)
- Client-server access to services (RPC, RMI)
- Systems: most of Java-agent systems (*Aglets*, *D'Agents*, …)
- Spatial coupling
  - entities have to know each other to interact
- Temporal coupling
  - synchronization of the activities during interaction

Direct Interaction: Pro & Cons

- Advantages
  - well-known and understood model
  - several tools available
  - sometimes very efficient
- Drawbacks
  - how to deal with agent unpredictability
    - e.g.: mobility (tracing, forwarding, only-once messages, performances)
  - how to deal with security
    - agent exchange message without mediation
  - how to scale up communication protocols & patterns
    - combinatorial explosion of communication channels

Case Study: Direct Interaction

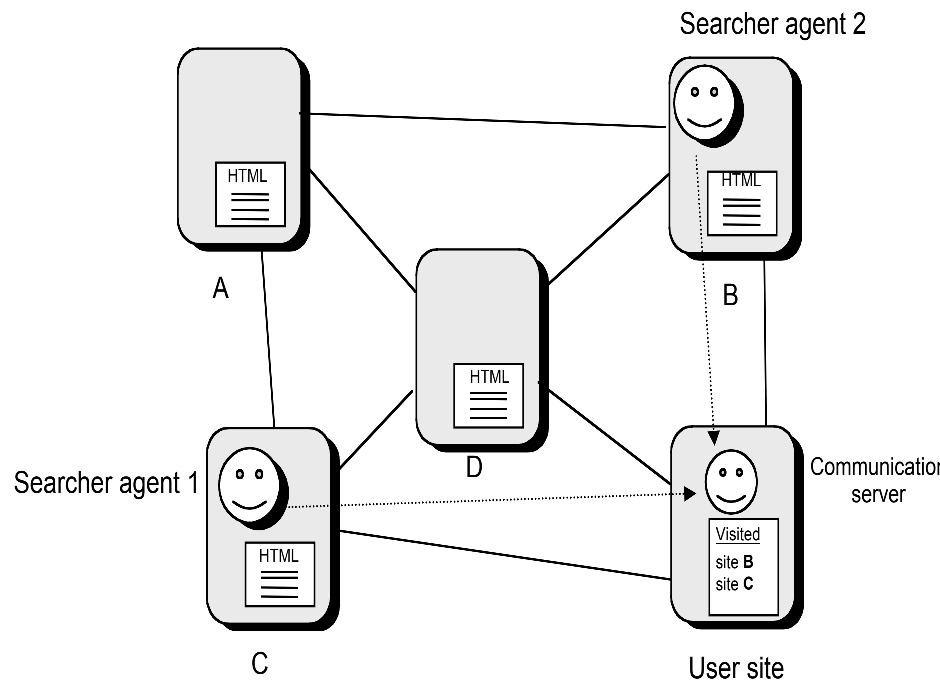
- Avoiding multiple visits
  - agents cannot communicate directly with each other easily
    - who they are?
    - where they are?
- Access to local resources via a WWW server
  - should the agent explicitly navigates to all local links to retrieve all HTML files?
  - complex solution!

```
URL starting_point = new URL (“www.starting.site”);
Vector other_local_HTML_files = recursively_parse(starting_point);
// the code of the recursively_parse method may be rather complex and time
consuming….
```

Case Study : Direct Interaction Design

- Application-level definition of a specialized communication server
  - agents ask to it about previous visits to a site

```
Message msg = new Message(my_id, today, local_URL);
AgletProxy com_ser = getAgletContext.getAgletProxy(URL,
                                                    com_ser_id);

visited = com_ser.sendMessage(msg);
```



- Drawbacks
  - centralisation
  - unreliability

Meetings

- "*Agora*" for agent interactions
- meetings are opened by application agents (always-open meetings abstract the role of services)
- an agent can join a meeting and there communicate with other entities in the meeting
- Systems: *Telescript*, *Ara*, *MOLE* (to some extent)
- Spatial uncoupling
  - only meeting point must be known to enable interactions
- Temporal coupling
  - agents must be on the same place at the same time to interact

Meetings: Pro & Cons

- Advantages
  - locally constrained meetings lead to predictable performances in interactions
  - meetings mediate all interactions and achieve good application-level control over them
- Drawbacks
  - limitation of the agents’ autonomy
  - who, where and when to open meeting points for an application?

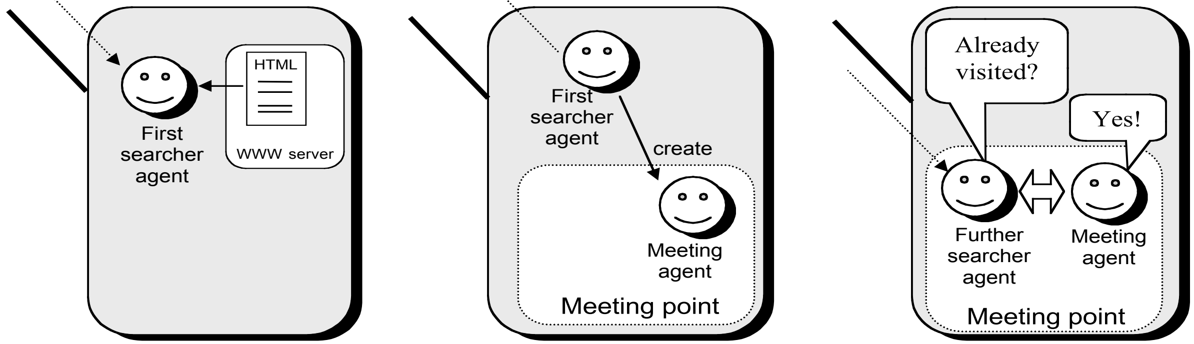
Case Study: Meetings

- Avoiding multiple visits
  - Other agents must explicitly open meeting point to get notified of already visited sites
  - Where to create meeting point?
    - one site: centralisation
    - selected given sites: requires agent knowledge
    - all visited sites: spamming of meetings
- Access to local data
  - An always-open meeting on a site can abstract a local WWW service

Case Study: Meeting Design

- The first agent on a node creates a meeting point
- Further agents arriving on that site will enter the meeting point and will get notified of a previous visit
- Advantages
  - fully decentralised
- Drawbacks
  - security and garbage collection of meetings

```
//aglet-like code for meetings
Message msg = new Message(my_id, today);
meet_ag =context.Meet(meeting_agent_id);
if(meet_ag == null)
{  meet = context.createMeeting(meeting_classURL,
                                meeting_agent_id);
   meet.StoreMessage(msg);     }
else   visited = meet_ag.sendMessage(my_id, today);
```



Blackboard-based Interaction

- Each node/domain of nodes should provide for a shared information space to be used by agents to store and retrieve information
- Wherever agents are, they can exploit the blackboard to communicate with other application agents
- Systems: *ffMAIN* (http-based), *Ambit* (file-based)
- Temporal uncoupling
  - messages can be left on and retrieved from blackboards at any time
- Spatial coupling:
  - requires agreement on message identifiers (e.g., URLs in ffMAIN, filenames in Ambit)

Blackboards: Pros & Cons

- Advantages
  - temporal uncoupling make it easier for agent to communicate while preserving their autonomy
  - security (all interactions can be monitored by the blackboard)

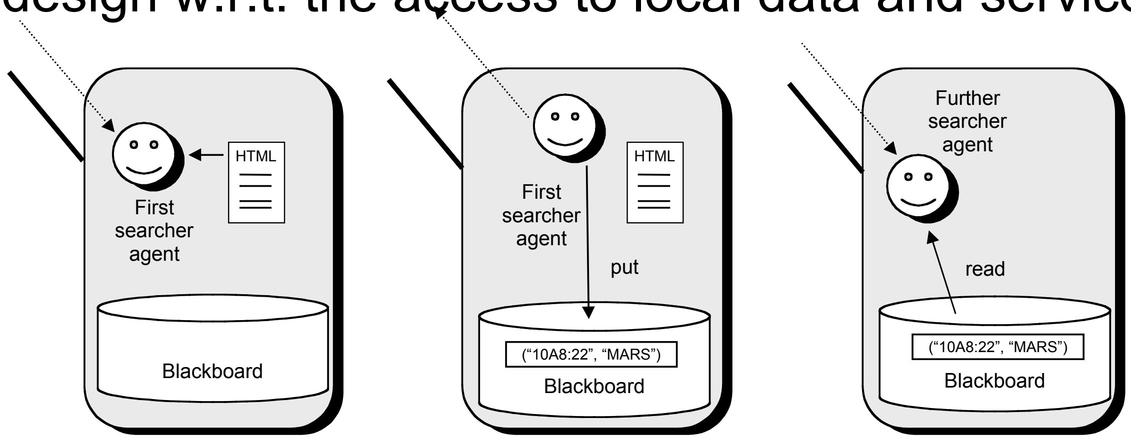
- Drawbacks
  - spatial coupling increases application complexity due to a priori agreements on message identifier
  - complex interaction protocols may be difficult to implement
  - requires a specific infrastructure for the Internet

Case Study: Blackboards

- Avoiding multiple visits
  - the blackboard can be used as a mailbox
- Access to local data
  - A blackboard can publish data about its public files and information on locally available services
  - Requires a standard way to identify information
  - How can agents selectively retrieve HTML files only?

