

A general research and outlook of Multi-Agent Systems

LiXizhi

Submitted in partial fulfillment of the
Web Agent Framework project

The CKC honored school of Graduate studies
of
Zhejiang University

Hangzhou, Harbin, 2003

Zhejiang University

Abstract

A general research and outlook of Multi-Agent Systems

LiXizhi

Chair of the Supervisory Committee:

Professor He Qinming
Department of Computer Science

Multi-Agent System (MAS) is an emerging field of study and application that will constitute an indispensable part in the near future of mankind. It applies Artificial Intelligence (AI) related disciplines and theories to problems (or situations) for more intuitive and prospective solutions. The author has done a general research on Multi-Agent System. The paper contains a comprehensive and up-to-date overview and a conclusive outlook on MAS. It's hoped that this article will be helpful to those who have just decided to pick up, change or choose a subject that targets on Multi-Agent System.

Key words: MAS, AI, overview

Preface

Before I did the research, I am designing and developing a program with the objective to visually connect all intelligent resources on the network. Worries of repeating existing projects and redesigning standards set me into informal researches on the internet. Resemblance brought me into a large network of resources on Multi-Agent System on the internet. As a result I redefined my program's objectives and revised it several times as my research went on. Please refer to [\[Section 8\]](#) for details.

So, although this article is about the general research and overview of MAS, it focuses on (1) the ramification and classification of concurrent research projects worldwide, (2) the choosing of a prospective subject for new researches, and (3) existing standards and important resource references (mainly white papers, websites and forming societies on the internet).

I hope that readers of this paper (including me) could derive from it a personal and non-overlapping outlook on MAS.

p.s. The materials in this article are mostly cited from other people's works. Since there are so many of them, I cannot paraphrase every sentence that is not entirely written by me. But I have tried to include all the citations in the bibliography and reference sections.

Acknowledgements

The author wishes to express sincere appreciation to Professor He Qinming for his vast reserve of patience and knowledge. This thesis would never have been completed without the encouragement and devotion of my family and friends.

Content:

Abstract.....	I
Preface.....	II
1 Introduction.....	2
1.1 Overview of MAS.....	2
1.2 Facts about MAS.....	3
1.3 General outlook.....	4
2 Ramifications and basic architectures.....	6
2.1 Static agent.....	6
2.2 Mobile agent.....	8
3 Classifications by examples.....	10
3.1 Classify by applications.....	10
3.2 Classify by level of AI technology employed.....	10
3.3 Classify by agent cooperating style.....	11
3.4 Classify by project objectives and vista.....	11
3.5 Classify by geography.....	12
4 Standards.....	13
4.1 Agent communication language.....	13
4.1.1 KQML and FIPA ACL.....	14
4.2 XML transport protocol and other protocols.....	14
4.3 CORBA and language tools.....	15
5 Useful resources on the internet.....	16
5.1 International agent societies.....	16
5.2 Some agent-developing platforms.....	16
5.3 Important white papers, people and web references.....	16
6 Chronicles and current emphasis.....	17
6.1 Chronicles.....	17
6.2 Research emphasis.....	17
7 Difficulties.....	18
7.1 Software developing difficulties.....	18
7.2 Standardization (Communication) difficulties.....	18
7.3 Balance between vista and applicability.....	19
7.4 Artificial Intelligent related difficulties.....	19
7.5 Promotional difficulties.....	19
8 Outlooks and introduction to Web Agent Framework.....	21
9 Conclusions.....	22
Glossary.....	23
Bibliography.....	24
List of references.....	25
Appendix A: Projects quick reference.....	26
Appendix B: Introduction to Web Agent Framework (WAF).....	27

1 Introduction

Multi-Agent System (MAS) is an emerging field of study and application that will constitute an indispensable part in the near future of mankind. It applies Artificial Intelligence (AI) related disciplines and theories to problems (or situations) for more intuitive and prospective solutions.

We are already in the age of electronics. As more and more tasks are automatically being done by machines, and more and more electronic devices are being connected, the world is going to experience another impact of technological advancement. The kernel (or brain data) of all those electronic devices is software programs whose development have become a quite industry.

Throughout the years, software engineers have been making the development of software programs more easily and the products more reliable, fast and friendly. Ever since the mechanism of computer was first introduced more than half a century ago, people have pointed out the inevitable trend that software will one day make a machine behave much the same way as human. This is exactly what researchers and scientists have always been working at, but they have met with great difficulties during the process. How our brain functions differently from that of a computer is still not clear, but practitioners cannot wait till everything is clear in order to use the already revealed corner of our brain's functioning style. Agent is a compromise drawn by those people. In order for agent to be ready for use in real situations, they define it in very simple ways. Like:

*An agent is a computer system that is capable of **independent** action on behalf of its user or owner. A multi-agent system is one that consists of a number of agents, which **interact** with one-another.*

And the result is fruitful. They have evolved the progress of programming from sub-routine, procedures and functions, abstract data type, objects to agent. And they have successfully applied it to serious situations like NASA's spacecraft control system, banking system, electronic market system, etc. and more casual but ubiquitous places like Blackboard system, online auction system, human society simulation, personal assistants, etc.

