Intelligent agent technology

Christophe Pincemaille

November 10, 2008

University: Cork Institute of Technology
Department: Department of computer science
Module: Artificial intelligence
Lecturer: Jeanne Stynes

Contents

Abstract

The goal of this paper is to discover what is the intelligent agent technology. This paper has been wrote for the artificial intelligence module, in the Cork Institute of Technology (CIT).

First, the definition of a software agent is rather hazy, so we based on several sources to come up with the fact that a software agent is a piece of software that acts, to do something delegated from somebody. The software agent could be more precisely defined by a set of characteristics, like autonomy, decision-taking, ability to interact with the environment...

Then, we see some frameworks, tools and technologies agent-based. There are quite a lot of them, so we see only a framework, a technology, and a applied technology. JavAct is a Java-based framework, developed by french researchers, that provides a basis for agent-based applications. Telescript technology provides a set of things that enable to make software agents, even a specific language. Finally, we see an example of a state-of-the-art technology based on intelligent agents, that will take part of the future of intelligent agent technology: the world-wide patented *Co-mining* technology.

The report finishes by a case study: how to obtain prime numbers, thanks to the Sieve of Eratosthenes, by collaborative agents. This example comes from JavAct documentation [9].

Introduction

Artificial intelligence is an old but very difficult field of research. So, many systems and technologies have been discovered, tested, improved, challenged. Each of them is a huge research field. We can list for example neuronal networks, genetic algorithms, semantic techniques...

One of the found techniques is intelligent agent technology. In a first chapter, we will investigate on what is actually agent technology, what this term covers. We will first try to reveal a definition. Then we will see the interests of this technology. In the third section, we will go more deeper on how actually that works. The second chapter will be dedicated to the frameworks and implementation which exist. It would be too to be complete and describe all the main existing systems, so we will only see a few examples : a framework, JavAct, a technology, Telescript, and a new innovative technology, based on agent, that makes a preview of what can be done in the future. Finally, in the third chapter, we will show an implementation example of agents, whose interactions will make them able to provide the prime numbers.

Chapter 1

Definition

1.1 What is a software agent ?

1.1.1 A first approach

1.1.1.1 From the human agents...

First of all, what is an agent ?

The Encarta World English Dictionary [3] says that the word *Agent* comes from the latin word *Agere*, which word gave also the words *act*, *active*, *agile*, *agitate*... This etymologic approach is interesting to have a first idea of what it is.

The Petit Larousse illustré [5] tells that all phenomenon, the action of which is important, is an agent. It tells also that it is someone who has to administrate for a state, a society, a single person. And it tells a lot of examples : business agent, litterar agent, police agent...

More generally, in their book "Artificial intelligence, a modern approach", S. Russel and P. Norvig [15] highlight the importance of the environment, defining an agent as something which percepts through sensors and acts through effectors.

So, with this first semantical approach, we can already view that the term of "agent" can cover different notions. But we can conclude that, generally speaking, an agent is somebody or something which acts, in an environment, generally in order to accomplish administration tasks, delegated from somebody.

We can finish this first approach by referring to the Franklin and Graesser classification, which illustrates kinds of existing agents, in very different domains (cf. figure 1.1 on page 3).

1.1.1.2 ...to the software agents

In computing sciences, analogies with the real world are done very often. First, there are analogies related to the way to make unmaterial software being representable, like for "folders", "desktop", "stack", and so forth. Then, there are also algorithmic analogies : software and algorithms are often figured out by inspiring of the reality. That is the case, for example, with genetic algorithms, like aunt-colonies algorithms [22].

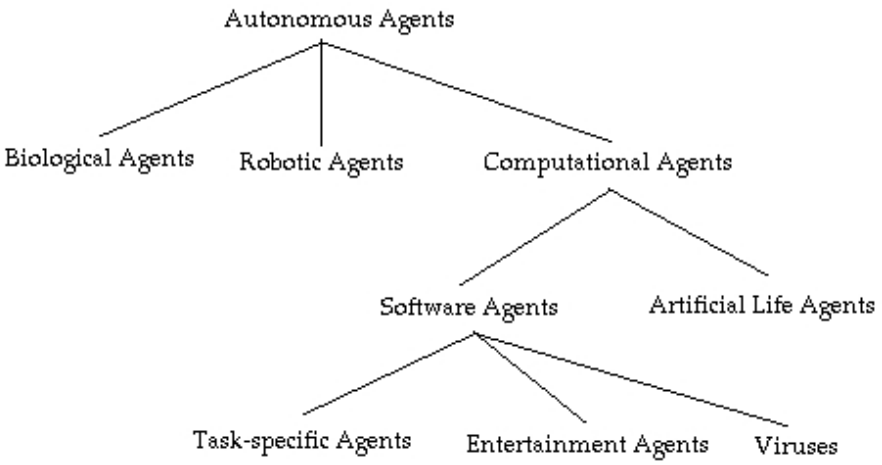

Figure 1.1: Stan Franklin and Art Graesser high-level classification of agents [12]

For software agents, it is the both of us. For the representation, agents are autonomous pieces of software which act, like a real agent. Algorithmically, software agents do tasks, delegated by their launcher.