Case Study: Blackboard-based Design

- Agents get/put messages to avoid multiple visits
- Agents access the local blackboard to discover the presence of HTML pages in the site
- Advantages
  - simple design w.r.t. inter-agent coordination
- Drawbacks
  - complex design w.r.t. the access to local data and services



Tuple Space Interaction

- Each node/domain of nodes should provide for a tuple space to be used by agents to store and retrieve tuples via a Linda-like interface
- Wherever agents are, they can exploit the local tuple space to communicate with other application agents
- Systems: *JavaSpaces, T Spaces, PageSpace, MARS, TuCSoN, Lime*
- Temporal uncoupling
  - as in the blackboard case
- Spatial coupling
  - associative access to tuples does not require strong a priori agreements or knowledge

Tuple Spaces: Pros & Cons

- Advantages
  - full uncoupling suits both mobility of application components and uncertainty of the Internet environments
  - security (inter-agent interactions can be monitored by the blackboard)
  - simplicity of communication primitives

- Drawbacks
  - requires a specific infrastructure for the Internet

Case Study: Tuple Spaces

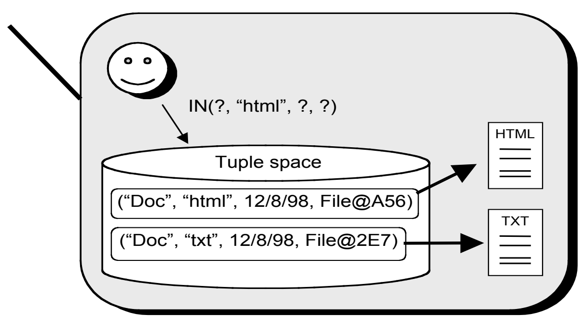
- Avoiding multiple visits
  - agents must check if the tuple space contains tuples left by other agents in a previous visit (similar to the blackboard case)
- Access to local data
  - the tuple space can publish data and information on locally available services
  - associative access eases the retrieval of this information

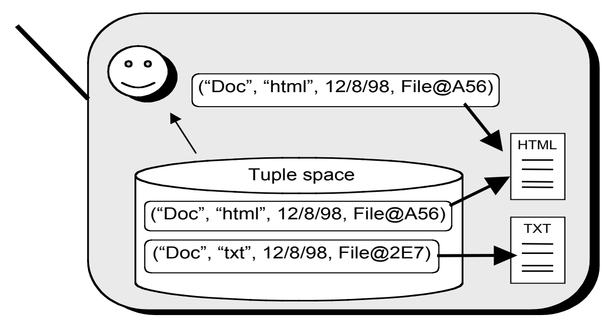
Case Study: Tuple-based Design

- previous visits of other applications agents
- info on all HTML pages via a single requests

```
if(!(visited = inp(my_application_id, when ?))
      out(my_application_it, System.GetTime());



HTMLFiles = read_all("Doc","html",date?,name?);
```

- Advantages
  – security for interactions
  – very simple agent design





Summarizing

- Coupled interaction do not suit Internet agents
  - difficult to directly interact at a world-wide scale
  - odd and complex design choices
- Tuple-based models suitable for the Internet
  - uncoupling suits autonomy, mobility and dynamicity
  - associativity suits uncertainty of the Internet
  - can lead to simpler application design
- Actually, it is a *coordination model*
  - clear separation between interaction and computation
  - clear definition of interaction laws
  - shift from *enabling interaction* to *ruling interactions*

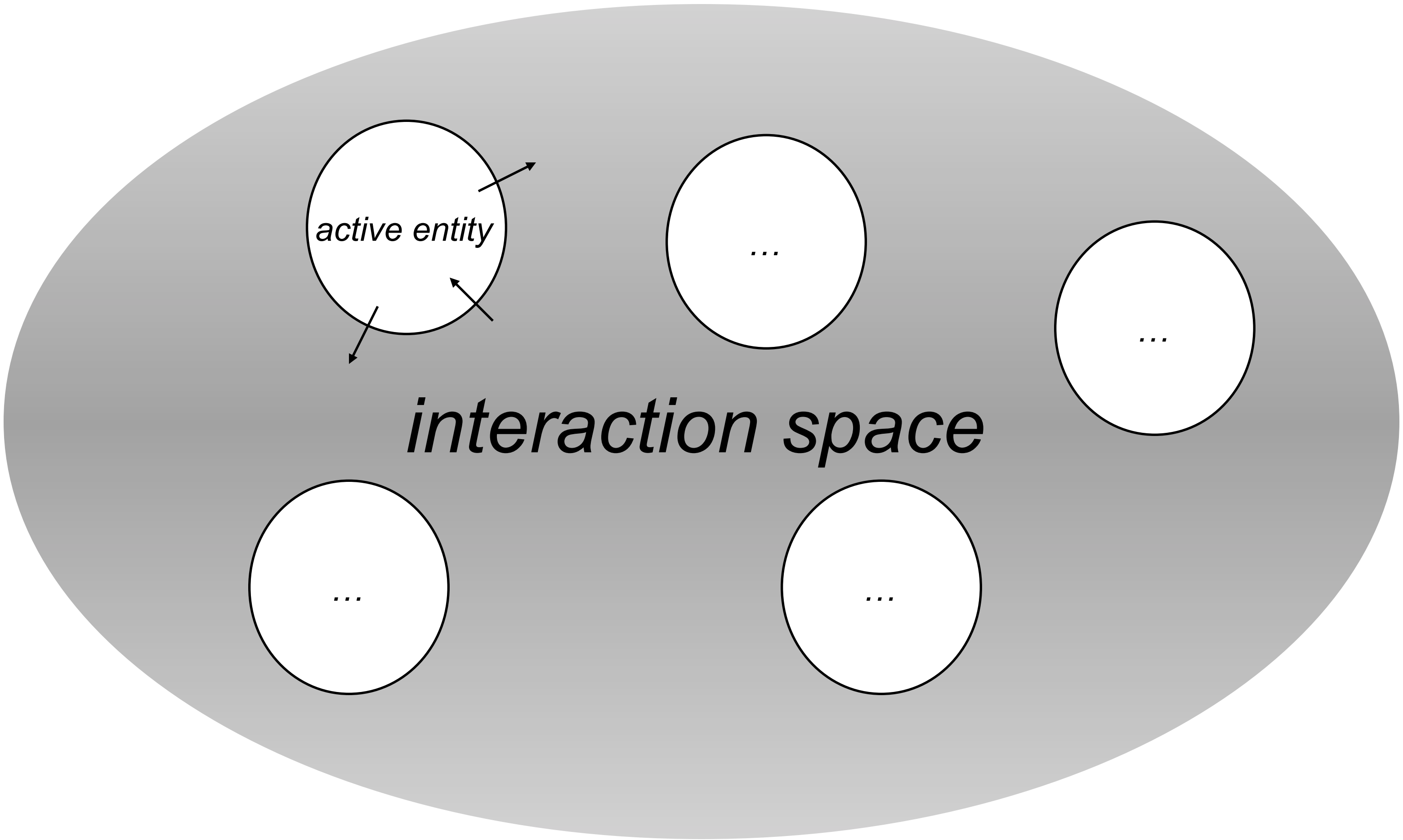
Part 2: Outline

- Coordination Models
  - definitions
  - motivations
  - the software engineering perspective
- Survey of existing systems
  - JavaSpaces
  - Lime
  - MARS
  - PageSpace
  - T Spaces
  - TuCSoN
- Open issues and research directions

What is Coordination?

- “Coordination is managing dependencies between activities” (Malone & Crowston)
  - interdisciplinary issue
- “Coordination is constraining the interaction” (Wegner)“
  - CS viewpoint
- Coordination is the process of building programs by gluing together active pieces” => “A coordination model is the glue that binds together activities into an ensemble” (Carriero & Gelernter)
- “A coordination model provides a framework in which the interaction of active and independent entities … can be expressed” (Ciancarini)
  - how to express interaction

The Interaction Space



What is a Coordination Model?

