Artificial Intelligence and Law (2004) 12: 5–37

Law-abiding and integrity on the Internet: A case for agents*

FRANCES BRAZIER[1], ANJA OSKAMP[2], CORIEN PRINS[3],
MAURICE SCHELLEKENS[3] and NIEK WIJNGAARDS[1]
[1]*Intelligent Interactive Distributed Systems, Faculty of Sciences, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands*
E-mail: {fmt.brazier, nje.wijngaards}@few.vu.nl
[2]*Computer and Law Institute, Faculty of Law, Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands*
E-mail: a.oskamp@rechten.vu.nl
[3]*TILT, Tilburg Institute for Law, Technology and Society, Faculty of Law, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands*
E-mail: {j.e.j.prins, m.h.m.schellekens}@uvt.nl

Abstract. Software agents extend the current, information-based Internet to include autonomous mobile processing. In most countries such processes, i.e., software agents are, however, without an explicit legal status. Many of the legal implications of their actions (e.g., gathering information, negotiating terms, performing transactions) are *not* well understood. One important characteristic of *mobile* software agents is that they roam the Internet: they often run on agent platforms of others. There often is no pre-existing relation between the "owner" of a running agent's process and the owner of the agent platform on which an agent process runs. When conflicts arise, the position of the agent platform administrator is not clear: is he or she allowed to slow down the process or possibly remove it from the system? Can the interests of the user of the agent be protected? This article explores legal and technical perspectives in protecting the integrity and availability of software agents and agent platforms.

Key words: software agents, integrity, liability, legal aspects, agent migration, agent platforms

1. Introduction

New technologies on the Internet have their impact on society and thus on the law. In the field of AI and Law the aims have always been to develop new intelligent tools to support legal practice but also to construct underlying theories to ensure that these tools can be applied in a way that is in accordance with the relevant legal regime. Agent technology is an example of a new

* This paper is an extended and fully revised version of Brazier et al. (2002a) and Brazier et al. (2003a).

technology that may have great impact on the construction of applications for the legal domain, especially now so many legal sources and legal services are appearing on the Internet. Software agents may, for instance, be of great help to keep track of and analysing new legal publications, e.g. e-Government, where software agents may help to perform legal services. E-commerce is another domain with relevant examples (e.g., Bradshaw 1997; Luck et al. 2003).

At the same time the potential application of these new technologies raises many questions. The Internet society is not restricted to geographical, legal, or corporate boundaries. The physical distribution of services, processes, and data, is no longer necessarily the same as the perceived location Agent technology is a key enabling technology in this Internet society; software agents are autonomous and pro-active, can autonomously (e.g., Castelfranchi 2001) roam the Internet, perform transactions, and gather information. Wide-area distributed (agent) systems are no longer fiction, they are a fact. For example, peer-to-peer networks in use today (e.g. Napster, KaZaa, Jabber, Gnutella,...) demonstrate the impact of large-scale distributed (file-sharing) applications on society. Regarding agents, the Agentland portal (http://www.agentland.com) is an example of a website offering free and paid-for agent applications for various tasks, including information retrieval and web-site management. Existing forums such as W3C (http://www.w3c.org/) and FIPA (http://www.fipa.org/) propose standardisation to pave the way for the development of agent applications. The Agentlink Roadmap (Luck et al. 2003) states the expectation that custom-made, specific, closed agent-based systems currently being deployed will become less constrained, e.g. by coping with more dynamic environments, in 5–10 years time, after which more open systems will be deployed.

A number of issues need to be clarified before software agents and related technology can be used on a large (world-wide) scale in open systems such as the Internet. In the context of current thinking on the relationship between technology and law, the legal status of software agents is unclear. This article focuses on the legal status of software agents in closed systems as a current testing ground for potential solutions, identifying and exploring challenges for fully open systems.

When considering the possible legal challenges and problems in more detail, we note that multiple prominent legal concerns arise in relation to the use of agents, such as: liability, security, privacy and contractual implications. These and other legal challenges facing users and producers of agent technology as well as regulators are compounded by the fact that intelligent agents are marked out by various unprecedented features. Traditional legal paradigms do not suffice (van Klink and Prins 2002). The well-known paradigms have all developed along the lines of intervention by natural or legal persons within local bounds of space and time. However, these confines have lost their meaning in a world that is characterized by a society in which autonomous and anonymous communication and interaction by machines is flourishing.