Which leads us to the software definition of a software agent.

1.1.2 The definition of a software agent

1.1.2.1 Definition on a classic way

There are several definitions of what is an agent, because, as we saw, it is not a unic and well-defined notion [10], and even, according to Hyacinth S. Nwana [16], a "truely heterogenous" concept. There are two of the definitions generally given.

According to the *dictionary of computing* [7], an agent is "an autonomous system that receive information from its environment, processes it, and perform actions on that environment".

According to J. Ferber [10], an agent is a real or abtract entity, capable of acting on itself and on the environment. It can have a partial representation of this environment. It can communicate with other agent. And finally, its behavior is the result of its observations, its knowledge and its interactions.

Eventually, we see that, although this notion is not a rigorous one, the definitions globally cover about the same scope. Thus, concerning software agents, it may be both more practical and more useful to define it by a set of characteristics.

1.1.2.2 Definition based on characteristics

Richard Murch and Tony Johnson, in their book *Intelligent software agent*[6], browse a great number of books which deal with software agent ; then they compile this information

to deduce the general-admitted characteristics of what is an agent[1].

We take that conclusion, and some other web sources (Wikipedia articles in french [24] and in english [23]) to come up to a built conclusion.

Eventually, a software agent :

- is autonomous (and in particular have its own resources) ;
- is made in order to "realize a set of goals" by acting in an environment (in function of its resources, its competences, its perceptions, the communications it gets...) ;
- have to reason, which enables it to select the good action, i.e. have sort of a processing unit (e.g. it is able to analyze its success and errors, it can learn and adapt itself...) ;
- can interact with other agents and can act for another agent, if it is intelligent and if the other has grant the right.

1.1.2.3 Abstract definition

According to *Multiagent systems* [21], we can thus define on an formalized way what is an agent. We assume that the agent's environment can be represented as a set of states, $S := \{s_1, s_2, \ldots, s_n, \ldots\}$. We also assume that an agent dispose of a set of action it can execute, $A := \{a_1, a_2, \ldots, a_m, \ldots\}$. Then the agent can be viewed as a function, $agent : S^* \rightarrow A$.

1.1.2.4 Classification

As the concept of software agent is compound, implemented agents are different, in function of their goal. Consequently, we can classify agents in several categories. The most known classification of agents seems to be the Nwana's one [16] (cf. figure 1.2 on page 5).

1.2 Why using agents in software ?

The agent technology has numerous advantages, which can justify the relevancy to use and develop this technology. But in spite of all the advantages, there are also some drawbacks, which enable to bring a nuance to the speech.

1.2.1 Power of agents in distributed systems

1.2.1.1 Improve the interactions

In distributed and network systems, the great advantage of agents system is that it makes the relationships between machines be closer. Indeed, the idea is replacing a long and far connexion between to points by an important quantity of little interactions. This enables to be more reactive, less sensitive to breakdowns. Moreover, this approach enables

[1] cf. Intelligent Software Agent, section I, chapter 1, section "Types of agents".

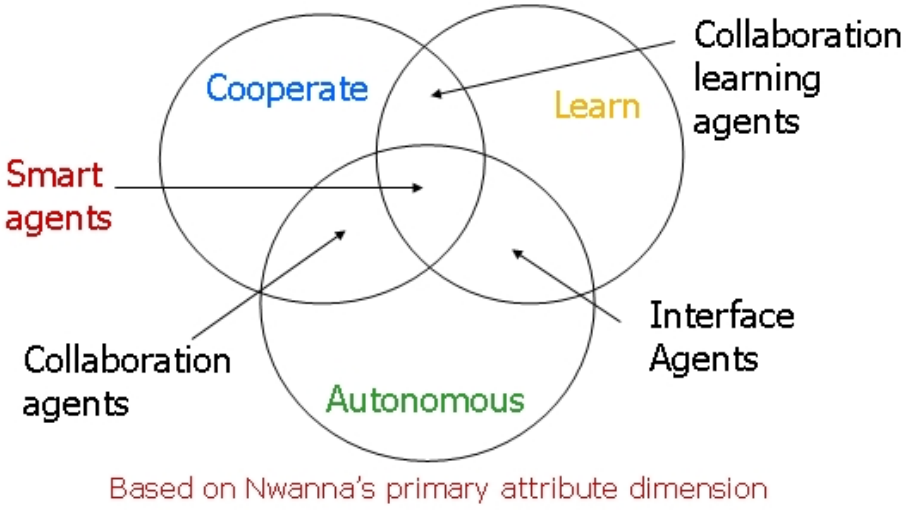

Figure 1.2: Classification of Hyacinth S. Nwana (GPL image)

distributed applications to be far more flexible. Finally, it is also a great advantage at the performance level, since all the method calls will happen locally, and not remotely.