In conclusion, MAS is not only the trend in software development, but also the driving force in the formation of the network of future's intelligent devices.

1.1 Overview of MAS

The more specific and widely accepted way of defining an agent is this:

An intelligent agent is a computer system capable of flexible autonomous action in some environment.

By flexible, we mean:

- reactive;
- pro-active;
- social.

For further explanation for each term, please refer to [1].

A Multi-Agent System (MAS) contains a number of agents, which

- (1) interact through communication,
- (2) are able to act in an environment,

- (3) have different “spheres of influence” (which may coincide),
- (4) will be linked by other (organizational) relationships.

For all agent systems from different domains to communicate with each other, we will need an Agent Communication Language or ACL and a shared speech-act pairs or ontology.

To design a multi-agent system, we need an agent architecture, which defines how the agent processes input messages and generates rational behaviors. Again there are a number of ways to do this. It may be one of the three types listed below:

- symbolic/logical;
- reactive;
- hybrid.

None of them is the type that Artificial Intelligent researchers do in labs. However, they can produce useful agents on a typical computer.

International agent communities have been sharing the progress, development tools as well as common knowledge for a long time. They also published standards on any possible parts of MAS like FIPA ACL, communication protocols, agent language tools, etc, hoping that researchers worldwide could conform to it, as that they don't need to redesign many common specifications and their agent based system can have some common knowledge about its neighbors. So it's important to know something about such communities and their published works before starting one's own project on MAS.

In the chapters that follow, I will make references and exemplify on the areas of discussion I mentioned above.

1.2 Facts about MAS

Facts about agents^[1]:

- Agents are not just objects: Agents are autonomous, smart and active.
- Agents are not expert system: Agents are situated in an environment and they take actions; some expert systems do have agent interface.
- Agents and Artificial Intelligence (AI) have different objectives:
AI aims to build systems that can (ultimately) understand natural language, recognize and understand scenes, use common sense, think creatively, etc — all of which are very hard; when building an agent, we simply want a system that can choose the right action to perform, typically in a limited domain. We *do not* have to solve *all* the problems of AI to build a useful agent.

Facts about MAS:

- MAS is harder than distributed system (DS): Building a MAS has all the difficulties of building a DS, which is the most complex one in computer programming, since a MAS is in essence a distributed system but the emulation of agent makes it a lot more sophisticated than just object components.
- Not every program could be called an agent, and not every system a MAS: there is no fine line between an agent and anything else. But you call anything an agent only when you deliberately attribute to it some characteristics that are unique to an agent.

- If you can't do it with ordinary software, you probably can't do it with agents.
- There is no evidence that any system developed using agent technology could not have been built just as easily using non-agent techniques.
- Agents *may* make it easier to solve certain classes of problems but they do not make the impossible possible.
- Agents are not AI by a back door.
- Don't equate agents and AI.
- Agents have been used in a wide range of applications, but they are not a universal solution.
- For many applications, conventional software paradigms (e.g., OO) are more appropriate.
- Given a problem for which an agent and a non-agent approach appear equally good, prefer non-agent solution!
- Process of scaling up from single-machine multi-threaded Java app to multi-user system *much* harder than it appears.
- Developing *any* agent system is essentially experimentation.
- No tried and trusted techniques
- This encourages developers to forget they are developing *software*!
- Mundane software engineering (requirements analysis, specification, design, verification, testing) is forgotten.
- Result a foregone conclusion: project flounders, not because agent problems, but because basic software engineering ignored. Frequent justification: software engineering for agent systems is none-existent.
- In any agent system, percentage of the system that is agent-specific is comparatively small.
- Therefore important that conventional technologies and techniques are exploited wherever possible.
- Architecture development takes years; Different architectures good for different problems.
- Any architecture that is truly generic is by definition not architecture.
- Build agents with a minimum of AI; as success is obtained with such systems, progressively evolve them into richer systems like what Etzioni calls "useful first" strategy.
- Be realistic: it is becoming common to find everyday distributed systems referred to as multi-agent systems.
- Agents don't have to be complex to generate complex behaviors.
- There are no widely-used software platforms for developing agent systems.

1.3 General outlook

Intelligent agents are usefully applied in domains where *flexible* autonomous action is required. This is not an unusual requirement! Agent technology gives us a way to build systems that mainstream software engineering regards as hard!

Main application areas:

- distributed/concurrent systems;
- networks;
- human-computer interfaces.

The function of multi-agent system can be seen as a transitional revolution that evolves common software to a network of intelligent agents that embedded in almost every electronic device. Those agents communicate with each other as well as humans. It is leading us to a true electronic age, where we (humans) are not the only species that communicate. We will receive many services from agent, and we are going to think of them more or less the same way as our own kind.

2 Ramifications and basic architectures

There are two types of Multi-agent system whose architecture differs substantially from one another. They are (1) static agent system where the agent remains in the same location throughout its life, and (2) mobile agent system where the agent travels from place to place. Since mobile agents are usually simple interface agent, the following architecture applies only to static agent system. For more information please see Section 2.1 and 2.2.