- A conceptual framework to model the space of interaction
- An ontology for coordination models
  - Coordinables
    - the entities (agents, processes, objects, components, and so on) whose mutual interaction is ruled by the model
  - Coordination media
    - the abstractions enabling and ruling the interaction among the coordinables
  - Coordination laws
    - the rules governing the interaction among coordinables as well as the behaviour of the coordination media
    - communication, coordination, meta-coordination languages

Coordinables

- A coordination model defines
  - what is a coordinable to it
- No interest in the coordinable's inner structure
  - coordination concerns interaction
- Definition *on the boundary*
  - observable behaviour
- *Admissible coordination entity*
  - any entity expressing through the model's communication and coordination languages

Coordination Language

- Acts of communication
  - speech acts
    - convey information
    - change the world
- A coordination model should define
  - both the *syntax*
  - and the *semantics* of the
- *Admissible coordination primitives*
  - semantics expressed in terms of effects on the interaction space
  - input, output, and interaction events

Communication Language

- A communication language
  - conveys information
- A coordination model *may* define
  - the *syntax*
  - but NOT the *semantics*

  of the admissible sentences of the communication language
- Relation with coordination language
  - admissible sentences associated to any admissible primitive

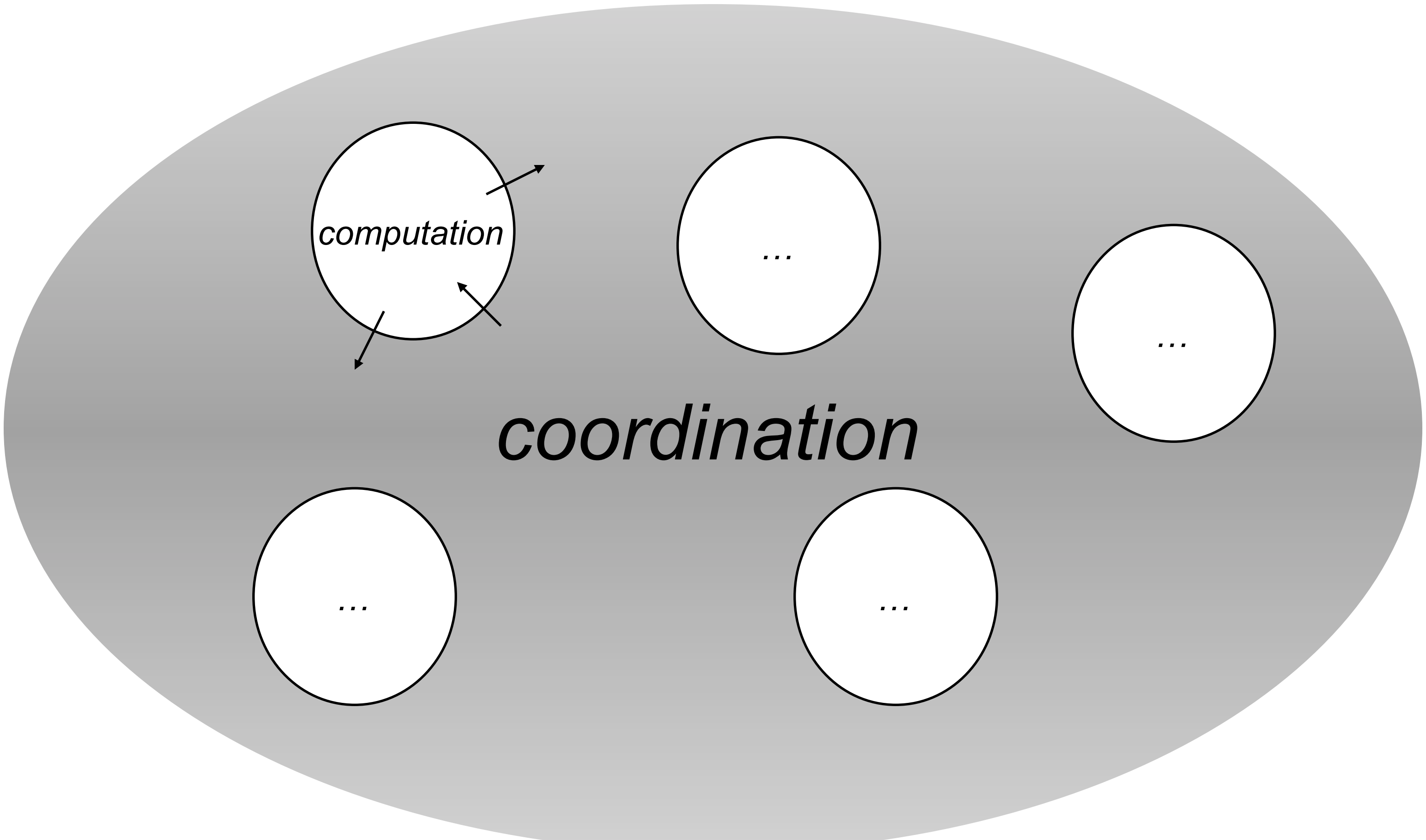
Coordination Media

- Populating the interaction space
  – channels, monitors, connectors, blackboards, …
- A coordination model defines
  – what is a coordination medium to it
- Full definition
  – observable & inner behaviour
  – state transitions
- Given a model, are the coordination media
  – a multiplicity in number/kind?
  – global or local, private or public?
  – physically distributed/centralised?
  – fixed in their behaviour?
  – inspectable?
  – dynamically modifiable?

Coordination Laws

- Rules governing the space of agent interaction
- Defined in terms of
  - communication & coordination languages
- Determined by
  - coordination media behaviour
- Given a model, are the coordination laws
  - completely fixed a-priori by the model?
  - generally fixed but specialisable by need?
  - fully programmable?
  - implicitly/explicitly represented?
  - inspectable?
  - dynamically modifiable?
  - globally/locally affecting the interaction space?
  - physically distributed/centralised?

Coordination: Ruling the Interaction Space



Enabling vs. Ruling Interaction

- ACL, middleware, mediators, …
  - *enabling* interaction/communication
  - *defining* the interaction space
  - agents can interoperate and exchange information to achieve their *individual tasks*
- Coordination models and languages
  - *ruling* interaction/communication
  - *governing* the interaction space
  - agents can be combined in societies pursuing *social tasks* as a whole

Social Tasks

- Agents' task in a MAS
  - individual and social tasks
- Example: conference management system
  - each referee reviews at least ten papers
    - individual: can be defined in terms of a single agent's task
  - each paper is ensured to have at least three reviews
    - social: *cannot* be defined in terms of any single agent's task
- How do we build social tasks? *(I)*
  - typical solution: *ad hoc* meta-level agent
    - inelegant, ineffective
    - an agent, oh, really?
      - do we need it
      - can we do it?
  - a symptom, rather than a solution
    - we have no suitable abstractions

Coordination Models: Engineering the Interaction Space

- Engineering viewpoint
  - how to rule agent societies?
  - how to design and implement collective behaviour?
- Coordination
  - ruling the interaction space
- Agent interaction space
  - space of communication ⇨ space of coordination
- How do we build social tasks? *(II)*
  - exploiting coordination models to shape the space of agent interaction

Why a Coordination Model?

- Engineering the interaction space
  - agent languages and architectures for engineering individual agents
  - coordination models and languages for engineering agent societies
- Need for specific high-level abstractions and powerful mechanisms
  - to support the design and development of multiagent systems as far as interaction is concerned
- Suggesting/supporting new methodologies
  - for the construction of open, distributed, heterogeneous, and mobile systems
- Intrinsically providing systems with features
  - e.g., flexibility, security, support for heterogeneity, intelligence, …

Multiagent System Engineering: the Coordination Viewpoint

- Individual & social tasks
  - driving the design
- Impact on the design
  - individual tasks ⇨ design of single agents
  - social tasks ⇨ design of
    - agent interaction protocols
    - agent interaction rules
- How do we build social tasks? *(III)*
  - defining agent societies
    - around coordination media
  - ruling agent societies so as to make them behave as we require
    - by defining and implementing the suitable coordination laws

Engineered Design

- Defining both individual and social tasks
- Designing individual agents and societies
  - task-oriented design (delegation of *responsibility*)
- Designing individual interaction protocols
  - in terms of
    - communication & coordination languages
  - according to
    - agent’s features (task, domain representation & knowledge, …)
    - agent’s involvement/commitment in societies
- Defining coordination laws
  - in terms of
    - abstractions and tools provided by the coordination model

Computation *vs.* Coordination

- Coordination *languages*
  - “Clean separation between computation and coordination”
  - WOW!!! … so what?!?
- Separation at the *language* level
- Coordination models should provide
  - clean separation between computation and coordination at the *design* level
    - agents should *not* care of coordination details
- Example: conference management system
  - each paper is ensured to have at least three reviews
    - well, it *could* actually be defined in terms of each single agent’s task: every agent should worry about this, and embed a complex verification protocol, negotiate with other agents, *etc.*
    - this would mix coordination and computation issues at the agent design level
  - *coupled design*: bad engineering

Incremental MAS Development

- Coordination media as independent component
  - embedding coordination laws
  - reusable
  - incrementally modifiable
- Example: conference management system
  - each paper is ensured to have at least three reviews
    AND
    at least two referees from two different institutes
    - when computation and coordination issues are mixed in the agent's design (*coupled* design)
      - agents have to be re-designed to embed the new coordination policy
    - when a suitably-powerful coordination model allows coordination laws to be embodied into coordination media (*uncoupled* design)
      - agents are not affected
      - coordination media can be re-defined (possibly *incrementally refined*) so as to embed the new coordination policy

Control-driven vs. Data Driven Coordination

- Control-driven vs. Data-driven (Papadopoulos & Arbab)
  - focus on *communication actions/events*
  - focus on *communication data*
- Control-driven models
  - max. flexibility and control
  - coordination as configuring a communication topology, BUT
  - low level of abstraction
    - not suitable for information-based or high-level symbolic systems
  - require full control of the interaction space
    - not suitable for open & unpredictable systems
- Internet MAS calls for data-driven models

Tuple-based coordination

- Communication based on tuples
  - communication language
    - tuple language T
    - tuple template language TT
    - template function τ: TT $\rightarrow 2^{T}$
  - C-Linda example
    - *("ok",30) $\in$ τ("ok",?i)*
- Tuple repositories as coordination media
  - tuple containers
  - associative access to tuples
  - reactive behaviour
- Coordination by
  - reading, writing, consuming tuples in the tuple repositories
  - synchronising over tuple occurrence in the tuple repositories