1.2.1.2 Strategic advantage

Then, there is another advantage, that James E. White[2] calls "Strategic advantage". This means that this system enables customization and extensibility. Indeed, when you have a remote system that enables a set of actions, through its methods, you can then add another method using the system methods. The new method will be included in an agent, which will take its data from the system, through its methods.

They take the example of a database system which can give name of people, that enables to delete entries. Then you can add a method that will delete all the entries of people who are a certain age. This method will be included in a software agent. [19]

1.2.1.3 The example of the Web

The World Wide Web is probably the best example of huge-sized system. Finding relevant (and only relevant) information in the WWW is very difficult, given the amount of data. This is the domain of "semantic web".

1.2.2 The group intelligence

We can also remark that, as the G. Luger and W. Stubblefield book [18] highlights[3], agent are interesting when they act in group because they take advantage of what is called the "group knowledge", or "group intelligence". That means that the result of the

[2] cf. [19], page 465
[3] cf. pages 15 and 16 of the book

action of all a numerous group can converge to an equilibrium point that is optimum, related to the problem you have. That means also that the knowledge of a great number of individuals evolving in an environment can be more consistent and extended than the knowledge of several specialists (idea which is the basis of wikipedia, for example).

1.2.3 Nuances to bring

Despite all these advantages, like for all technologies, the intelligent agent technology has its drawbacks. A quick overlook will bring some ideas on the possible problems linked to the agent technology.

First, there is a real problem of platforms. Indeed, the technology in which the agents are implemented have to be supported by the machines in which the agent goes – as a real moving agent moves with its code.

Then, there is a problem in the movement of agents, because when going in the next target, the agent has to restore itself with all its state – not only it initialization state. So it can cause problem to migrate the all current state of the agent, moving to another location.

The last problem we will see, that is also probably the main one, is the problem of security. We can see it at two levels. First, the platforms in which the agents will move can have being protected against external pieces of software to come, by limiting access rights and rights in term of resource consumption. The second level is for the agents themselves : it can be a problem to make agents whose communication are secured, so as to preserve the privacy of the data contained and transmitted by the agents.

1.3 Way the software agents work

In general, there is a general principle of how software agents work, which is always the same. And after, above this basic way of working, can be built a more elaborated mechanism, to comply with the speciality of the considered agent.

In this section, we will see the basic principle of intelligent software agents, and then we will see some examples of conceptions for some specialized agents.

1.3.1 The base of agents

1.3.1.1 The bases of the agent working

We said that an agent evolves into an environment. It percepts some parameters of this environment, and with this information, it can update the partial representation he has of its environment (or not, if it is kind of a state-less agent). So generally, the agent has software components named sensors, which enable it to percept some things of the environment.

Once the environment is updated, the agent can call its internal mechanic. It normally have a software component, supplied by a database (its “memory”) or not, which is its core algorithm, which will process the information, in order to return the appropriate action to do, considering the purpose, or set of purpose, it has.

Then, knowing the action to do, it will consequently do this action, through its actuators. We can note that the actuators are the equivalent of the sensors, but in the opposite direction.