There are basically three types of *agent architecture*¹:

- symbolic/logical;
- reactive;
- hybrid.

Originally (1956-1985), pretty much all agents designed within AI were *symbolic reasoning* agents. Its purest expression proposes that agents use *explicit logical reasoning* in order to decide what to do. Problems with symbolic reasoning led to a reaction against this — the so-called *reactive agents* movement, 1985–present. From 1990-present, a number of alternatives proposed: *hybrid* architectures, which attempt to combine the best of reasoning and reactive architectures [1].

2.1 Static agent

Static agents are agents that remain in the same location throughout their life. They communicate with other agents and take locally authorized actions. I will be talking about mostly this type of agent system in this article.

There are three types of *agent architecture*: symbolic/logical, reactive and hybrid. Please refer to [1] for more details.

Symbolic/logical architecture:

The classical approach to building agents is to view them as a particular type of knowledge-based system, and bring all the associated methodologies of such systems to make it. This paradigm is known as *symbolic AI*. We define a deliberative agent or agent architecture to be one that:

- contains an explicitly represented, symbolic model of the world.
- makes decisions (for example about what actions to perform) via symbolic reasoning.

If we aim to build an agent in this way, there are two key problems to be solved:

¹ Maes defines an agent architecture as:

‘A particular methodology for building [agents]. It specifies how . . . the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions . . . and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology.’

Kaelbling considers an agent architecture to be:

‘A specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules. A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks.’

1. translating the real world into an accurate, adequate symbolic description, in time for that description to be useful for vision, speech understanding and learning.
2. how to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful.

In other words, we need to deal with knowledge representation, automated reasoning and planning.

Symbolic architecture is well studied and it's not even possible to list the descriptions of its branches. Some references are listed below for readers to check out themselves.

- *Agent oriented programming*. Some instances are AGENT0² and Planning Communicating Agents (PLACA)³ language.
- Concurrent METATEM⁴
- Means-Ends Reasoning
- BDI (Believe-Desire-Intention) architecture. One instance is Vere & Bickmore developed HOMER: a simulated robot submarine, in a two-dimensional 'Seaworld'.

Reactive architecture:

There are many unsolved (some would say insoluble) problems associated with symbolic AI. Among those problems, some can be solved very easily through reactive approaches. Reactive agents do the reasoning *off line*, at *compile time*, rather than *online at run time*.

The idea is first given by Brooks who has put forward three theses:

1. Intelligent behavior can be generated *without* explicit representations of the kind that symbolic AI proposes.
2. Intelligent behavior can be generated *without* explicit abstract reasoning of the kind that symbolic AI proposes.
3. Intelligence is an *emergent* property of certain complex systems.

He identifies two key ideas that have informed his research:

1. Situated-ness and embodiment: 'Real' intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems.
2. Intelligence and emergence: 'Intelligent' behavior arises as a result of an agent's interaction with its environment. Also, intelligence is 'in the eye of the beholder'; it is not an innate, isolated property.

The drawback of reactive agent is that the more expressive the agent specification language, the harder it is to compile it.

² the first *Agent oriented programming (AOP)* language

³ Planning Communicating Agents (PLACA) language was intended to address one severe drawback to AGENT0: the inability of agents to plan, and communicate requests for action via high-level goals.

⁴ Concurrent METATEM is a multi-agent language in which each agent is programmed by giving it a *temporal logic* specification of the behavior it should exhibit.

Hybrid architecture:

Many researchers have argued that neither a completely deliberative nor completely reactive approach is suitable for building agents. They have suggested using *hybrid* systems, which attempt to marry classical and alternative approaches. An obvious approach is to build an agent out of two (or more) subsystems:

- a *deliberative* one, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI.
- a *reactive* one, which is capable of reacting to events without complex reasoning.

One example is The TOURINGMACHINES architecture⁵.

2.2 Mobile agent

Mobile agent is agent that actually duplicates itself on the network. As you can see in Fig 2.1 that the top advantage of using mobile agent is the save of round-offs and bandwidth.

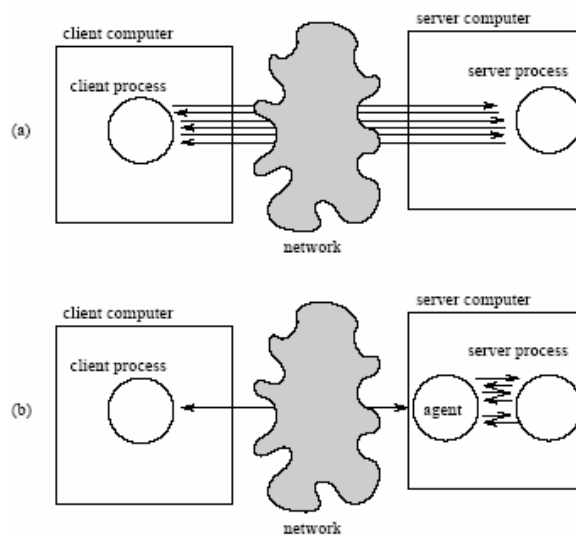


Fig 2.1: Remote procedure calls (a) versus mobile agents (b)
figure taken from [1]

Why we need mobile agents?