Benefits of Tuple-based Coordination

- Generative communication
  - spatial/temporal uncoupling
- Expressive communication abstractions
  - information-based design
- Associative access
  - good for incomplete/dynamic knowledge
- Computation vs. coordination
  - simple protocol design (*perhaps*…)

The Linda Model

- The coordination media
  - (Multiple) tuple spaces
    - globally accessible abstractions
- The coordination language
    - out, rd, in, rdp, inp
    - some notation for addressing a single tuple space at a time
- Relationship with the communication language
    - out(T), rd(TT), in(TT), rdp(TT), inp(TT)
- Limits of the model
  - fixed behaviour of the coordination media
  - limited expressiveness in terms of coordination laws
  - not enough control for Internet applications

Tuple-based Coordination on the Internet

- Definition of the coordinables
  - several entities involved in applications
  - what the coordinables are?
    - Mobile agents, WWW services, CORBA applications, etc.
- Definition of the coordination media
  - how many? Where?
  - how can they refereed and accessed?
- Definition of Coordination and Communication Languages
  - object-oriented, logic, DBMS oriented models….
- Definition of the coordination laws
  - fixed versus programmable

Coordinables in the Internet

- Different solutions possible
- All entities becomes coordinables
  - application agents
  - “agentification” of other entities
    - e.f. interface agents to make a WWW server interact accordingly to the Linda model
- All entities but application agents are reflected in the tuple space
  - the tuple space provides Linda-like access to other applications and services
- The tuple space as an additional Internet service
  - no specific accommodation of other entities w.r.t. the coordination model

Coordination Media in the Internet

- Where are tuple spaces? How agents access them?
- Transparent approach
  - tuple spaces are referred in a location-independent way and in a network-unaware way (e.g. object references)
- Implicit approach
  - tuple spaces associated to each node/domain of nodes
  - tuple spaces are not explicitly referred
  - agent implicitly access the local tuple space
- Explicit approach
  - agents access to tuple spaces in a network-aware way, e.g., via URLs
  - mixes of implicit and explicit approaches are possible

Tuple Space Models and Coordination Languages

- Which tuple space model?
  - classical Linda model (tuples as structured data types)
  - object-oriented tuple space model (tuples as objects and with object fields)
  - logic tuple space model (tuples as logic atoms)
- Which coordination language?
  - basic Linda primitives (in, out, rd, inp, outp)
  - collective operations on tuples (in_all, rd_all …)
  - asynchronous operation (rd_future, notify)

Coordination Laws

- Fixed coordination laws
  - basic pattern-matching (defined fields are primitives data types that, if actual, must have the same value)
  - object-matching (e.g., equality of Java serialised form, tuple sub-classing)
  - unification
- Programmability of the Coordination language
  - primitives overriding (the default behaviour of a primitive can be changed)
  - adding of new primitives
- Programmability of the Coordination media
  - full monitoring of access events (who, what, on which tuple)
  - which behaviour in response to which access event

Tuple-based Coordination: Summary of Systems

	Coordinables	Coordination Media	Coordination Language	Coordination Laws
JavaSpaces	Java Agents	OO referencing	OO (Java) Linda primitives	Fixed (object-matching)
Lime	Abstract mobile components	Implicit access (to agent-own tuple space)	Linda	Fixed (pattern-matching)
MARS	Java mobile agents	Implicit access (to local tuple space)	OO (Java) Linda primitives	Programmable (Java reactions)
PageSpace	Agent + Services	OO referencing	OO (Java) Linda primitives	Fixed (object-matching)
T Spaces	Agent, PDA	OO referencing	Linda + user-defined primitives	Programmable (primitives overriding)
TuCSoN	Heterogeneous (possibly mobile) Internet agents	implicit access (to local tuple space) + URL referencing	Logic-oriented Linda	Programmable (FOL reactions)

JavaSpaces

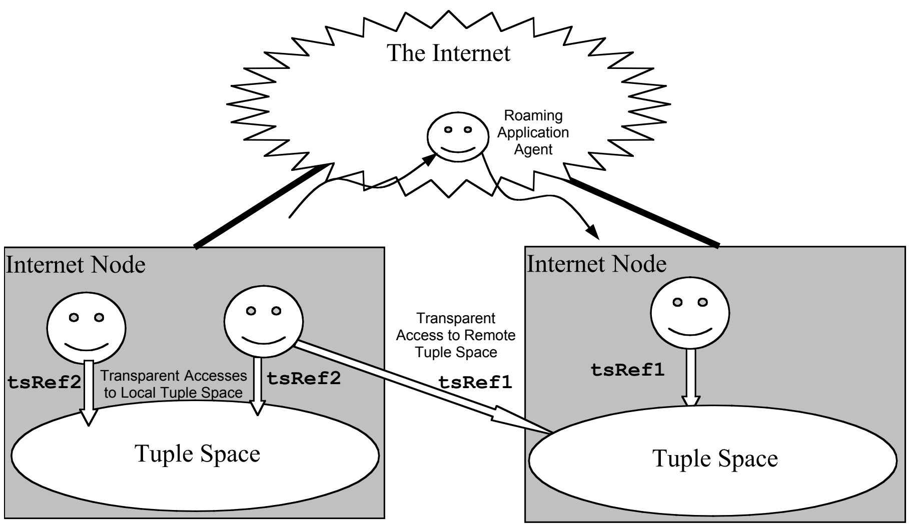
- SUN Microsystems
- Object-oriented tuple space model
- Event notification mechanisms
- Enabling technology for JINI

- Coordinables:
  - not specifically tuned to Internet agents
  - coordinables as generic Java components
  - In JINI, JavaSpaces used for associative access/lookup network data and services by distributed components

JavaSpaces: Coordination Media

- Tuple spaces as Java objects
- Location independence
  - tuple spaces identified and accessed via object-references
  - accesses are network-unaware (no matter where agent is)

- How to retrieve JavaSpaces references?
  - Private application objects
  - Lookup services



JavaSpaces: Coordination Language

- Object-oriented Linda operations
- JavaSpaces interface
  - *`Read (Entry tmp, Tr txn, long lease)`*
  - *`Take (Entry tmp, Tr txn, long lease)`*
  - *`Write (Entry e, Tr txn, long timeout)`*
  - `txn` can specify a transactions (*all-or-nothing* semantics)
  - *`lease`* blocking time for operations
  - *`timeout`* time to live for tuples
- The object tuple *`tmp`*:
  - subclass of *`Entry`*; its instance variable are the tuple fields;

```
SubEntry mytuple = new SubEntry ();
mytuple.field1= O1, mytuple.field2= null, etc…
read(mytuple, null, 0)
```

JavaSpaces: Event Notification

- No programmable coordination laws
  - fixed object-oriented pattern-matching
- Reactive event notification
- Agents can register in a space as interested in a tuple matching a template, and require to be notified about its insertion

  notify(template, null, listener, lease.FOREVER, null)
  - the listener will be sent an event it has to handle when a matching tuple will be inserted in the space
- Can catch a limited set of events!

Case Study: Brief Summary

- Agents visits Internet sites to analyze HTML pages
- Agents clone themselves to follow interesting links

- Avoiding multiple visits
  - use of tuple spaces to store information about previous visits

- Access to local data
  - use tuple spaces to retrieve information on local services, files, and to access to services through the tuple space

Case Study in JavaSpaces

- Inter-agent coordination
  - how can they agree on which space to use for advertise each other of visited sites?
    - single home space? Centralization!
    - distributed spaces? How to look them up?
- When accessing to local resources
  - if there is a local tuple space for local resources
    - the tuple space can be used to access to files
  - otherwise
    - tuple space to be used to look up local services
    - local services accessed in a client-server fashion
- Complex design
  - agents have to explicitly discover tuple spaces and services

JavaSpaces: Sample Code

```
class FileEntry extends AbstractEntry
{ // tuple fields
  public String PathName;  public String Extension;
  public Date ModificationTime; public File ActualFile;

// Tuple constructor. The order of the parameters defines the tuple field order
public FileEntry(String name, String extension, Date modificationTime, File file)
{               PathName = name; Extension = extension;
                ModificationTime = modificationTime; ActualFile = file;      }
}
….
FileEntry FilePattern = new FileEntry(null, “html”, null, null);
// creation of the template tuple
FileEntry HTMLFile = LocalSpace.read(FilePattern, null, NO_WAIT);
// access to the local tuple space referred by the LocalSpace interface
// returns a matching FileEntry tuple, if any
//however, the reference has to be somehow explicitly initialised by the agent

if (HTMLFile != null)     // if a matching tuple is found
                {         // can access the tuple fields
                          String found_name = HTMLFile.PathName;
                          …...
```

T Spaces

- IBM research
- “a network communication buffer with database capabilities”
  - extensible Linda interface
  - a single interface for enabling communication between all entities in a network (middleware)
  - database oriented
- Coordinables
  - all the entities in a network can connect to T Spaces
  - all interactions (e.g., service requests, message-passing, database queries) occur via T Spaces

T Spaces: Coordination Media

- T Spaces servers handles all T Spaces
  - T Spaces lifecycle
  - servers identified via URLs:

```
TupleSpace ts = new
TupleSpace(”newTS",myhost.com)
// ask TS creation to the server on myhost.com
// new TS locally created
```

  - directory service for T Spaces and servers
  - no limits on the number of T Spaces
- For Internet applications:
  - all servers have to maintain information about all other servers in the Internet
  - scalability?