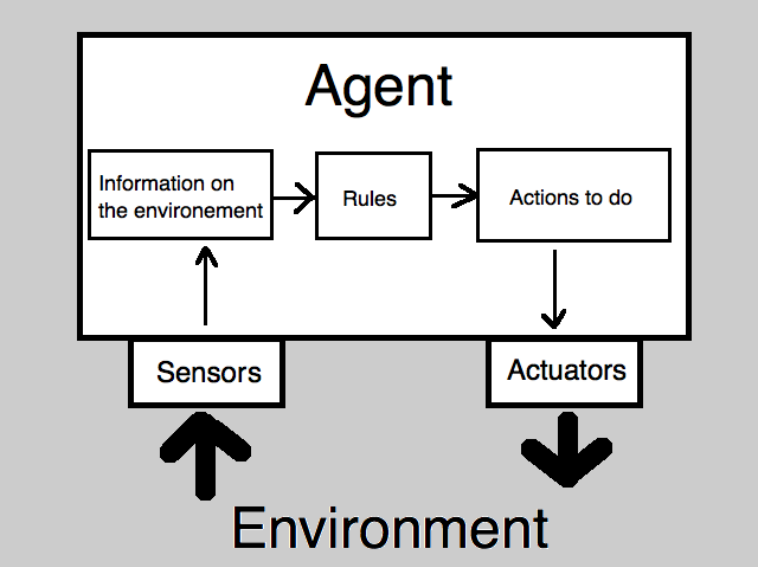

Figure 1.3: Principle of a generic intelligent agent

1.3.1.2 Associated technologies

For agents, we can consider different levels of languages, corresponding to the different parts that constitute an agent.

Implementation language First, there is the language of implementation. We will come back on that later, because generally it is linked to the fact that the agent moves or not. If the agent moves, remote technologies are needed (cf. paragraph 1.3.2.4 on page 9). Else, any language could be used. Then, there is the processing internal language of the agent.

Languages of processing The language of processing is the language with which the agent processes the information, takes decisions, select the appropriate action, and so forth. This is the field of artificial intelligence. Generally, the technologies used are Prolog, SQL, or even KIF, which is a language based on first order logic, and extends it [14]. The language of processing goes along with the ontology, which is a pre-defined and agreed common vocabulary, to define a subject domain. And then, there are language used to communicate.

Agent Communication Language (ACL) To communicate, agents can use a lot of technologies. The two main ACL are KQML [14] and FIPA.

KQML is a high-level, message-oriented, communication language. This is independent of the implementation and transport languages and protocols (RMI, CORBA, Java, TCP/IP...), of content language (Prolog, KIF...), and of the ontology. KQML have originally been designed as a language based on Lisp, but there are other versions of the syntax, that fit more with SMTP, MIME, HTTP... KQML enables agents to communicate, sending message of asking, informing, replying, promising...

FIPA is the ACL of the Foundation for Intelligent and Physical Agents. The first version has appeared in 1997. It is spreading quickly, and has a "tendancy to replace KQML" [20].

1.3.2 Example of the mobile agent

Even if there could be agent which are not mobile, acting for example in a piece of software installed locally, intelligent software agents are generally mobile. Moreover, with the development of the Internet, mobile technologies have become very important [11]. So it is important to see how mobile agents work, or at least see some of the possibilities that exist.

1.3.2.1 Some examples

First, we can see some examples of applications in which agents need to be mobile (cf. [13]) :

Notification agent that is like the design pattern observer ; the agent waits for an event, and when the event occurs, notifies the client. There, it is different from the pattern observer by the fact that it is a distributed application, and the agent is on the same computer as the server, so it need to reach that location.

Itinerant agent this kind of agent has to move in different locations, and make interactions with the servers it reaches. An example is here an agent which makes a link between data[4].

Adapting agent here, this is close to the pattern adapter : the agent has the method which the client will call instead of the same method of the server, because the agent will realize an adaptation. The idea is the same as notification agent : the difference is in the fact that the agent is close to the server[5]. Here, the agent can even move to make adaptations for more than one server...

1.3.2.2 General principle of moving agents

The principle is rather simple. There are the *agents*, which can move to *places*, by *moves*. Agents are active entities, generally threads [11]. Once they have moved, they can *meet* other agents or servers which are in the same place, or send *messages* to contact remote agents.

[4]Cf. [13], slides 9 and 10.
[5]Cf. [13], slides 11 and 12.

1.3.2.3 Problems linked to implementation

The main problem is the mobility of the code. Indeed, the source and destination machines might not be under the same system. Besides, it could be difficult to save the state of the agent, to send the code and the state, and then to launch the code and retrieve the state. Finally, there can be security issues.

For the homogeneity of systems, generally it is solved using virtual machine or interpreted technologies. The security issues are generally resolved by introducing rights to the agents. Research are made in that domain [17].

1.3.2.4 Implementation technologies

So, we have seen that well-adaptated technologies for mobility are virtual machine or interpreted technologies. The main technologies used are :

Python Python is an interpreted language, which makes it rather independent of the system - the target system just have to have a Python interpreter.

Java thanks to the virtual machine, the code is platform-independent, which is a great advantage. The dynamic class loading is also powerful in that case. Moreover, Java includes useful technologies like RMI and the serialization.

1.3.2.5 Application to Java : process of migration

Java is probably the most used technology for distributed agents. We see here the process to move an agent from a location to another. The process is the following one.

1. the agent is serialized, with its current state ;
2. the byte-code and the serialized agent are sent to the target computer ;
3. the agent on the source computer is destroyed ;
4. on the target computer, a new thread is created ;
5. the code is loaded and the agent de-serialized ;
6. execution is launched.

