Some Issues on Agent Ownership[1]

Alexander Yip & Jim Cunningham
{ay1, rjc}@doc.ic.ac.uk
COMMA Group, Department of Computing,
Imperial College of Science, Technology and Medicine,
Huxley Building, 180 Queen's Gate, London SW7 2BZ, UK.

Abstract

Agent ownership is a complex issue that crosses legal domains. There are at least two different dimensions to agent ownership. On one dimension, an agent is a piece of software and as such it is an intellectual property which belongs to its owner and is governed by the law on intellectual property. On another dimension, a software agent can act like a business agent. In this paper we discuss some of the social and legal ramifications of software agents and highlight areas where agent ownership will be of interest.

1. Introduction

As agent technology develops, it will be applied to more and more real world applications. However, before this technology can be widely adopted by the general public, it is proper to consider its legal aspects. One particular area is agent ownership.

There are at least two different dimensions to agent ownership. On one dimension, an agent is a piece of software and as such it is an intellectual property which belongs to its owner and is governed by the law on intellectual property. This raises a whole set of questions like "Who is the owner?", "What is owned?", "How can an agent's ownership be transferred?", "How can an agent owner protect his/her property?", "What is the legal relationship between an agent, an agent user and an agent developer?" and "What are the legal responsibilities of these principals?". To support some of the issues, it seems necessary to enhance the existing FIPA agent model *[AMS00]*. One such enhancement is the provision of a more detailed agent authentication model.

On another dimension, a software agent can act like a business agent. There is a different set of laws governing the legal relationship in which a human business agent deals with a third party on behalf of some principal. One important issue addressed by the law of agency is the contract formation between an agent and its principal. However, as the existing law did not foresee the arrival of software agents, it may be necessary to adapt the law in this area to accommodate this new technology.

Thus agent ownership is a complex issue that crosses legal domains. Although some of the issues are already addressed in network and software security, agent ownership introduces many new social and legal challenges. In this paper we discuss some of the social and legal ramifications of software agents and highlight areas where agent ownership will be of interest. Our conclusion will address the improvements needed for the FIPA agent security model to support these findings.

2. Some missing links in current agent ownership model

It is naïve to expect that a customer will be able or willing to program a software agent from scratch. Instead software companies can be expected to provide readily useable software agents for purchase or "for hire", thus presenting agents as products, not just a technology. Here we sketch three scenarios through which we seek to probe the law and technology for agents.

[1] In "The Law of Electronic Agents: Selected Revised Papers", Proceedings of LEA Workshop, CIRSFID, University of Bologna, July 2002, pp. 13-22

2.1. Multi-agent Service systems

Company X, a supermarket chain, has started to use software agents to handle orders from its online customers. X has a large number of electronic sales agents available at a portal to welcome customers' personal agents…

1. When Amy's personal agent visits this site on a weekly basis to purchase necessities for Amy, it will encounter different software agents every time. It is important that they look past the identity of each individual agent and recognise the authority they are representing.

2. Company X will have to replace or upgrade its pool of software agents from time to time because of policy change and new software releases etc. But these new software agents will still represent the company's authority. Other software agents should have the same level of trust for these new Company X agents without the need to build up the trust from scratch.

2.2. Trustworthy Information Agents

Amy bought a personal agent from Alpha Corporation. Amy sent her personal agent onto the web to find the cheapest airfare to New York. Amy's PA contacted a number of travel agents and asked for the availability and price. The search came up with two possible companies, "National Prestige Airways" (NPA) and "Here Today, Gone Tomorrow Airways" (HTGTA), although HTGT Airways' ticket is much cheaper by far…

1. Alpha Corporation is a specialised software company and its business model depends on the ability to sell software agents to customers. It is in its interest that it owns the software agents' mechanisms and prevents unauthorised distribution of its software agents.

2. Amy would like to maintain the ownership of her copy of the personal agent, because she has trained it to recognise her preferences and more importantly, the personal agent contains her personal information.

3. Can HTGTA really honour the cheap airfare to New York? When a software agent contacts other software agents for information, it is necessary to verify the trustworthiness of the data. Many research have already tried to model the trust in agent interaction purely in the agent world. However, an agent can only be as trustworthy as the source of its information; identifying the ownership of the agent can help to establish the level of trust among the participants. Knowing who owns the agent that gives out information is vital in certain situations.