T Spaces: Coordination Language

- Linda-like model and operations

  in/read(int, "Franco", float), out(6,"Andrea",7.3)

- tuple fields are dynamically added to generic tuples

  Tuple t1 = new Tuple()

  t1.add(field1) …

- Collective operations on tuple

  scan, consumingScan, countN

- Database-oriented queries
  - tuple fields can be named and indexed

  Tuple r=ts.scan(new IndexQuery("f",new Integer(8)));

  // return tuples whose field named "f" has value 8

T Spaces: Coordination Laws

- New operations (commands) can be added on a T Space and the behavior of operations can be changed
- Complex management
  - for a *T Space, a TSFactory* can manage several *TSHandler*, each devoted to a new/overridden operation on the space
  - several TSFActory can be stacked for incremental refinement
  - definition and installation of handlers and operations rather complex and verbose
- Event notification
  - all types of event, not only write as in *JavaSpaces*

Case Study in T Spaces

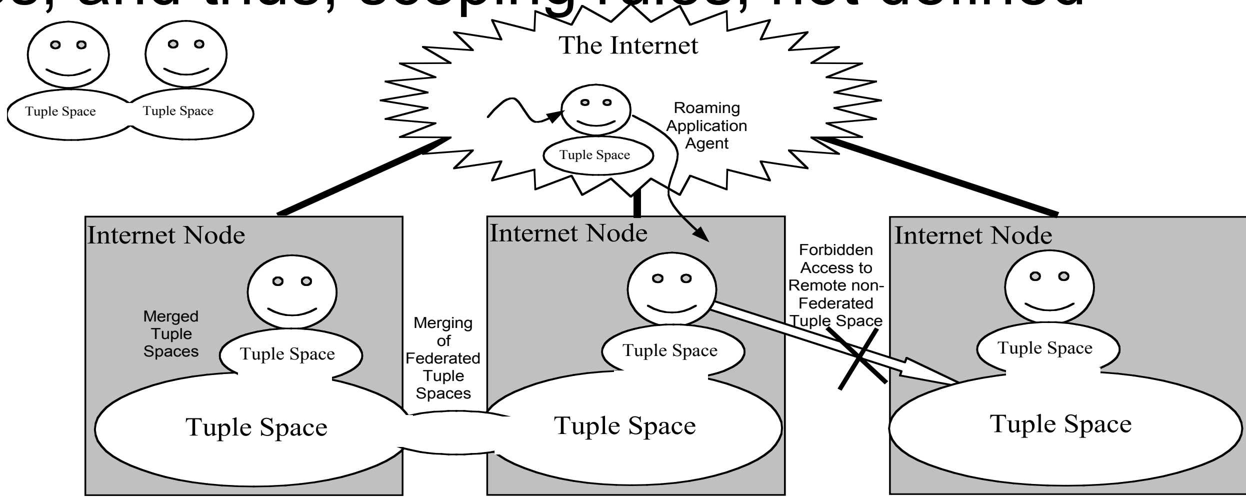
- Inter-agent coordination
  - they can refer to the local T Space server and create a private T Space on the visited nodes to interact
  - they can add handlers on the private T Spaces to handle more complex coordination rules than the simple “avoid multiple visits”
- When accessing to local services
  - a local T Space can host the needed services
  - agents must discover which services (handlers) are available on a local T Space OR
  - they can simply lookup services and then access them

Lime

- Washington University, St. Louis
- currently a model, not an implemented system
- formalised with Mobile UNITY
- Goal
  - unifying interaction model for devices, users, and agents
- Coordinables
  - all mobile entities carry on 1 or several associated tuple spaces
  - host have a local tuple space, possibly federated with other tuple spaces (and becoming a single one)
  - do not assume/require a fixed network infrastructure

Lime: Coordination Media

- Interactions through the local tuple space
- Merging tuple spaces in case of meetings
  - connection in case of mobile devices
  - execution of an agent on a node/federation of nodes
  - only tuple spaces with the same name are shared
- Naming rules, and thus, scoping rules, not defined



Lime: Coordination Language

- Linda-like operations on tuple spaces

  T.in(template), S.out(tuple)

  // T, S names of tuples spaces

  - What would these names be in the Internet?

- Specification of tuple locations:
  - one can specify where a tuple must be located

  T.out[L](tuple)

  // when L merges, the tuple flows to L’s tuple space

  - one can specify from where a tuple must be retrieved

  T.rd[F,T](template)

  // in location F, read tuples whose location is T

Lime: Coordination Laws

- Introduction of a reactive statement

  ```
  S.reactsTo(C,T)
  // execute C when a tuple matching T is found in S
  ```

  – the reactsTo statement register the reaction in S

  – the only event that can trigger a reaction is the introduction of a tuple in the tuple space (i.e., an out operation)

- Reactions can be annotated with locations

  – however, reactions takes place in the local tuple space only

- Weaker reactive mechanism provided

  ```
  T.upon(C,T)
  ```

  – similar to JavaSpaces' notify mechanism

Case Study in Lime

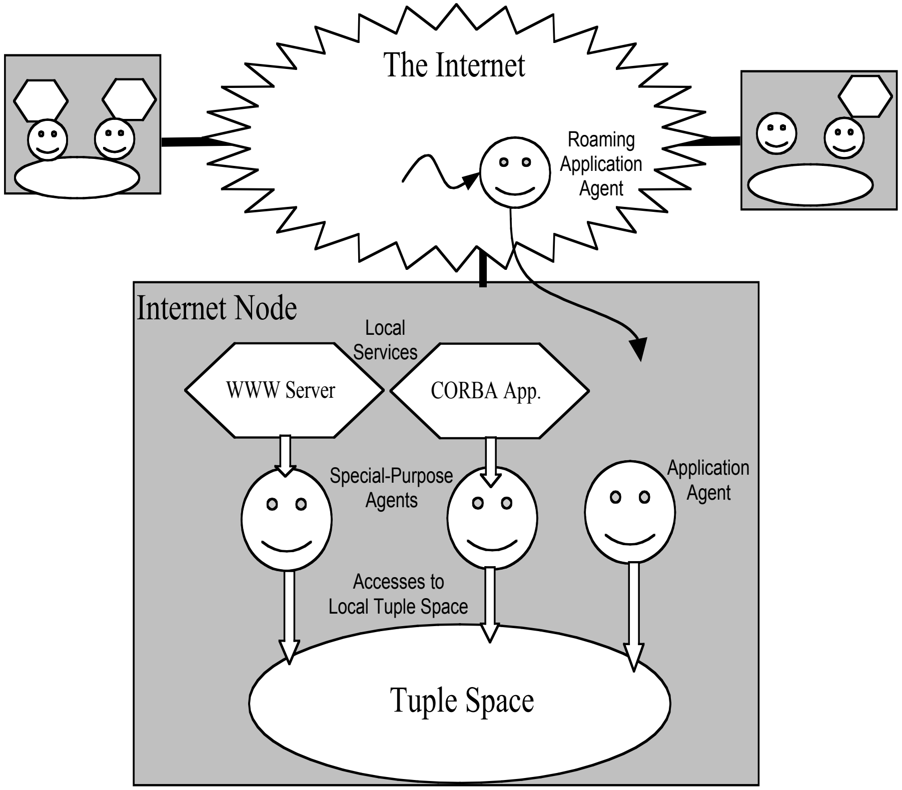
- Inter-agent coordination
  - merging of agent tuple space with the one of the current host
  - possible tuples left by other agents having visited the node flow towards the agent tuple space
  - the agent can check or being notified about this incoming tuples
- Accessing to the resources on a node
  - via the merged tuple space, the agent can interact with a "service agent" on a node, to request services and retrieve results
- Control Problems
  - tuple space merging and tuple locations: global naming?
  - accesses to tuple spaces: global ACLs?