The complex issue of agent technology thus introduces a variety of challenges and opportunities that span the domains of agent systems applications on the one hand and law on the other hand. Within the individual domains of technology and law, the implications are known to some extent. For example, in the context of e-commerce: issues such as liability and contract formation (Karnow 1996; Stuurman and Wijnands 2001; Geurts 2002). Yet experiences, needs and requirements from novel applications (such as mobile agents) have not yet been dealt with from a legal perspective. Moreover, what is missing is a thorough and overall evaluation of the relation between technical perspectives and interests on the one hand and legal perspectives and interests on the other hand. The interaction between agent development and legal dimensions is mostly unknown, yet technical progress warrants the need for legal clarification, and legal implications may require resolution of technical issues.

Agent technology refers to not only software agents, but also agent platforms: supportive middleware which enables the functioning of software agents (see Section 2.2 for examples and explanations). Both software agents and agent platforms are discussed in this article because of their legal and technical interdepencies. Section 2 sketches how software agents and agent platforms may operate, and what risks there are related to integrity and availability. Section 3 explores various aspects of the protection of the integrity of software agents' processes and agent platforms. It investigates which legal requirements exist regarding integrity and whether the law mandates protection of the integrity of agent platforms and running software agents' processes, and if so how and when. It also considers the rights and obligations of agent platform owners. Technical measures are explored that can be taken to protect the integrity of both the software agent and the agent platform and which counter measures are possible. The relevant legal questions regarding availability and integrity are dealt with from the perspective of Dutch law, a statutory legal system, in contrast to the Anglo-Saxan/American case-law system. Section 4 suggests several legal recommendations for the use of software agents and agent platforms. Section 5 concludes this article with a discussion.

2. Software agents and agent platforms

Agents are used in a wide variety of applications (e.g., Jennings and Wooldridge 1998): process control, manufacturing, air traffic control, information management, e-Commerce, business process management, patient monitoring, health care, games, and interactive theatre and cinema. The role of agent technology is expected to grow in the foreseeable future (Luck et al. 2003). Currently, software agents are used in restricted applications involving no outside parties or agents, with the expectation that in 5 years time agents may be involved in cross-boundary systems, leading to the use of agents in fully open systems in 10 years time. It is important to note that agent technology involves agents and agent platforms. *Agents* are usually envisioned as ‘‘active entities’’, while *agent platforms* constitute a technical (software) infrastructure to support agents. Their responsibilities differ: agents pursue goals for user applications, while agent platforms manage agents and their needs for resources.

A number of different parties are involved in the use of software agents and agent platforms, including users, owners, distributors, developers, and designers. Although distinguishing (legal) parties is important for, e.g., litigation (e.g., Biasiotti et al. 2003), it is not of particular importance for the purpose of this article. When in this article the ‘‘owner’’ of an agent's running process is mentioned, the word ‘‘owner’’ denotes the person or organisation who, other than a system administrator, has control over the running agent. Often this may be the person in whose interest the agent runs. The word ‘‘owner’’ is thus not used in its legal meaning when used with respect to agents or agent processes (e.g., see Yip and Cunningham 2003). Here, for practical purposes we assume the user of a software agent to be the owner, whereby the owner of an agent platform is represented by system administrators.

An example is used throughout this article to facilitate the explanation of agents and agent platforms, involving a *marketplace* in which books can be bought and sold. The marketplace is created so that multiple companies can cooperate to provide a single place for buyers to buy books. For ease of explanation, participants are either sellers or buyers, but not both. Each seller has his or her own booth to display books, advertisements, engage in conversation and negotiation with potential buyers, etc. A seller may represent a specific company and/or a theme: thriller, fantasy, computer, law, comics, etc. A company may use multiple seller agents. A central cashier is present for buyers to buy books for the agreed price and conditions, possibly including book transportation arrangements. Floor plans and catalogues are present so that buyers can navigate the marketplace and browse titles to find books to their liking.

This section is structured as follows. Section 2.1 describes agents, both as individuals and as an aggregate (a multi-agent system). Section 2.2 explains agent platforms, and the services they provide to agents. Section 2.3 identifies a number of risks involved with the use of agents and agent platforms.

2.1. WHAT IS AN AGENT?

An important contribution of agent technology is the following metaphor: the ‘‘agent’’ concept facilitates modelling inherently distributed processes and systems as autonomous pro-active processes (e.g., Luck et al. 2003). A number of definitions of software agents are provided in Artificial Intelligence literature (e.g., Shoham 1993; Wooldridge and Jennings 1995; Nwana 1996; Bradshaw 1997; Jennings 2000) in which a common element is that agents are autonomous and pro-active and are able to:

- communicate with other agents, e.g. to co-operate in a joint-endeavour, negotiate on prices for goods and services, or collaboratively form decisions;
- interact with their environment, e.g. with data-objects and web-services; and possibly to
- migrate to specific agent platforms (see below in Section 2.2.3), e.g. to bring the computations by an agent close to large repositories of data, thereby preventing unnecessary data-transmission and facilitating secure access.