Chapter 2

Actual frameworks, implementations and prototypes

Introduction

Intelligent agent technology is interesting mainly for applications deployed on distributed systems. There are such a lot of applications which work on distributed systems, that it makes a great range of application domains for intelligent agent technology.

We will present some, like JavAct and Telescript, but there are some other systems, like the aglets, AgentTCL, Mole, or ObjectSpace Voyager Core Technology, we will not present here. Indeed, for all these technologies, the principle are globally the same, with variation in details of implementation, framework conception, and employed languages. So we will see the principles with one or two, knowing that the others are based on the same global principles.

For more details on all these technologies, we recommend to see the book *mobility*, which contains a lot of articles and publications explaining all these technologies : [2].

2.1 A framework of distributed agents: JavAct

JavAct is a library, written in Java, which provides a basis for implementing software based on software agents. These agents can be mobile, concurrent and distributed.

2.1.1 The framework

This library is distributed under LGPL Licence [4]. It is composed by :

- the actual framework, the last version of which is the 1.5.3 (it can be download at the URL `http://www.irit.fr/PERSONNEL/SMAC/arcangeli/JavActv153.jar`) ;
- a tutorial, in french, of the library [9] ;

- and even a plugin for Eclipse, which makes easier the development with the library (the plugin for Eclipse 3.2 is available at `http://www.javact.org/JavAct.V.1.2.4.zip` with its user manual at `http://www.javact.org/JavAct-MU-v3.0.htm`).

2.1.2 The producers

The framework has been developed by the SMAC team[1] of IRIT Lab.

The IRIT is a research center in computer sciences, based in Toulouse (France). The internet presentation tells that this research center has played an "important role in Toulouse computer science research". It is a relatively important center, given that is includes a staff of 400 members, 300 of whom are researchers.

The researchers are organized in 7 main topics. The SMAC team (cf. `http://www.irit.fr/personnel.php3?theme=1&id_rubrique=65&equipe=SMAC`) is in the third theme, "Interaction, autonomy, dialogue and cooperation".

For more information, you can go to their website, which has a part in english : `http://www.irit.fr/sommaire.php3?lang=en`.

2.1.3 Elements provided by the library

The basis of the library is the implementation of "actors" (the agents) which will communicate through "messages".

JavAct defines several elements, which have to be developed in a given order. First, it is necessary to define the actors' behaviors, through the definition of java interfaces. Then, JavAct can generate automatically message classes, and abstract classes for the behaviors. Finally, using the auto-generated classes, the behaviors can be implemented.

2.1.4 Advantages of this platform

As we saw in the advantages and drawbacks part (cf. section 1.2 on page 4), one of the main problem in distributed autonomous agents is the platform. Indeed, agents have to be developed in a code which is kind of multi-plateform, to be able to spread. So for this point, JavAct is interesting because it is developed in Java, known for its portability.

Besides, JavAct enables the developer to forget all the low-level and distributed systems mechanisms, like threads, synchronization, RMI... The result is, this framework can be used by a classic java developer.

Finally, there are some more interesting elements, like the absence of preprocessing needs.

2.2 Telescript technology

The first commercial implementation of mobile agent technology is General Magic's *Telescript Technology*^{TM}. It is a e-commerce application, which models a market center. [19]

[1] Systèmes Multi-Agents Coopératifs

2.2.1 System description

In that application, there are some *places*, one enabling to buy flowers, another enabling to buy tickets to sporting events... In each of these places, there is an agent. Agents can move from a place to another. This is distributed technology, as it is a mobile agent technology, so the agents' actions can be performed concurrently. At one time, there is one agent for each place, which provides the services related to the place : give information for example.

2.2.2 The traveling agents

There are agents for users, which are capable of traveling through the different place, in order to obtain what they want. That is done thanks to the *Telescript language*, which enables moving the program during execution (and not before, like conventional C or C++ programs), by a `go()` procedure. We find here one of the main characteristic of mobile agent technology.

2.2.3 Security

Another interesting thing in that technology, is that it includes security considerations. Indeed, there are systems of *authorities* and *permissions*.

For the authority system, the agents have an *authority*, related to the individual or organization the agent represents in the real world. That can be useful to control access to data, for example.The absence of anonymity can also prevent the acceptance of virus.

The agents have also a characteristic of rights, which will depict the *permissions* they have. For example, their permissions can grant them the right to execute a code stub, to create new agents... The permissions can also determine characteristics of the agents, like their lifetime, their maximum size... An agent that overall at least one of the parameters is directly removed. That insures a coherence of the system, and insures that the system will not explode or make non-authorized things.