- low-bandwidth networks (as in the hand-held PDAs, such as NEWTON)
- efficient use of network resources.

There are *many* issues that need to be addressed when building software tools that support mobile agents. They are:

- security for hosts and agents
- heterogeneity of hosts;
- dynamic linking.

⁵ The TOURINGMACHINES architecture consists of *perception* and *action* subsystems, which interface directly with the agent's environment, and three *control layers*, embedded in a *control framework*, which mediates between the layers.

We can divide mobile agents into at least three types:

(1) *autonomous*

By *autonomous* mobile, we mean agents that are able to *decide for themselves* where to go, when, and what to do when they get there (subject to certain resource constraints, e.g., how much ‘e-money’ they can spend. Such agents are generally programmed in a special language that provides a go instruction... best known example is TELESRIPT⁶.

(2) *on-demand*

The idea here is that a host is only required to execute an agent when it explicitly demands the agent. The best known example of such functionality is that provided by the JAVA language, as embedded within `html`.

(3) ‘*active mail*’-type

The idea here is to pack agent programs onto mail. The best-known example of this work is the *mime* extension to email, allowing Safe-Tcl scripts to be sent. When email is received, the ‘agent’ is unpacked, and the script executed. . . hence the email is no longer passive, but *active*.

⁶ TELESRIPT was a language-based environment for constructing mobile agent systems. TELESRIPT technology is the name given by General Magic to a family of concepts and techniques they have developed to underpin their products.

3 Classifications by examples

In this section, I will list some contemporary MAS projects with their objectives. I will mainly focus on the different classification disciplines. A comprehensive list of MAS projects worldwide can be found at www.agent.org⁷ which contains an agent-project database for free access.

3.1 Classify by applications

This is the common way to classify multi-agent system. I cite it directly from [1].

Main application areas:

- distributed/concurrent systems;

In this area, the idea of an agent is seen as a natural metaphor, and a development of the idea of concurrent object programming.

Example domains:

- air traffic control (Sydney airport);
- business process management;
- power systems management;
- distributed sensing;
- Space shuttle fault diagnosis;
- factory process control.

- networks;

There is currently a lot of interest in *mobile* agents that can move themselves around a network (e.g., the Internet) operating on a user's behalf. This kind of functionality is achieved in the TELESCRIPT language developed by General Magic, Inc, for *remote programming*.

Applications include:

- hand-held PDAs with limited bandwidth;
- information gathering.

- human-computer interfaces.

The idea is to move away from the *direct manipulation* paradigm that has dominated for so long. Agents sit 'over' applications, watching, learning, and eventually doing things without being told — taking the initiative.

Pioneering work at MIT Media Lab (Pattie Maes):

- news reader;
- web browsers;
- mail readers.

3.2 Classify by level of AI technology employed

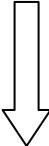
There are in theory agent systems that employ a great deal of Artificial Intelligence techniques like extensive symbolic reasoning, and those neuron-based simulations

⁷ Largest community on cooperating agents formed in Switzerland.

done by the Connectionists. Such kind of software program is usually referred to as Strong AI. But they are not the usual types we see in multi-agent systems.

Currently, almost in any agent system, percentage of the system that is agent-specific is comparatively small. As Etzioni put it: “apply “useful first” strategy”. Agents don’t have to be complex to generate complex behaviors. One should try to build agents with a minimum of AI, as success is obtained with such systems through progressively evolving them into richer systems.

It is becoming common to find everyday distributed systems referred to as multi-agent systems. And we can always classify MAS by the level of AI it uses. Generally they can be ordered like below from weak AI to strong AI.

	Type	Examples
Weak AI  Strong AI	self-interested agent, non-cooperating agent	MAXIMS ⁸
	Implement only Agent Communication Interface	WAF (Please see section 8)
	Using reactive architecture	Bargain Finder from Andersen; Jango from NETBOT
	Using symbolic architecture or an agent language tool that implements it internally	Digital city (Q language ⁹) Expert system and other knowledge services
	Using a hybrid architecture; agents cooperates rather than self-interested	
	A general purpose agent; agent that implements Strong AI.	

The above table only tells vaguely about the use of AI technology. For example a single agent may be very ‘Intelligent’ internally.

3.3 Classify by agent cooperating style

Some agent only communicates with its one user like those email and web browsing assistants.

Some agents are self-interested agent.

Some agents have capabilities of negotiation and argumentation. They will reach a mutually beneficial agreement on matters of common interests. Examples are the On-line Auction agents.

3.4 Classify by project objectives and vista

The objective each research group or corporation set for their project varies, yet the mainstream generally falls into the following categories:

⁸ Pattie Maes developed MAXIMS – best known email assistant: ‘learns to prioritize, delete, forward, sort, and archive mail messages on behalf of a user’.

⁹ Q is a language for describing interaction between agents and users based on agent external roles.