PageSpace

- TU Berlin & University of Bologna
- Tuple-based middleware for coordinating components and agents in the Web scenario
- All the components of the Web scenario are given the possibility of interacting and coordinating
- More structured than T Spaces
  – Definition of different classes of special-purpose agents
  – Reference Architecture for Interactive Web applications

The PageSpace Reference Architecture

- Special-purpose agents with different roles
  - user interface (applet-based) agents
  - homeagents, as persistent representation of users
  - legacy application interface agents
  - control-agents
  - application specific agents



PageSpace: Coordination Media

- First proposal similar to JavaSpaces
  - tuple spaces accessed via object-references in a location unaware fashion
  - no emphasis on mobility and network-awareness
- Evolution
  - global hierarchy of nested tuple spaces
  - global naming of tuple spaces
    space1, space1/space2 ...
  - agents can bind a tuple space reference to a given tuple space:

```
TupleSpace ts = new TupleSpace();
ts.join("Space1/Space2")
```

- Still location unaware

PageSpace: Coordination Language

- The *Jada* coordination language
  - a simple yet effective object-oriented implementation of Linda (similar to JavaSpaces one)
  - Basic Linda operations

    `Tuple t new Tuple("Hello", new Integer(1));`

    `out(t), in(t) read(t)`
  - Additional operation for tuple collections

    `readAll, InAll`

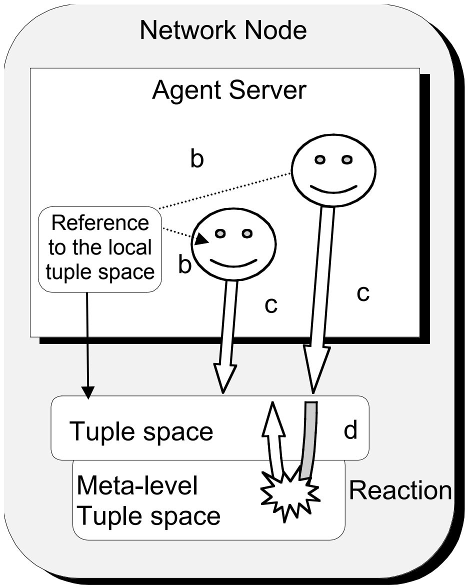
- Coordination laws
  - object-oriented pattern-matching
  - no programmable coordination laws

Case Study in PageSpace

- Inter-agent coordination
  - they can refer to a well-known tuple spaces to avoid multiple visits or
  - can create private tuple spaces on the visited nodes to interact
- When accessing to local services
  - a local tuple space on a node can be used to interact with the needed services on that node
  - agents must discover which services are available on a node and how they have to interact with them

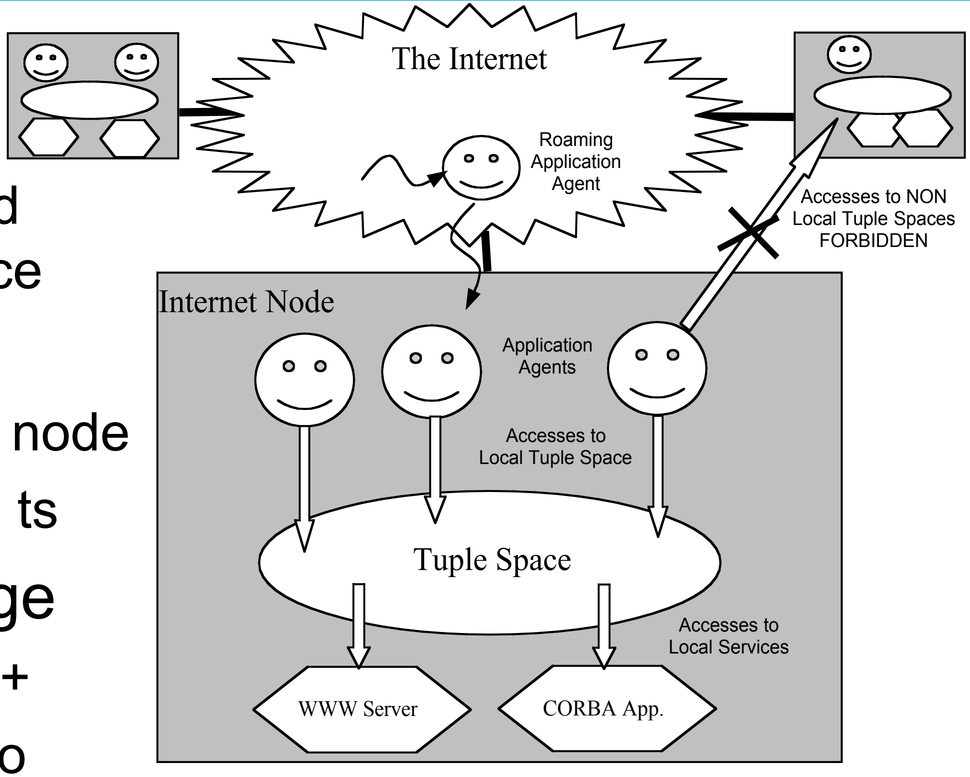
MARS: Mobile Agent Reactive Spaces

- University of Modena, ITALY
- Ported on different agent systems (*Aglets*, *Java2Go*, *SOMA*)
- One tuple space on each node
- Linda-like access to local resources
- Meta-level tuple space for programming tuple space behavior



MARS Features

- Coordinables
  - Java Agents
  - Everything is accessed via the local tuple space
- Coordination media
  - a tuple space on each node
  - implicit access to local ts
- Coordination Language
  - JavaSpaces interface +
  - `readAll`, `takeAll`, to access all matching tuples
- Coordination Laws
  - fully programmable in Java



MARS: Coordination Laws

- Meta-tuples to associate reaction to access events (meta-matching)

  `(Ag-Id, Tuple, OpType, ReactObj)`

- Access the base-level tuple space triggers pattern-matching in the meta-level to look for reactions to execute
- *`ReactObj`*: Java object with a single method (the reaction itself)
- Example

  `(ag@mo.it, null, read, O1)`

  triggers the reaction method of object `O1` when the agent `ag@mo.it` performs a `read` operation

MARS: Security Model

- ACLs on tuples
- Agents associated to *roles*
- *Examples:*
  - reader role: can only read tuples
  - write role: can write and extract tuples
  - manager: can install/deinstall reactions
- Agent-installed reactions
  - affects only those agents of the same applications
- Problem
  - garbage collection of reactions

Case Study in MARS

- Inter-agent coordination
  - automatic binding to the local tuple space
  - the agent can check or being notified for tuples left by other agents to signal a previous visit
  - reactions can monitor agent activities, or collect garbage
- Accessing to the resources on a node
  - associatively retrieve references to HTML files
  - reactions can deal with heterogeneity (e.g. HTM instead of HTML files), deny access to files already accessed by another agent (solving inter-agent coordination)
- Problems
  - garbage collection of reactions

MARS: Sample Code

```
// tuple definition as in JavaSpaces
class FileEntry extends AbstractEntry
{ public String PathName;  public String Extension;
  public Date ModificationTime; public File ActualFile;
  public FileEntry(String name, String extension, Date modificationTime, File file)
  {               PathName = name; Extension = extension;
                  ModificationTime = modificationTime; ActualFile = file;      }   }
….
FileEntry FilePattern = new FileEntry(null, “html”, null, null);
// creation of the template tuple, as in JavaSpaces
Vector HTMLFiles = LocalSpace.readAll(FilePattern, null, NO_WAIT);
// access to the local tuple space. The reference is automatically initialised as the agent
arrives on that node
// returns all matching FileEntry tuple, if any
if (!HTMLFiles.isEmpty())               // if some matching tuples are found
                  { for (int i = 0; i < HTMLFiles.size(); i++)    // for each matching tuple
                  {               FileEntry Hfile = (FileEntry)HTMLFiles.elementAt(i);                //
       Hfile: tuple representing a  single file
                                        if (this.SearchKeyword(keyword, Hfile.ActualFile))
                                                     ……
```

MARS: Examples of Reactions

```
// can be associated to readAll operations on FileEntry templates with HTML extension
//via the meta-level tuple:   (HTML2HTM_instance, (null, “html”, null, null), readAll, null)

class HTML2HTM implements Reactivity
{
public Entry[ ] reaction(Space s, Entry Fe, Operation Op, Identity Id)
// no match already occurred if the site has only htm files
{ Fe.Extension = “htm”;                       // modifies the extension of the required files
  return s.readAll(Fe, null, NO_WAIT);        // read FileEntry tuples with HTM extension
}}
```

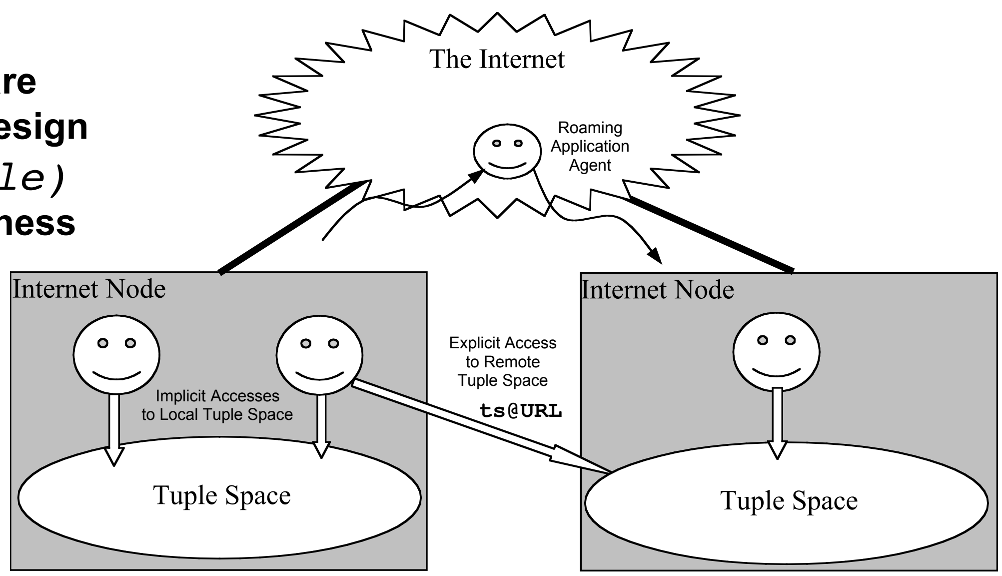
```
// can be associated to take operations via: (TransformTakeObj, null, take, null)

class TransformTake implements Reactivity
{
public Entry reaction(Space s, Entry Fe, Operation Op, Identity Id)
{ if(matched(Fe)) { // if a match has been produced
            if (Id.equals(manager)) // check for the identify of the agent performing the operation
                        return s.take(Fe, null, NO_WAIT);// the tuple is deleted from the spac
            else  {SecurityRegister.add(“takeAll”, Fe, Id);   // log the access
                        return Fe }  // the tuple is returned but it not extracted
else return null;          }            // no match has been produced   }}
```