2.2.4 typical scenario

So an agent goes from a start point (probably inside the user machine), thanks to a *ticket* which contains information on the destination, moves and meet another agent in the destination place (by the `meet()` procedure, that can fail if the agent is too late, for example). Then, the destination agent can provide the wanted thing if it has got it, or redirect our user's agent to another place, to meet the good agent, giving our user's agent a new *ticket* and calling the `go()` procedure. So this enables a complex and composite service.

2.2.5 Implementing in Telescript

The telescript language is object-oriented. Thus, it includes features of oriented-object languages : encapsulation, inheritance, exception system...

For the transport, Telescript protocol can be used with a lot of networks protocols : TCP/IP, X25, even electronic mail.

2.3 A new technology about agents

Now, new technologies are developed with intelligent agent technology. These technologies result in building new innovative, state-of-the-art enterprises. A good example of that is the technology of the french enterprise *Co-decision technology SAS*, founded and managed by Martine Naillon.

2.3.1 The technology

2.3.1.1 What is it ?

The technology, named "Co-mining", is based on a *cyber-decider*, named `decider.track`. This technology, world-wide patented, has been discovered and developed by Martine Naillon (PhD is mathematics). It reproduces the conscious and subconscious working of human brain, like an "electronic neurotransmitter", in order to make relationships between distributed and heterogenous databases. The technology goes to the databases, collects data, makes relationships with this data, and eventually is able to find some wanted information, extracted from heterogenous sources.

2.3.1.2 Cycle "Alarm-Warning-Response"

Thus, one of the power of this tool, is that it is able to take decisions face to some extracted complex information. Indeed, it automizes the cycle "Alarm-Warning-Response": when successfully extracted a rumor, it is able to make relationships between these alarms (Alarm), and evaluate if it is nothing important, or if it is something for which a focus is necessary (Warning). Then, it is able to decide of the adapted answer to give to the potential threat (Response) [8].

2.3.1.3 An innovative agent technology

This agent based technology is really new and innovative. Indeed, there are actually agents which are involved, but not as the classic way of agent-working. Here, the agent are not moving to collect the information. The information is included in portable distributed objects, which are moving among the motionless agents, that use the objects to construct the relevant deduced information.

Figure 2.1: Co-decision technology

2.3.2 The enterprise: application domains and clients

Co-decision technology SAS is the enterprise founded and lead by Martine Naillon, based on Co-mining technology.

This technology is applicable within several domains. This is data-mining technology, able to extract information from heterogenous databases. So it is perfectly adapted for databases of police and defense services, whose bases are not necessarily unified, but for what it is important to have decisions based on all the available information.

In the domain of defense, co-mining technology achieve projects in global terrorism in particular. Its clients are the authorities of France, United Kingdom, Spain, Germany, the USA... and is also present in Doubaï or Israël [8].

Co Decision Technology has also activities in the financial sector. The technology contributes to what is done in semantic technologies in that sector: financial market analysis, automatic decision taking...

For more information, you can go on the website : `http://www.comining.com/`.

Chapter 3

Case study

3.1 Introduction

To finish this report, we will see here a real implementation case of software agents. We have seen that there can be a lot of different kinds of agents: moving, collaborating, and so forth. Here, the example deals with agents that collaborate but don't move. Indeed, the goal is to present a short example, in order to see how software agents can be implemented, and not to detail a huge example of agent system. So we prefer not to see distributed agents. Moreover, in the same idea of simplicity, we will present an example based on a framework for software agents. Then we will able to see more quickly the actual working of software agents.

3.2 The Sieve of Eratosthenes

In this example, agents interact to implement a Sieve of Eratosthenes[1]. This is one of the examples givent in JavAct documentation[2], so the authors of this example are J.-P. Arcangeli, V. Hennebert, S. Leriche, F. Migeon, and M. Pantel [9].

3.2.1 General idea

In this example, each agent is linked to a given number, which will be one of its attribute. The agent can receive a number, process the number, and give or not the number. The processing here will be simple: if the received number is not a multiple of my attribute, I give the number to my successor, else I do nothing (the received number stops its travel here).

The idea is to give each prime number to agents, and to put agents in sort of a "responsibility chain". Numbers are launched to the first agent at the beginning of the chain, which filtrate the number or let is pass to the following agent, and so forth. At the end of the chain, are only prime numbers, because no prime ones have been filtrated

[1] To have more information about what is sieve of Eratosthenes, you can read: `http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes`

[2] Cf. `http://www.javact.org/tutoriel_4/node5.html#SECTION00053000000000000000`

before. Finally, when a prime number arrives at end of chain, a new agent is created, associated to this prime number, which will filtrate multiples of this prime number [9].