Description	Examples
Personal assistant human-computer interface	Web browsing assistant Expert system
Network services	Many E-commerce applications
Societies simulation	Information Cities ¹⁰
Research and platform development	Q language; Human-centered Semantic web; emorphia ¹¹
Agent communities	www.agentlink.com ; WAF

3.5 Classify by geography

There are nearly a hundred MAS projects going on round the world. Many are from University Institutions in Europe, North America and Japan.

I have observed the following geographical characters, but they are of course not absolute and a good MAS system may arise from any place.

- *Many agent communities are formed and standards are defined in Europe.
- *Many solid applications of MAS are developed in the US.
- *Many imaginative projects are being carried out in Japan.

¹⁰ Information Cities: The Information Cities project models the aggregation and segregation patterns in a virtual world of info-habitants (humans, virtual firms, on-line communities and software agents acting on their behalf). The objective is to capture aggregate patterns of virtual organization, emerging from the interaction over the emerging information infrastructure, a virtual place where millions (or billions) meet of info-habitants meet, cooperate and trade.

¹¹ This is a JAVA based tools on FIPA ACL platform support

4 Standards

Generally there are not many well-formed standards in developing multi-agent system. But it doesn't mean there is none. The most important standards defined so far is on the Agent Communication Language or ACLs.

4.1 Agent communication language

Many agents together in a community will form a society. Just like a real life society with humans, the need for a common medium for communication is essential for the agents to reason or co-operate with each other. The rise in popularity of agent based systems and greater demand for interoperability of agents have led to the need for a language that can be used not just in a proprietary domain, but inter domain, i.e., between different vendors over a network or internet. This will be the focus of this article, the wonderful world of Agent Communication Language or ACLs^[2].

Agent communication protocols or languages provide agents with a means of exchanging information and knowledge, which is really the essence of all forms of interaction in multi-agent systems. Such a protocol or language can be divided into three major components or layers, an "inner" and an "outer" language and its vocabulary.

The "vocabulary" or terminology is known as the ontology. This layer ensures that a term and indeed, even an object or entity, will have a uniform meaning amongst all agents involved in interaction even if different names are used for them. In short, ontology semantically unifies agent communication.

In short, we can say that the ACL is the medium through which the intention regarding the content of the exchange between agents is communicated. Using such performatives as assert, request or query, with regards to content specified with the inner language.

Here is a snapshot:

```
(inform
:sender agent1
:receiver agent5
:content (price good200 150)
:language sl
:ontology hpl-auction
)
```

Some better known examples are the DARPA initiated KSE and FIPA's effort.

The KSE was set up with the aim of developing techniques and software tools to allow for efficient communication and co-ordination between agents that can be re-used and eventually become the common tool for all agents based systems.

The KSE concerned itself with the development of each of the three layers mentioned. The Interlingua group developed an inner language, that of Knowledge Interchange Format (KIF). This serves as a common language for expressing the content of a knowledge base. Below is a table of acronyms for agent communication language

4.1 Table of acronyms for agent communication language

Acronym	Full Form
ACL	Agent Communication Language
DARPA	Defense Advanced Research Projects Agency
KSE	Knowledge Sharing Effort
FIPA	Foundation for Intelligent Physical Agent
KQML	Knowledge Query Manipulation Language

4.1.1 KQML and FIPA ACL

Although the two ACLs appear to be in competition with each other, we believe that they are not actually in conflict with one another, since they are all based on the “Speech Act” theory and essentially the same in concept. They bear the same idea, and are bringing the technique and technology of ACL towards the same direction.

KQML is a message format that describes the structure of a message that is passed between agents in a run-time knowledge sharing system. It also provides a library of open-ended primitives or performatives that describe loosely the permissible actions/operations that agents may attempt in communicating with one another.

The FIPA is a non-profit organization that was started to encourage and promote agent-based applications, services and equipment. FIPA is supported by a huge list of major industrial giants, such as NEC, Alcatel, NHK and Siemens. It consists of technical committees assigned to topical as well as long standing issues regarding agent-based systems. One of which is charged with the responsibility with developing an ACL. The result of which, was the FIPA ACL. Which is an outer language that specifies message format and include descriptions of their pragmatics that is the communicative acts or intentions of the agents?

Despite many industry and non-developers adopting some variant of KQML, systems using different dialect of KQML still cannot inter-operate. There still lacks a universally agreed upon semantics foundation. However, multi-agent based systems research is still immature, and further efforts by both FIPA and KSE are hoped to solve the impending issues. Furthermore, the development of KQML have played an important role in describing what an ACL is and what it should entail. ^[2]

4.2 XML transport protocol and other protocols

FIPA does not mandate the usage of one particular content language, but instead allows applications choosing an 'appropriate' one. There is however one important requirement: the language should be able to express 'actions', 'statements' and 'objects'. Languages such as KIF have built-in support for expressing those concepts, but are

only popular in closed academic circles. When using XML as content language, the meaning of the elements can be mapped into those concepts.