TuCSoN

- University of Bologna, ITALY
- Implemented in form of a variant model (LuCe)
  - pure Java
  - currently converging to TuCSoN
  - along with coordination technology for a specialised development framework
- Partially formalised
  - ReSpecT tuple centres
  - coordination language
- TuCSoN
  - **Tu**ple **C**entres **S**pread **o**ver **N**etworks
- Multiple local tuple centres as Internet services
  - supporting both network-aware and -unaware coordinables

The TuCSoN Coordination Model

- Multiple Tuple **Centres** in a local coordination space
  - TuCSoN nodes
- Multiple coordination spaces
  - The TuCSoN coordination space: twofold interpretation
    - One global –
    - Multiple local –
  - Local vs. remote interaction
    - *tc?op(tuple)*
      - **network-unaware**
      - **adaptivity by design**
    - *tc@node?op(tuple)*
      - **network-awareness**
- Coordination as an Internet Service



TuCSoN Coordinables and Languages

- Coordinables
  - conceptually, any agent accomplishing the coordination language
  - pragmatically, C, C++, Java, Prolog agents
    - heterogeneity addressed from the very beginning
- Coordination language
  - Linda-like
    - out, in, rd, inp, rdp
- Communication language
  - based on first-order logic
    - uninterpreted symbols
  - logic tuples
    - tuple centres as theories of the communication

TuCSoN Coordination Media

- Tuple centres
- The idea
  - standard tuple space interface
    - **data-driven coordination *(to the agent's perception)***
  - programmable behaviour in response to standard communication events
    - **control-driven coordination *(transparently to agents)***
  - information perception vs. representation
    - **tuple space: ≡**
    - **tuple centre: ≠**
- Programmable behaviour specification
  - through reactions to communication events
  - ReSpecT

ReSpecT

- **Re**action **Spec**ification **T**uples
- First-order Logic Tuples
  - `map(OpName,Event),react(OpName,ReactBody)`
  - `reaction(Operation,ReactBody)`
  - FOL tuples for both communication and coordination
- Tuple Centre composed by
  - Tuple space
  - Specification space
- Execution model
  - reactions as transactions
  - communication events *plus* triggered reactions perceived atomically at the agent level
- Uniform specification protocol
  - `outSpec/1, inSpec/1, rdSpec/1, …`

ReSpecT: Main Predicates

Tuple centre's state access and modification	
`out_(Tuple )`	Succeeds, by putting tuple `Tuple` into the tuple centre
`rd_r(Tuple )`	Succeeds, if a tuple `T` matching `Tuple` is found in the tuple centre, by unifying `T` with `Tuple`; fails otherwise
`in_r(Tuple )`	Succeeds, if a tuple `T` matching `Tuple` is found in the tuple centre, by unifying `T` with `Tuple` and removing `T` from the tuple centre; fails otherwise
`no_r(Tuple )`	Succeeds, if no tuple matching `Tuple` is found in the tuple centre; fails otherwise
Communication event's information access	
`current_tuple(Tuple )`	Succeeds, if `Tuple` matches the tuple involved by the current communication event
`current_agent(Agent)`	Succeeds, if `Agent` matches the identifier of the agent which triggered the current communication event
`current_op(Operation)`	Succeeds, if `Operation` matches the operation which produced the current communication event
`current_tc(Name)`	Succeeds, if `Name` matches the identifier of the tuple centre performing the computation
`Pre`	Succeeds in the pre phase of any operation
`Post`	Succeeds in the post phase of any operation
`Success`	Succeeds in the pre phase of any operation, and in the post phase of any successful operation
`Failure`	Succeeds in the post-phase of any failed operation

TuCSoN: Coordination Laws

- Coordination laws
  - Any computable law into the coordination medium
  - Global rules into global abstraction
    - Coordination policies
    - Mediation services / Ontologies
    - Enforced/refined security policies
  - Global intelligence for a MAS
    - Emerging behaviour
- Coordination laws as (meta-level) tuples
  - Spreading the intelligence of a MAS
  - Self-modifiable MAS

Case Study in TuCSoN

- Access to local resources and inter-agent coordination similar to MARS
- In addition:
  - The agent can access a site from remote (without having to move there) to check if the site has been already visited
  - By using absolute names, the application works even if agents are not mobile
  - The logic model for both tuples and reactions can facilitate the access and the intelligent" analysis of large information sources
- Problems:
  - explicit management of tuple space URLs
  - untyped data

TuCSoN: Reaction Examples

```
% makes a rd of an in on page/3 tuples

reaction(in(page(_,_,_)), ( post,
            current_tuple(page(Name,FileName,Ext)),
            out_r(page(Name,FileName,Ext))
)).
```

```
% safe update of page/3 tuples

reaction(out(update(PageName,OldFileName,NewFileName,OldExt,NewExt)), (
            in_r(update(PageName,OldFileName,NewFileName,OldExt,NewExt)),
            in_r(page(PageName,OldFileName,OldExt)),
            out_r(page(PageName,NewFileName,NewExt))
)).
```

Summarizing: Our Ideal Scenario

- Coordinables
  - all entities must find a place in the coordination architecture –> PageSpace
- Coordination Media
  - location-dependent identification and access to tuple space:
    - MARS, TuCSoN for Internet nodes
    - Lime for mobile users and devices
- Coordination Language
  - Standard interface (JavaSpaces?)
  - Multiple models for tuple spaces: OO (MARS, PageSpace), logic (TuCSoN), DB-oriented (T Spaces)
- Coordination Laws
  - fully programmable (MARS, TuCSoN)

Open Issues

- Security issues
  - Internet asks for secure coordination models
- Economic issue
  - if the Internet will become a global market
- Software engineering tools
  - patterns for Internet-agent composition
- Standardization
  - Java? XML?

Coordination and Security

- Two facets of the same problems
  - Handling security means ruling interactions
  - Coordination means ruling interactions
- Cryptography is simply an enabling mechanism
- Security policies must impose rules on interactions and make them harmless
- A coordination model can help in
  - monitoring of interactions through the coordination media
  - checking for authorisation
  - denying/enabling access to a coordination media
  - denying/enabling access to specific information in a coordination media

Economic Models

- Internet resources no longer for free
  - tragedy of the commons
  - E-commerce of intangible goods
- Models and tools needed for
  - resource control
  - micro-payment
- What about coordination?
  - economic transactions imply interactions
  - can a coordination model support them?
  - what a market-oriented coordination model should be?

Coordination Patterns

- Patterns as high-level abstraction for composition
  - rules for component composition and interaction
- Internet-agent patterns
  - mobility patterns: one-hop, looping, itinerant
  - interaction patterns: master-workers, contract-net, agent trees…
- Problems
  - patterns cannot abstract from the communication infrastructure
    - what kinds of interaction mechanisms are available?
    - what will patterns be if (programmable) tuple spaces are available?

XML Tuple Spaces

- Is Java the right choice for Agents?
  - heavy-weight
  - agent are not self-contained
  - requires programming skills
- XML agents
  - light-weight
  - self-contained
  - require (little more than) document-preparation skills
- XML Interactions?
  - Tuple space models based on XML
  - tuples as XML documents OR
  - XML documents as tuples?

Conclusions

- Advantages of coordination models and architectures for Internet applications:
  - to facilitate an engineered approach to Internet application development
  - to leads to a simple and secure application design
- Several open issues and, of course:
  - need for unifying models and reference architectures
  - need for real-world applications

Selected References

- Introductory to Agents:

J. Bradshaw ed., “Software Agents”, AAAI Press, Menlo Park (CA), 1997.

M. Wooldridge, N. Jennings, “Intelligent Agents: Theory and Practice”, The Knowledge Engineering Review, Vol. 10, No. 2, 1999.

G. Weiss, “Multiagent Systems -- A Modern Approach to Distributed Artificial Intelligence”, The MIT Press, Reading (MA), 1999.

Proceedings of the European Workshops on Modelling Autonomous Agents in a Multi-Agent World (MAMAAW), Lecture Notes in Artificial Intelligence, Springer-Verlag (D).

Proceedings of the International Conferences on Autonomous Agents, ACM Press.

Selected References

- Introductory to Internet Agents and Code Mobility:

J. White, “Mobile Agents”, in J. Bradshaw ed.: Software Agents, AAAI Press, Menlo Park (CA), pp. 437-472, 1997.

A. Fuggetta, G. Picco, G. Vigna, “Understanding Code Mobility”, IEEE Transactions on Software Engineering, Vol. 24, No. 5, pp. 352-361, May 1998.

N. M. Karnik, A. R. Tripathi, “Design Issues in Mobile-Agent Programming Systems”, IEEE Concurrency, Vol. 6, No. 3, pp. 52-61, July-September 1998.