3.2.2 Implementation

In the implementation, there are two different behaviors: the intermediate behavior, for actors at the middle of the chain, and the last behavior, for the actor at the end of the chain. Actors can change of behaviors, given that the actor in end of chain can change and become in the middle of the chain (when a new prime number arrives in end of chain, and a new actor is created, related to this new prime number) [9].

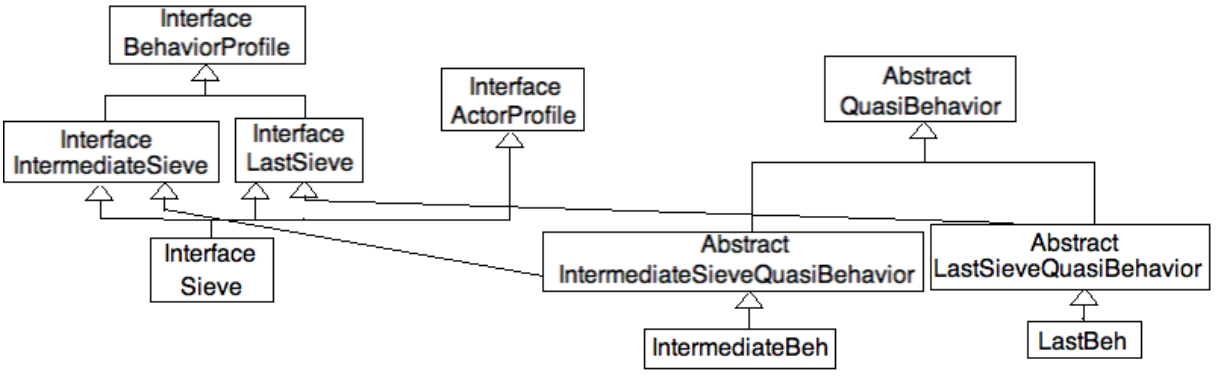

Figure 3.1: Class diagram of the example

So the profiles can be defined like that:

```
public interface IntermediateSieve extends BehaviorProfile {
    public void sift(int i);
}

public interface LastSieve extends IntermediateSieve {
    public void become(IntermediateSieve b);
}

public interface Sieve extends IntermediateSieve, LastSieve, ActorProfile {
}
```

The behaviors can be implemented like that:

```
class IntermediateBeh extends IntermediateSieveQuasiBehavior {
    protected int prime;
    protected Actor next;

    public IntermediateBeh(int i, Actor a) {
```

```
        super();
        prime = i;
        next = a;
    }

    public void sift(int i) {
        if ((i % prime) != 0)
            send(new JAMsift(i), next);
    }
}

class LastBeh extends LastSieveQuasiBehavior {
    protected int prime;

    public LastBeh(int i) {
        super();
        prime = i;
        System.out.println("*** " + prime + " is a prime number ***");
    }

    public void sift(int i) {
        if ((i % prime) != 0) {
            Actor next = create(new LastBeh(i));
            become(new IntermediateBeh(prime, next));
        }
    }
}
```

To run the application, it is just necessary to create a new actor, associated to the prime number 2. Then, all the integers are sent to this actor, in crescent order. The integers will be transmitted along the chain, stopped in the middle of the chain if they are multiple of the prime number associated to one actor, going at the end of the chain (that creating a new actor associated to this number) if it is prime.

```
public class PrimeNumbers {
    public static void main(String[] args) {
        if (args.length != 0) {
            int valint = Integer.parseInt(args[0]);
            if (valint > 2) {
                Actor two = CreateCt.STD.create(new LastBeh(2));
                for (int i = 3; valint >= i; i++)
                    SendCt.STD.send(new JAMsift(i), two);
            } else System.out.println("Give an int > 2");
        }
    }
}
```

Conclusion

We saw that a software agent, inspired from the reallity, is a piece of software that acts for somebody or something. It has several characteristics, which generally are autonomy, goal-oriented, decision power from a processing, interactions with environment and other agents. It is very interesting for distributed systems, and to take advantage of some concepts like group intelligence. They can be implemented in many ways, but Java is often used because of its portability.

A lot of frameworks are developed, like JavAct, which provides a basis to make java agent-based applications. But there are also less conventional technologies which exist, like Telescript.

The distributed systems take more and more importance in the computing systems, and in the way to imagine software. We already see, with the Internet, a actual and real take-off of software agent technologies. It is true with all the new technologies which are developing with the Internet: semantic web and all the linked technologies (asynchronous technologies that enable to exchange data...) is the best example.

For the technologies, the important development of Java and Corba technologies, combined with the development of frameworks, make possible and even easy to make agent technology, which participate to its probably growth.