The Resource Description Framework (RDF) defines a mechanism for describing (Web) resources, 'to enable automated processing of these resources'. It provides both a model for representing these meta-data and proposes XML as serialization syntax. Using RDF Schema a meta-model of the RDF data model can be defined. The combination of RDF and RDFS allow the description and modeling of different concepts with entity-relationship graphs and are clearly well suited as software agent languages.^[3]

Other protocols (not all) include: ^[5]

- Enterprise message systems such as those from IBM and Tibco
- A Java Messaging System (JMS) service provider, such as Fiorano
- CORBA IIOP used as a simple byte stream,
- Remote method invocation, using Java RMI or a CORBA-based interface
- SMTP email using MIME encoding
- XML over HTTP
- Wireless Access Protocol
- Microsoft Named Pipes

4.3 CORBA and language tools

There are some popular agent development environment and design patterns. If you choose to work with them, you will gain some benefits of starting something on something. Such things include CORBA, JAVA, etc. .Net Framework will also be suitable for developing MAS. But for its relatively new debut, not much source code is available in it.

5 Useful resources on the internet

There are plenty of useful resources on the internet, most of which are open source. They are International agent societies, ACL specifications, platform and language tools, useful articles, white papers and a great many institutional or personal websites, etc.

5.1 International agent societies

There are more to be located in the Reference section.

Largest community on cooperating agents formed in Switzerland
<http://www.agentcities.org/>

FIPA official site
<http://www.fipa.org/>

Open source
<http://www.osdn.com/>

5.2 Some agent-developing platforms

Please refer to Reference section.

5.3 Important white papers, people and web references

Please refer to Reference section.

6 Chronicles and current emphasis

Multi-agent system is a new field of study. Although some pioneering Institutions have started relative researches as early as 1980s, the spawning of agent societies are just a few years before in late 1990s. In 1997 and 1998, FIPA^[5] issued a series of agent system specifications that had as their goal inter-operable agent systems. This work included specifications for agent infrastructure and agent applications. The infrastructure specifications included an agent communication language, agent services, and supporting management ontologies. There were also a number of application domains specified, such as personal travel assistance and network management and provisioning. At the heart FIPA's model for agent systems is agent communication, where agents can pass semantically meaningful messages to one another in order to accomplish the tasks required by the application. In 1998 and 1999 it became clear that it would be useful to support variations in those messages. A lot of agent-based systems are started during this time.

6.1 Chronicles

[Please refer to one of the on-line chronicles and the Reference section]

6.2 Research emphasis

Since most of the projects on MAS are at early stage, there is no field of study that has proven to be impractical. Each research group has reason to declare that their project to be useful and prospective.

To achieve the common goals of multi-agent system is the emphasis of all MAS-related projects. So the job done by communities such as FIPA gets attention from every MAS researcher. Those published specifications and standards on MAS as is the job of FIPA not only track the most recent advances in this field, but also serve as an idea subscribing platform. So please refer to FIPA's on-line resources for the most up-to-date research products.

Besides those common works, there is no predominant emphasis on MAS. All categories of MAS as exemplified in the previous sections are receiving comparable emphasis which depends in large to people's personal interests and imaginations.

7 Difficulties

The difficulty of building MAS is obvious. The effort of realizing abstract agent architecture is far more difficult than building just a common distributed system regardless of what development tools you need. The following figure taken from FIPASC00001L[5] shows realizations using a shared element realization.

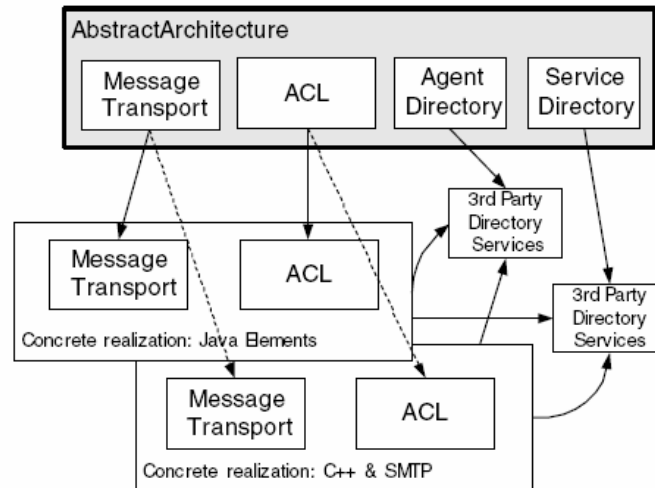


Fig 7: Concrete Realizations Using a Shared Element Realization

7.1 Software developing difficulties

There are just too many non-agent-specific works to be done before we can make even the simplest MAS to function. The lack of agent developing platform adds to this overhead job.

7.2 Standardization (Communication) difficulties

The following figure is taken from FIPASC00001L. A general communication standard can be very hard to implement. They need to identify an agent (misidentifying may lead to security problems), locate it through one of the available means and process the message content that may be in one of the available ACLs.