4. Amy's PA is freshly created for Amy, it has not interacted with any software agents before. How can it assign its trust to any software agent that it is going to encounter on a task? Castelfranchi and Falcone *[Castel.] suggest that* trust can be described as a prediction of the probability of something happening based on one's past experience. A human infant has a long time to build up this experience but a software agent may be expected to behave perfectly on the day it is installed. Moreover, human society has many aids to help with the decision: a brand conveys more than just a name, it also has the reputation of the company behind it. The Media helps to distribute information regarding companies' trustworthiness and also trust in the justice system which seeks to protect members of its society. Our existing agent models lack the mechanism to distribute either brand or trustworthiness information throughout the system.

2.3. Agents for hire

Amy needs some financial advices. As she only needs this service once, she thinks it is more economic to hire the agent for the task rather than buying one. She has contacted EZ Financial Solutions for the hire. A suitable agent is selected and delivered to her…

1. How can EZ Financial Solutions protect its interest by ensuring only Amy can use that agent? A user of an agent system with the direct control over the agent code can pass the code to other user.

2. Can EZ Financial Solutions protect its interest by imposing and enforcing more restricted terms in the contract? For example, can only a single copy be allowed, can an agent be time limited and action limited etc? In many current agent systems, a user also controls the execution of the agent code and may create as many copies as he/she wants.

3. For sensitive information service e.g. financial advice, legal advice etc, there are laws governing professional conducts of a human agent. Can a software agent satisfy these requirements?

3. Agent management scenarios

3.1. The FIPA Agent Management System

Agent A can register itself with an FIPA AMS [AMS00] with the following agent description:

```
(request
          :sender
          (agent-identifier
          :name A
          :address (sequence iiop://foo.com/acc))
……
          :content
          (action
                    (agent-identifier
                    :name ams@foo.com
                    :address (sequence iiop://foo.com/acc))
          (register
                    (ams-agent-description
                              :name
                              (agent-identifier
                              :name A@foo.com
                              :addresses (squence iiop://foo.com/acc))
                              :state active
                              :ownership NoBody
                    )
          ))
)
```

1. Does agent A really belongs to NoBody? At this moment there is no defined protocol or action associated with the ownership field. It is an optional field in the Agent Management Ontology. For example, Eva can set the content of this field to make the agent looks like it is owned by Bob.

2. What does it mean by NoBody owns this agent? At present the way an AMS will respond to this request depends on the platform developer. However, as FIPA is promoting connectivity between heterogeneous agent platforms, it is necessary to standardise this response to support a coherent ownership model in the FIPA agent world.

3. It is possible that not all platforms will support ownership. In such a mixed environment, agents still need to be able to reason about the concept of ownership e.g. the concept of authority and delegation from owner etc. In order to support this, an agent needs a standard ontology to provide a foundation for this reasoning.

4. There is no mechanism to verify a claim of ownership. It seems necessary for FIPA to specify a management interface between the human world and the agent world so that information like ownership can be conveyed between the two domains.

3.2. Trust and Agent Ownership

Many researchers treat ownership as a passive ingredient of trust. For example, Rahman and Hailes *[Rahman]* propose a distributed trust model based on the assumption that an agent will be able to keep a history of interaction with other agents and hence assign different level of trust to the peers. However, as pointed out in the scenarios, it is not always possible to build up the necessary record to decide the level of trust towards an agent, especially when there is no direct relationship between the real world legal entity and the software agent representing it.

In other research *[Castel, Mamdani, Dignum]*, it is argued that in the design of multi-agent systems, in order to support rich collaborative behaviour it is essential to provide individual agents with various forms of social awareness. Mamdani and Pitt have suggested that an agent should be able to express the fact that an agent is owned by some human entity and be able to reason about the consequences and responsibilities of the delegation.

Should ownership and trust be viewed as a mutual relationship? Ownership helps to sustain a trust model by providing a legal foothold for trust to develop. For example, a consumer may prefer a product from company A more than a product from company B. The consumer’s explanation may be that he/she trusts the product from company A more. This can be interpreted as the consumer trusting the brand owned by company A more even though each individual item can be of different quality. At the same time, to establish ownership requires trust on the underlying legal and technical systems. The legal system must be trusted to support the benefit of ownership and the technical system trusted to sustain ownership.