According to Murch and Johnson[3] [6], it seems that there is no need for agent technology to disappear. It may even be the opposite: other systems may, from now, be made with agent technology, as they integrate in distributed systems (like the Internet).

As for the application domains, it seems that agents "will be the backbone of e-business"[4]. For network operations, there is a "big opportunity", to use software agents instead of classic means, and save staff and money.

In conclusion, intelligent agent technology is growing in the future, as there are many opportunities (business and technological), and techniques should consequently improve.

[3]cf. section III chapter 22 page 181

[4]Murch and Johnson [6] page 183

Bibliography

[1] Jeffrey M. Bradshaw. *Software agents*. The MIT Press, 1997.

[2] Frederick Douglis Dejan Milojicić and Richard Wheeler, editors. *Mobility – processes, computers, and agents*. Addison-Wesley, February 1999. isbn: 0-201-37928-7.

[3] Encarta, editor. *Encarta world english dictionnary*. Bloomsbury, 1999. isbn: 0-262-52234-9.

[4] Free Software Foundation. Gnu lesser general public license. `http://www.gnu.org/licenses/lgpl.txt`, June 2007.

[5] Larousse, editor. *Petit Larousse illustré 1987*. Larousse, 1987.

[6] Richard Murch and Tony Johnson. *Intelligent software agents*. Prentice Hall, 1999.

[7] Oxford, editor. *Dictionnary of computing - fourth edition*. Oxford, 1996. isbn: 0-19-853855-3.

[8] *(no author)*. Co-decision technology sas – Pour optimiser la synergie entre les services, les hommes et les cultures. *Guide européen de 'innocation*, page 134, 2008.

[9] J.-P. Arcangeli, V. Hennebert, S. Leriche, F. Migeon, and M. Pantel. JavAct 0.5.0 : principes, installation, utilisation et développement d'applications. `http://www.javact.org/tutoriel_4/index.html`, 2008. Documentation from IRIT (Institut de Recherche en Informatique de Toulouse).

[10] Gilles Balmisse. Les agents, 2002.

[11] J. Baumann, F. Hohl, K. Rothermel, and M. Strasser. Mole - concepts of a mobile agent system. In Frederick Douglis Dejan Milojicić and Richard Wheeler, editors, *Mobility – processes, computers, and agents*, chapter 14.5, pages 536–556. Addison-Wesley, February 1999.

[12] Stan Franklin and Art Grasser. Is it an agent, or just a program?:a taxonomy for autonomous agents. `http://www.msci.memphis.edu/~franklin/AgentProg.html`, 2007.

[13] Daniel Hagimont. Modèles à agents mobiles. *(no date)*. `http://rangiroa.essi.fr/cours/car/98-flips-agents-mobiles.pdf`.

[14] Yannis Labrou and Tim Finin. Agent communcation languages: past, present and future. http://www.cs.umbc.edu/pub/finin/talks/icmas00.pdf, July 2000.

[15] Georges F. Luger and William A. Stubblefield. *Artificial intelligence - structures and strategies for complex problem solving - third edition*. Addison - Wesley, 1997.

[16] Hyacinth S. Nwana. *Knowledge Engineering Review*, volume 11 of $N^0 3$, chapter Software Agents: An Overview, pages 205–244. Cambridge University Press, 1996.

[17] Yanfeng Peng, D.J. Holding, and K.J. Blow. A possible secure solution for mobile agents. http://www.ee.aston.ac.uk/research/acrg/papers/iawtic01.pdf, *(no date)*.

[18] Stuart J. Russel and Peter Norvig. *Artificial intelligence : a modern approach*. Prentice Hall International, 1995.

[19] James E. White. Telescript technology: mobile agents. In Frederick Douglis Dejan Milojicić and Richard Wheeler, editors, *Mobility – processes, computers, and agents*, chapter 14.1, pages 461–493. Addison-Wesley, February 1999.

[20] Zhao Yanping. Agent software development and electronic commerce. http://www.cs.usask.ca/grads/yaz720/856/asummary.html, Consultation on october 22^{th} 2008.

[21] Gerhard Weiss, editor. *Multiagent systems – a modern approach to distributed artificial intelligence*. The MIT Press, 1999. isbn: 0-262-23203-0.

[22] Wikipedia-en. Ant colony optimization. http://en.wikipedia.org/wiki/Ant_colony_optimization, September 2008.

[23] Wikipedia-en. Intelligent agent. http://en.wikipedia.org/wiki/Intelligent_agent, October 2008.

[24] Wikipedia-fr. Agent (informatique). http://fr.wikipedia.org/wiki/Agent_(informatique), March 2008.