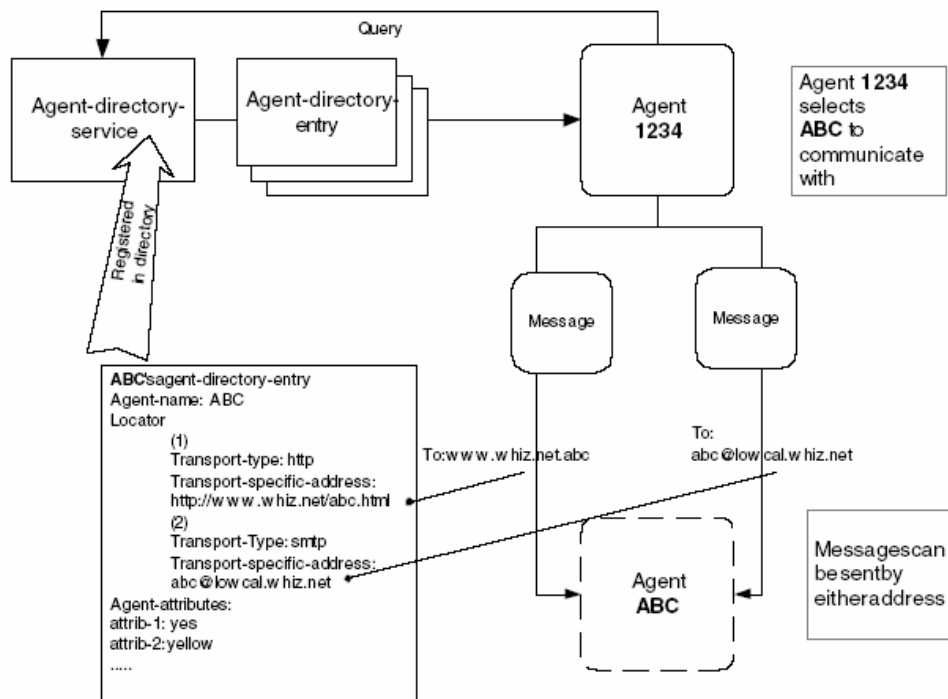


Fig 7.2: Communicating Using Any Transport

7.3 Balance between vista and applicability

Since human beings are general, we hope that agents are also general, in other words, eligible of every task, but its developing complexity grows exponentially when generality extends.

Confronted with this dilemma, MAS developers need to make a compromise between vista and applicability.

Usually architecture development takes years; Different architectures are good for different problems. Any architecture that is truly generic is by definition not architecture. Most experienced developers would probably advise one to build agents with a minimum of AI.

7.4 Artificial Intelligent related difficulties

This is needless to say. The most impending difficulty of MAS is on its brain kernel-how it is going to develop its reasoning. This is not only the job of MAS developers, but also jobs of scientists from a great many other disciplines.

Another problem that is unique for MAS in this category is the agent relationship maintenance. Relationships among agents usually wind up with authority and function that each agent has, thereby lead to the sensible problem of system security.

7.5 Promotional difficulties

Another difficulty is to convince users, company executives, co-workers in a research group that a certain multi-agent system is useful. Project initiator need to exploit and exemplify all the virtues of the system before others can decide to investigate in it.

Developing *any* agent system is essentially experimentation. There are no tried and trusted techniques. This encourages developers to forget they are developing *software*! Mundane software engineering (requirements analysis, specification, design, verification, testing) is forgotten. Projects experience frequent redefinitions during developing stage since the world of agent systems is changing fast.

Please refer to [[Appendix B](#)] for an example.

8 Outlooks and introduction to Web Agent Framework

Web Agent Framework (WAF) is initiated by me and developed by me jointly with a couple of colleagues of mine. The web agent program is to populate the agent society on the web with this specially-designed agent for civilian info-management and service providing. It simulates relationships between real human beings. The relationship could be browsed in a tree-like manner graphically, which is the exploratory way of accessing the agent network.

To be brief, WAF is live intelligent agents on the web (including any computing devices) that could perform simple tasks and reply to anyone (human or other agents) in their request.

Usually, there is one **human master** that is the owner of an agent. And the agent is said to represent this human on the web.

Agent can be created for both **serious and casual usage**. It maintains (1) **relationships** with other master-certificated agents, (2) any kind of information that the master told him, (3) any kind of information which the agent himself collected or recorded that might be helpful to the master. Another fascinating feature is that the agent could safely communicate with other agents or just a human visitor through '**speech**'. If asked, they can give them information that they are allowed to have.

9 Conclusions

The development of multi-agent system is no easy task. But the prospect is enticing. One should be familiar with the agent resources on the internet, be prepared for the potential difficulties and take measures beforehand, all of which are the main purpose of this article.

Glossary

Please refer to one of the on-line glossary service on MAS.