4. The Agent as a program – The Agent ownership life cycle

Mamdani and Pitt have used a motorcar analogy in their paper *[Mamdani]* to illustrate the relationship between the principals in an agent world:

- *A driver (= user) owns a vehicle (= agent and/or software) but pays road tax to drive it (legally) on the roads (= third-party owned infrastructure);*

- *A driver registers ownership with Vehicle Licensing Authority (= trusted third party), needs to pass a test to gain a license to drive a certain class of vehicles, and buys insurance;*

- *A driver fills up with gas (= buying CPU cycles), requires a yearly Roadworthiness Certificate (= agent upgrade by self configuration), and may have the steering wheel on the left or right (= communication protocol as either HTTP or IIOP).*

One can extend this analogy to illustrate some ownership issues within the agent world:

- A car manufacturer designs and manufactures motorcars (= program and off-the-shelf agent software). The manufacturer is the owner of the motorcar design and no one is legally allowed to copy the design without obtaining his/her permission.

- A car manufacturer uses the same design (= agent design) to produce a large number of vehicles (= packaged agent software), which are identical except the unique serial number assigned to each copy. However, when a user acquires a vehicle, the user may want to choose the fittings and decor (= customising the software agent).

- A driver (= user) buys a vehicle (= software agent) from the manufacturer or hires a vehicle from the car rental company (= software rental/agency)

- If a driver buys the vehicle from the manufacturer, he only owns a single vehicle (= an executable instance of the agent) and not the design. The driver is not allowed to manufacturer the vehicle for other people even if he has the necessary technical skills.

- If a driver hires the vehicle from a car rental company. The driver (= user) is the beneficial owner of the vehicle (= software) under the terms set out in a contract between the driver and the rental company.

Although this analogy is useful for illustrating some principles of agent ownership, but there exists a serious limitation in this analogy – software processes do not have the same constraints as physical objects:

i) If suitably written, one piece of code can be executed concurrently by one or more physical processors

ii) Code duplication is an important technology for mobility and distributed processing.

Following the analogy, a user may be allowed to create a pool of agents from the single copy of agent code he/she purchased (subject to contract and not for resale) but it is not possible to turn a single car into a fleet in the real world.

Despite these discrepancies between software and physical objects, they still share some common concepts. Five principle entities can be identified in the agent ownership life cycle, they are: User, Owner, Beneficial Owner, Agent code and the Active agent (or Agent process instance). This is a summary of their relationships:

- The ownership of an agent can begin when a designer invents the software that can be executed as an agent. The programmer or the programmer's employer is the copyright owner of the agent code.

- Through a sale or hire contract, the copyright owner of the code can license the right to other beneficial owners. A beneficial owner is a person who will enjoy the benefit derived from the copyrighted material under the terms set in the contract.

- A user purchases or hires the program and executes it on a computer. The person will be the owner of the active instance or instances of the software or the beneficial owner of the agent code. During the "life" of the software agent, it belongs to the user.

- Upon the termination of a software agent execution, the beneficial ownership of the User of that particular active agent will come to an end.

5. The Software Agent as Legal entity: Business Agents

A common application for agent-based technologies is to provide services for their owners. Human business agent models are being re-applied to the computer network and software agents are replacing human agents.

The problem is, existing law did not foresee the development of software agent technologies. The legal position of software agents is not defined in the existing law system; one way to solve this problem is to try to make software agents that will satisfy the existing legal requirements.

5.1. Legal entity

A legal entity is an entity that is authorised by law to enter into a contract with other legal entities. Generally an entity must have two attributes to qualify as a legal entity:

1) It must have a well-defined decision making authority.

2) It must have the ability to bond its contracts credibly.

A natural person, who is rational, satisfies these two requirements so can be a legal entity. A firm with a well defined decision-making structure and with a pool of assets that the managers can offer as satisfaction for the firm's obligations can also be a legal entity.

Based on these general requirements, existing software agents cannot be classified as legal entity because:

1) There is no well-defined decision making authority. This requires two mechanisms:

    a) the ability to audit an agent's decision and actions. There are research in this area like the enhancement proposed in the EU MIAMI project *[MIAMI]* but there is no effort to make this a standard
    b) the ability to identify the authority that an agent is representing.