D. Milojicic, F. Douglis, R. Wheeler, eds., “Mobility: Processes Computers and Agents”, ACM Press, Reading (MA), 1999.

D. Chess, C. Harrison, A. Kershenbaum, “Mobile Agents: are They a Good Idea?”, Mobile Object Systems, Lecture Notes in Computer Science, No. 1222, Springer-Verlag (D), pp. 25-45, February 1997.

Proceedings of the 1st and 2nd International Workshop on Mobile Agents, Lecture Notes in Computer Science, Springer-Verlag (D).

Selected References

- Systems based on direct interaction models

D. B. Lange and M. Oshima, "Programming and Deploying Java™ Mobile Agents with Aglets™", Addison-Wesley, ISBN 0-201-32582-9, August 1998.

R. Gray, "Agent Tcl: A flexible and secure mobile-agent system", in M. Diekhans, M. Roseman eds., Proceedings of the fourth Annual Tcl/Tk Workshop (TCL '96), Monterey, California, July 1996.

- System based on meeting(-like) interaction models

H. Peine, "Ara - Agents for Remote Action", in William R. Cockayne and Michael Zyda eds.: Mobile Agents: Explanations and Examples, Manning/Prentice Hall, 1997.

J. E. White, "Telescript Technology Mobile Agents", General Magic White Paper, 1996.

J. Baumann, F. Hohl, K. Rothermel, M. Straßer, "Mole - Concepts of a Mobile Agent System", WWW Journal, Special issue on Applications and Techniques of Web Agents, 1998.

Selected References

- Systems with blackboard-based interaction models

P. Domel, A. Lingnau, O. Drobnik, “Mobile Agent Interaction in Heterogeneous Environment”, Proceedings of the 1st International Workshop on Mobile Agents, Lecture Notes in Computer Science, Springer-Verlag (D), No. 1219, pp. 136-148, April 1997.

L. Cardelli, D. Gordon, “Mobile Ambients”, Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, No. 1378, Springer-Verlag, Berlin (D), pp. 140-155, 1998.

- Agent Communication Languages

T. Finin, et al., “KQML As An Agent Communication Language”, Proceedings of the Third International Conference on Information and Knowledge Management, ACM Press, 1994.

M. P. Singh, "Agent Communication Languages: Rethinking the Principles", IEEE Computer, Vol. 31, No. 12, pp. 40-47, Dec. 1998.

Selected References

Agents and CORBA

D. Milojicic, et al., "MASIF The OMG Mobile Agent System Interoperability Facility", 2nd International Workshop on Mobile Agents, Lecture Notes in Computer Science, vol. 1477, pp. 50-67, Springer-Verlag (D), Sept. 1998.

T. Magedanz et al., “Integrating Mobile Agent Technology and CORBA Middleware”, Agentlink Newsletter, No. 1, Nov. 1998.

Software Engineering with Agents

M. J. Wooldridge and N. R. Jennings, D. Kinny, "A Methodology for Agent-Oriented Analysis and Design, ACM Conference on Autonomous Agents, Seattle (WA), ACM Press, pp. 69-76, June 1999.

M. J. Wooldridge and N. R. Jennings, "Software Engineering with Agents: Pitfalls and Pratfalls", IEEE Internet Computing, Vol.3, No. 3, May-June 1999.

P. Ciancarini, A. Omicini, F. Zambonelli, “Multiagent Systems Engineering: the Coordination Viewpoint”, Proceedings of ATAL ‘99, Lecture Notes in Artificial Intelligence, 1999, to appear

Selected References

- Introductory to Coordination Models and Languages

G. A. Papadopoulos, F. Arbab, "Coordination Models and Languages", Advances in Computer, Vol. 46, Aug. 1998.

D. Gelernter, N. Carriero, "Coordination Languages and Their Significance", Communications of the ACM, Vol. 35, No. 2, pp. 96-107, February 1992.

P. Ciancarini, “Coordination Models and Languages as Software Integrators”, ACM Computing Surveys, Vol. 28, No. 2, pp. 300-302, June 1996.

R. M. Adler, ‘Distributed Coordination Models for Client-Server Computing’, IEEE Computer, Vol. 29, No. 4, pp. 14-22, April 1995.

P. Wegner, “Why Interaction is more Powerful than Computing”, Communications of the ACM, Vol. 40, No. 5, pp. 80-91, May 1997.

Proceedings of the 1st, 2nd and 3rd International Conferences on Coordination, Lecture Notes in Computer Science, Springer-Verlag.

Selected References

- Tuple-based Coordination Models

S. Ahuja, N. Carriero, D. Gelernter, “Linda and Friends”, IEEE Computer, Vol. 19, No. 8, pp. 26-34, August 1986.

N. Carriero, D. Gelernter, "Linda in Context", Communications of the ACM, Vol. 32, No. 4, pp. 444-458, 1989.

D. Gelernter, ‘Multiple Tuple Spaces in Linda’, Proceedings of PARLE ‘89, Springer-Verlag (D), 1989, pp. 20-27.

P. Ciancarini, ‘Distributed Programming with Logic Tuple Spaces’, New Generation Computing, Vol. 12, No. 3, 1994, pp. 251-284.

A. Omicini, "On the Semantics of Tuple-based Coordination Models", Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio (TX), Feb. 1999.

Selected References

- Internet Agent Coordination

G. Cabri, L. Leonardi, F. Zambonelli, “Coordination Models for Internet Applications based on Mobile Agents”, IEEE Computer, 1999, to appear.

P. Ciancarini, A. Omicini, F. Zambonelli, “Coordination Technologies for Internet Agents”, Nordic Journal of Computing, 1999, to appear

P. Ciancarini, A. Omicini, F. Zambonelli, “Coordination Models for Multiagent Systems”, Agentlink Newsletter, No. 3, July 1999

All available at: http://www.dsi.unimo.it/Staff/st35/Zambonelli.idc

Selected References

- JavaSpaces http://chatsubo.javasoft.com

  E. Freeman, S. Hupfer, K. Arnold, “JavaSpaces: Principles, Patterns and Practice”, Addison-Wesley, Reading (MA), 1999

- T Spaces http://www.almaden.ibm.com/cs/Tspaces

  "TSpaces: The Next Wave", 32nd Hawaii International Conference on System Sciences (HICSS-32), January 99

- PageSpace http://flp.cs.tu-berlin.de/pagesp

  P. Ciancarini, et al., “Coordinating Multi-Agents Applications on the WWW: a Reference Architecture”, IEEE Transactions on Software Engineering, Vol. 24, No. 8, May 1998.

- Lime http://www.cs.wustl.edu/

  G. P. Picco, A. L. Murphy, G.-C. Roman, "LIME: Linda Meets Mobility", 21st International Conference on Software Engineering, Los Angeles (CA), ACM Press, pp. 368-377, May 1999.

Selected References

- MARS http://sirio.dsi.unimo.it/MOON

  G. Cabri, L. Leonardi, F. Zambonelli, “MARS: a Programmable Coordination Architectures for Mobile Agents”, IEEE Internet Computing, 2000, to appear.

  G. Cabri, L. Leonardi, F. Zambonelli, “Mobile Agent Coordination for Distributed Network Management”, Journal of Network and Systems Management, 2000, to appear.

- TuCSoN http://www.lia.deis.unibo.it/Research/TuCSoN

  A. Omicini, F. Zambonelli, “Coordination for Internet Application Development”, Journal of Autonomous Agents and Multiagent Systems, Vol. 2, No. 3, Sept. 1999.

  E. Denti, A. Natali, A. Omicini, “On the Expressive Power of a Language for Programming Coordination Media”, Proceedings of the 1998 ACM Symposium on Applied Computing (SAC'98), Atlanta (GA), Feb. 1998. ACM.

Selected References

- Agent Security

Mobile Agents and Security", Lecture Notes in Computer Science, No. 1419, Springer-Verlag, Stuttgart (D), 1998.

M. Cremonini, A. Omicini, F. Zambonelli, "Extending the Scope of Coordination towards Security and Topology", MAMAAW 99, Lecture Notes in Computer Science, No. 1647, Springer-Verlag (D), July 1999.

- Market-oriented Internet computing

J. Bredin, D. Kotz and D. Rus, "Market-based resource control for mobile agents", Proceedings of Autonomous Agents '98, ACM Press, 1998.

A. Gupta et al., "Streamlining the Digital Economy: How to Avert a Tragedy of the Commons", IEEE Internet Computing, Vol. 1, No. 6, Nov.-Dec. 1997.

Selected References

Agent Patterns

Y. Aridor, D. Lange, "Agent Design Patterns: Elements of Agent Application Design", Proceedings of Autonomous Agents '98, ACM Press, 1998.

E. Kendall et al., "Patterns of Intelligent and Mobile Agents", Proceedings of Autonomous Agents '98, ACM Press, 1998.

XML and Agents

D. B. Lange, T. Hill, M. Oshima, "A New Internet Agent Scripting Language Using XML", AAAI-99 Workshop on AI in Electronic Commerce, July 1999

G. Cabri, L. Leonardi, F. Zambonelli, "XML DataSpaces for Mobile Agent Coordination", Technical Report No. DSI-99-13, University of Modena, Sept. 1999.