Glossary list without explanation:

CORBA

Multi-agent system or MAS

XML protocol

Speech-Act theory

Agent Communication Language or ACL,

FIPA

KQML

Reactive agent

Pro-activeness

Social agent

Hybrid architecture

Distributed system

Strong/weak AI

AI

Mobile/static agent

PDA

Human-computer interface or HCI

JAVA

.Net Framework

Web services

Bibliography

- [1] Michael Wooldridge. An Introduction to MultiAgent Systems. Published by John Wiley & Sons, 2002.
<http://www.csc.liv.ac.uk/~mjw/pubs/imas/>
- [2] Haw Siang Hon. Agent Communication Language. ISE. 2001.
- [3] Bart Bauwens. XML-based Agent Communication: VPN Provisioning as a Case study.
www.alcatel.com
- [4] LiXizhi. White Paper: Web Agent Framework. 2003
- [5] www.fipa.org. FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents:SC00001L. 2002

List of references

[The Agent Society](#)

[Agents Group @ MIT Media Lab](#)

[Agent Collaboration Language Project](#)

[KQML: The Knowledge Query and Manipulation Language](#)

[Ontolingua: A Tool for Developing Ontologies](#)

[IBM Web Browser Intelligence](#)

[Extensible Markup Language \(XML\)](#)

Appendix A: Projects quick reference

[General Magic, Inc. \(Telescript manufacturers\)](#)

[Autonomy](#)

[BargainFinder from Andersen \(?\)](#)

[Cipher HomePage](#)

[Highlights - client-side agent from Tierra Communications](#)

[IBM Web Browser Intelligence](#)

[InTEXT - WWW search tools](#)

[Intraspect - Knowledge management systems](#)

[Jango from Netbot](#)

[NetscapeWorld - Agents](#)

[Newbot from Wired](#)

[Searchbots from UMass](#)

[Secret Agent from Ariel](#)

[Taxi - internet assistant](#)

[TechTools -- Offline Web Agents](#)

[WebCompass from Quarterdeck](#)

[WiseWire Corporation](#)

Appendix B: Introduction to Web Agent Framework (WAF)

[note:] This short article is also written by the author and serves as a sample scenario description of a general purpose MAS project.

The web agent program is to build, over the network, distributed automatic computer agents that are to be regarded as individuals for every possible use, and for no one in particular. In the first stage of the program, each agent is associated with a master, and is regarded as representative of its master on the web. All agents share some common functions so that they can communicate and form relationship with each other. In the next stage(which is not to be carried out in the present program), web agents are diversified due to the extensive use of them in the form described in stage one, and some of them will even break away from the restrictions set initially.

The efficacy of this program lies in the first stage which deals with issues like:

1. How agents could find jobs and what kind of them on the web that can be regarded as individuals.
2. How human will gain extra benefit by using these agents.
3. How agents should debuting themselves and reproduce themselves by being a networked community on the web and by being more and more useful to human users.

In the program(the stage one), we will (1) develop agents with common functions and write some customer software, (2) assign them with jobs on the web and PCs,(3) illustrate a couple of applications where such agents can do good jobs.

Here are some **short examples** of its applications to give you an imaginable experience.

1. Web service booking system
Most web service discovery sites and applications use only search method with key words and categorization information. There is no relationship between two successive searches. With web agent, you can not only search but browse dynamically through web service providers by their relationships. Your browse track is always available to you as tree graphs with each provider as a node. On the provider side, UDDI specification includes description of relationships between service providers, but most publishers usually leave this field blank. With web agent, relationship information will be kept and maintained by the agent, and is utilized to create dynamic browse functionality for visitors.
2. Campus professor network system
Nowadays, professors use the internet/intranet to publish information for himself and for his students (including potential ones). A student may need to get assignment online from all his teachers daily. Another student may be browsing the web to find professors that share the same interest as his.
One autonomous approach is that the professors build their different websites separately.
Another approach is that someone stands out to build a comprehensive campus website with index/search pages for visitors and customizable pages for all registered teachers.

Both of them have their advantages and disadvantages. Even the combination is not the ideal solution. Now with web agent, the problem is solved even more than what is required. We start by adopting the first approach where the professors are glad to build their personalized website anywhere. Next, we suggest them to put an agent (actually a set of pages) to the web as representative of the professor and set it to oversee the website for its master (the professor). Agent will maintain relationship with other agents that may represent other professors, a department of the school or even other related people inside or outside the university. Besides, the agent exposes web services that enable students with slight knowledge of it to build personalized client and database manager to build centralized searchable database out of it. A client agent-browser and a filter can help common visitors browse through the agent network consistently and clearly. A visitor may drop at an agent and jump to its master's website.

Overview of the web agent framework's applications

[From Searching to Browsing]

Most of the web agent's applications in stage one; utilize the agent's relation forming capabilities. Before, most part of the internet is only **searchable**. Web agents, however, are a **browsable** community on the web. The relationship used in browsing is currently (in stage one), more or less, their human master's relationship in reality. When browsing, instead of losing state of the last search result, the client agent-browser control can aggregate the tree graph as you explore more of the relationship network. Saving and loading of the browse results are available in special client application for agent browsing.

[Amalgamation of web service and website, agent and people]

Web service and people can be more available and accessible on the web than before. When people come to a website, they prone to think of its potential web services and the group of related people who build the website. In this viewpoint, the program can be regarded as a navigation system on the web using people and their relationship as road signs and linkage.

[Eliciting more advanced agent technology on the web]

In the long run, agent technology will be used extensively on many computing devices. Most of them will be designed to aid or even on behalf of humans. Publicizing agent technology as described in this program will build agent idea into common internet user's mind and as the number of individual agents increases, more innovative ideas will be made out of it.