2) An agent generally does not have asset that can be used to bond its contract credibly.

If a standard framework supports agent ownership, it should be possible to improve this situation because identifying the agent owner will complete the chain of decision-making authority and the owner will be able to provide the necessary asset to bond with an agent's contractual obligations. However, the issue on how to limit the liability of an agent owner is outside the scope of this paper.

5.2. Business Agents

In law, an agent is a legal entity who is employed to bring his principal into contractual relations with third parties and various forms of agency, regulated by law.

The term Agency means the relationship that exists when one person or party (the principal) engages another (the agent) to act for him. The law of agency thus governs the legal relationship in which the agent deals with a third party on behalf of the principal. On the one hand, the law of agency is concerned with the external business relations of an economic unit and with the powers of the various representatives to affect the legal position of the principal. On the other hand, it rules the internal relationship between principal and agent as well, thereby imposing certain duties on the representative.

5.3. Can a software agent act as a business agent?

Under the current law system, agent and agency is based on the interaction between legal entities and the whole definition is to assign the legal responsibilities between the entities. In our existing law these entities are expected to be human and each rational individual is expected to take responsibility for his own actions. If such an entity is a software agent:

i) it is necessary to determine that the software agent was functioning according to its specification (under the same scrutiny as any software system)

ii) it is necessary to identify the entity responsible for delegating the action to the software agent, hence taking legal responsibility of the action.

6. Owning an agent

As presented in the scenarios, it is possible to exert two different types of ownership on an agent. On one hand, an agent can be treated as a piece of software hence its intellectual property can be owned. On the other hand, the user of an active software agent owns the agent's service. However, this introduces a legal hurdle because existing law does not support the ownership of an active software instance.

The most common way to protect software is copyright. In this form of protection, a piece of software is being treated as a creation and the owner is the creator/programmer of that piece of code. As a software agent is a piece of software, the source code (agent code) can be protected by copyright. There are two main forms of proof of ownership – the proof can be applied externally or be integrated into the software. An example of external proof is Signed Java Code. In Java, the software components can be packaged into a type of archive file called JAR. A developer can then digitally sign the JAR file to protect the integrity of the contents. This mechanism is supported by Java in the system level so that the Java system will refuse to execute a piece of software that has an invalid signature. In the case of integrated proof, some research have been done in this area. Collberg and Thomborson *[Collberg]* suggest an algorithm for watermarking a piece of software so that the program will function as normal but each copy will carry a unique signature. The mechanism protects the watermark so it is detectable even after a distorting attack

Thus a digital signature and digital watermarking can protect the integrity of a piece of software and provide proof in case of a dispute. In order to protect the secrecy of a piece of software, there is research in the field of Function hiding. An example is Sander and Tschudin's work *[Sander]*, their paper presents a generic algorithm to transform and hide functions inside a program so that the program can be executed to obtain the desired result without disclosing the internal algorithms.

However, how can a user maintain his/her ownership of an active instance of the software agent? Traditionally when software ownership is discussed, there is no distinction between a piece of software and an active/executing instance of that piece of software. This is partly due to the legal definition of copyright – only creation recorded on a tangible media can be copyrighted. Also traditional programs do not distinguish between different instances. When a program is executed a hundred times, the different copies should behave the same. However, an instance of a software agent will differ from another instance because the agents' behaviour is not fixed. An agent's beliefs (data collection) are constantly being modified by the environment and by its interactions with other entities. In turn these beliefs modify an agent's goal and the plans to achieve the goal.

When trying to solve the problem of how to "own" an active instance of a software agent, it may be possible to treat it as two separate issues:

When an agent is active in the system's memory, if the agent is designed to maintain ownership information and if it is possible to prove that agent is executing correctly, then it is possible to query the agent to obtain its owner's information. There is research to create software that is tamper-resistant. Aucsmith *[Aucsmith]* presents a software architecture in his paper that can dynamically inspect its integrity to ensure it is executing correctly. However, at this moment not all agents will maintain ownership information and it may be necessary to mandate such a requirement in standard likes FIPA.

When an agent is going into an inactive state, e.g. the suspend state in the FIPA agent life cycle or the transit state in the FIPA mobile agent life cycle, an agent may stay in this suspended animation for a long period of time. To maintain agent ownership over a long period of time, it is necessary to be able to capture an agent's internal states and be able to restore those states upon the agent's resurrection. In order to capture the internal states of an agent, the mechanism must be supported by the whole software system. Truyen, Robben, Vanhaute, Coninx, Joosen and Veraeten *[Truyen]* propose an extension to the existing Java model so that the internal states of a Java object can be captured and preserved. As Java is the underlining architecture for many software agent systems, this is a potential solution to the persistence problem. Once an agent is put into a suspended animation, the software data can be protected using traditional means: using encryption to maintain the secrecy and digital signature for ownership and integrity.

7. FIPA specification and Agent ownership

7.1. Current FIPA specification

In the FIPA 1998 specification, there were a series of fields in the AMS Agent Management description related to the ownership of the agent system. These include the ownership field in the Directory Facilitator description (fipa-df-description) to name the owner of an agent service; the ownership field in the platform profile (fipa-platform-profile) naming the owner of an agent platform and the ownership field in the Agent AMS description (fipa-agent-ams-description) denoting the legal entity responsible for the actions of the agent.*[AMS98]* However, these fields were declared obsolete in the subsequence releases of the FIPA AMS specification and replaced by the two optional ownership fields in the Agent management system agent description and Service description.

7.2. Proposed enhancements to the FIPA specification

The main motivation behind FIPA is to create a standard to allow heterogeneous multi-agent platforms to interact. FIPA specification has mandated essential issues such as agent life cycles, communication languages, communication protocols, interaction protocols and agent management system to facilitate the interactions. However, FIPA currently lacks a standard to express ownership information coherently in this open heterogeneous multi-agent system. There are three main areas that we suggest FIPA should address:

Human Agent Interaction

Currently there is no well-defined interface to convey ownership information between the human world and agent world. Since the withdrawal of the Human-Agent Interaction *[HAI98]* specification from FIPA 2000 specification, FIPA has removed all the links between human world and agent world. It may be necessary to reintroduce a specification to standardise the Human Agent Ownership Interaction. The FIPA Abstract Architecture may provide a suitable grounding for such specification.

FIPA should mandate a minimum set of requirements for legal agents. For example, a legal agent will need to be a legal entity; to be a legal entity it must be able to prove it has a well defined decision making structure. In other word, for a FIPA agent to become a legal agent, its action must be auditable (auditing service), its algorithms must be verifiable so that software error is accountable (through the agent developer ID) and its authority must be identifiable (through the agent user ID).

Standard Ontology

A standard ontology is needed to define the meaning of terminologies related to ownership e.g. platform owner, agent owner and agent user, etc. Efforts have been made to create standard ontology to describe trust *[UMBC]*, and indeed they have referred to the concept of "owner". However, the main objective of this ontology is for describing the trust relationship. It is necessary to extend the ontology in order to support the five principle entities in the proposed agent ownership life cycle.

Standard Protocols

Currently an agent's registration is initiated by the FIPA Request *[Request]* interaction protocol. However, this protocol does not support the concept of ownership. In order to support the registration of a legal agent, the protocol at least needs to be able to interact with the proposed Human Agent Interface to verify the ownership information. Furthermore, during agent interactions, an agent may need to query about another agent's ownership. Further studies and field trials are required to evaluate the effectiveness of the existing Query *[Query]* communication protocol in supporting this type of query.

8. Conclusion and Future work

This paper has raised a number of issues regarding software agent ownership. It has tried to address the issues from both legal and a technical point of view. In order to give a more thorough evaluation to these issues, it would be beneficial to conduct a field trial in an open agent environment. As the authors are members of the EU Agentcities.RTD project (IST-2000-28385), it is their hope to conduct such experiment by implementing an architecture to support agent ownership in an open heterogeneous agent network architecture provided by the Agentcities project.

This paper is also intended as a reply to the FIPA Security workgroup request for information, as a contribution towards the FIPA Security workgroup white paper.

9. Acknowledgments

The research described in this paper is partly supported by the EC project Agentcities.RTD, reference IST-2000-28385. The opinions expressed in this paper are those of the authors and are not necessarily those of the Agentcities.RTD partners.

10. Bibliography

[AMS98] Foundation for Intelligent Physical Agent, *FIPA Agent Management System Specification 1998*, http://www.fipa.org/repository/fipa98.html

[AMS00] Foundation for Intelligent Physical Agent, *FIPA Agent Management System Specification 2000*, XC00023F, http://www.fipa.org/repository/managementspecs.html

[Aucsmith] D. Aucsmith, Tamper Resistant Software: An Implementation, In Information Hiding, volume 1174 of Lecture Notes in Computer Science, pages 317--333. Springer-Verlag, 1996

[Busetta] P. Busetta & K. Ramamohanarao, *An architecture for mobile BDI agents,* Technical Report 97/16, The University of Melbourne, Department of Computer Science, Melbourne, Australia, 1997, http://citeseer.nj.nec.com/busetta97architecture.html

[Castel.] C. Castelfranchi & R. Falcone, *Socio-Cognitive Theory of Trust*, ALFEBIITE Deliverable report D1, 2001

[Collberg] C. Collberg & C. Thomborson, *Software Watermarking: Models and Dynamic Embeddings*, In Conference Record of POPL '99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (Jan. 1999), http://citeseer.nj.nec.com/collberg99software.html

[Dignum] F. Dignum, D. Morley, E.A. Sonenberg & L. Cavedon, *Towards socially sophisticated BDI agents,* In Proceedings of the ICMAS 2000, pages 111-118, 2000, http://citeseer.nj.nec.com/dignum00towards.html

[Georgeff] M. Georgeff, B. Pell, M. Pollack, M. Tambe & M. Wooldridge, *The Belief-Desire-Intention model of agency,* Proceedings of the 5th International Workshop on Intelligent Agents {V} : Agent Theories, Architectures, and Languages ({ATAL}-98), Springer Publishers, 1998, http://citeseer.nj.nec.com/georgeff99beliefdesireintention.html

[HAI98] Foundation for Intelligent Physical Agent, *FIPA Human Agent Interaction Specification 1998*, http://www.fipa.org/repository/fipa98.html

[Hansmann] H. Hansmann & R. Kraakman, *The essential role of organisational law*, The Yale Law Journal, Dec 2000

[Mamdani] A. Mamdani and J. Pitt, *Responsible Behaviour for Networked Agents – A Distributed Computing Perspective*, IEEE Internet Computing, Vol. 4, No. 5, Sept./Oct. 2000, pp. 27-31.

[MIAMI] Deliverable report D5, *Revised MIAMI Specification and Evaluation*, Mobile Intelligent Agent for Managing Information Infrastructure, EU ACTS project (AC338), http://www.fokus.gmd.de/research/cc/ecco/miami/public/doc

[Poslad] S. Poslad and M. Calisti, *Towards improved trust and security in FIPA agent platforms,* Proceedings of AA2000, Barcelona, Spain, June 2000

[Query] Foundation for Intelligent Physical Agent, *FIPA Query Communication protocol specification*, XC00027, http://www.fipa.org/repository/ips.html

[Rahman] A. Abd ul-Rahman & S. Hailes, *A Distributed Trust Model,* New Security Paradigms 97, http://citeseer.nj.nec.com/347518.html

[Request] Foundation for Intelligent Physical Agent, *FIPA Request Communication protocol specification*, XC00026, http://www.fipa.org/repository/ips.html

[Sander] T. Sander & C.F. Tschudin, *On Software Protection via Function Hiding,* Submitted to the 2nd International Workshop on Information Hiding, http://citeseer.nj.nec.com/sander98software.html

[Sartor] G. Sartor & E. Pelino, *Software Agents and the Law*, an ALFEBIITE project presentation in the Agentcities.NET meeting, Lausanne Feb 2002

[Security] FIPA Security Workgroup, *FIPA Security Workgroup Request For Information*, http://www.fipa.org/activities/security.html

[Truyen] E. Truyen, B. Robben, B. Vanhaute, T. Coninx, W. Joosen & P. Verbaeten, *Portable Support for Transparent Thread Migration in Java*, Proceedings of ASA/MA2000, http://www.cs.kuleuven.ac.be/~tim/MOS/brakes.html

[UMBC] UMBC Trust ontology, http://www.daml.org/ontologies/182