Pro-activeness and autonomy are related to an agent's ability to reason about its own processes, goals and plans. The ability to communicate and co-operate with other agents and to interact with the outside world often relies on an agent's ability to acquire and maintain its own knowledge of the world and other agents.

Agents often work for a human user, directly or indirectly via other agents. In some definitions, humans are also considered to be "agents", thereby unifying and anthromorphising the notion of agency. A rationale for this modelling perspective is that each pro-active autonomous (and possibly intelligent) entity is modelled as an agent and takes part in the multi-agent system by means of pre-defined communication and interaction protocols. A "human" agent is often "implemented" as a software programme with graphical (web) interfaces, through which a human can participate in a multi-agent system. In the context of this article, an agent is assumed to be a *software agent*, not a human. When humans wish to communicate with an agent, they can employ, e.g., facilities such as e-mail or instant messaging.

In the example of the marketplace, participants can be modelled as agents: each participant is autonomous, and can start, continue and/or stop a transaction with other participants. A participant is able to interact with the environment, e.g. to consult a floor plan to locate specific booths at the marketplace. The participant may move to a booth to inspect and trade in locally available books, an action which was not possible from another location outside of the booth. The only way to influence a participant is by communication or interacting in the environment. (Note that a move to a booth may constitute moving to another agent platform, depending on the design of the underlying infrastructure (see Section 2.2.3).)

Agent communication languages (ACLs) describe "speech acts" in the form of message exchanges, e.g. KQML (Finin et al. 1997) and FIPA-ACL (Dale and Mamdani 2001). The semantics of a message in an ACL is generally left open (e.g., Dale and Mamdani 2001) to the extent that it is associated with an ontology, e.g. the Semantic Web effort (Berners-Lee et al. 2001; Ding et al. 2002) develops ontologies by which machines may understand information and overcome an interoperability problem (Jennings 2000). An ontology is, generally stated, a description of the concepts and relations that can be used to represent (part of) a domain.

In the example marketplace, languages, protocols and ontologies are standardised by the marketplace organisation, forcing compliance of seller agents and buyer agents and facilitating inter-agent communication and object/service interaction. Figure 1 depicts interaction (dashed lines) and communication (straight lines) in a simple protocol for buying a book. A buyer agent interacts with a book-catalogue object (Figure 1a) to select a book to be bought. The buyer agent then asks the associated seller agent for the book's price (Figure 1b). The seller agent, in turn, replies with information necessary to complete a book sale (Figure 2c). Finally, the buyer agent interacts with the cash register to buy the book. This rather simple protocol does not include more realistic aspects such as negotiation on price and delivery conditions and avoids issues such as security, privacy and copyright (e.g., Bing 2003).

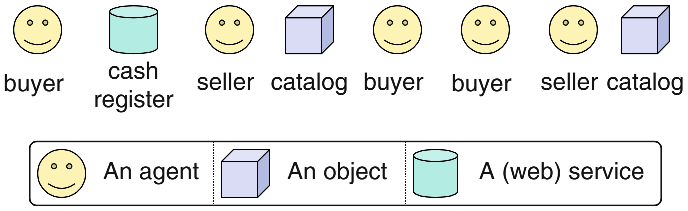

Figure 1. Interaction and communication involved in a simple protocol for buying a book.

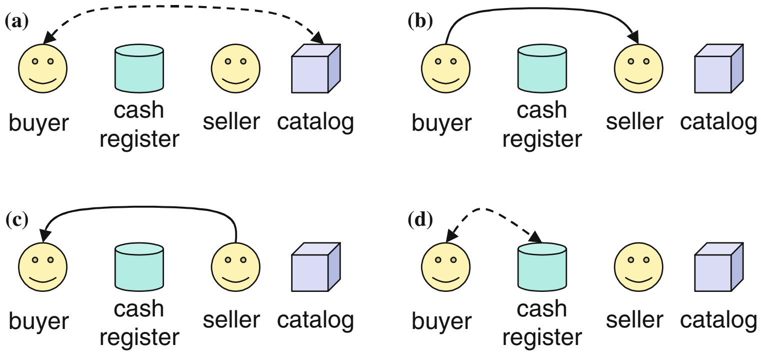

Figure 2. Agent platforms support agents, objects and web-services; in turn agent platforms are supported by one or more computer systems.

In the example marketplace, a book-catalogue is modelled as an object: it can be inspected by agents, but the catalogue itself does not actively contact agents. The floor plan in the example, however, is capable of alerting buyers to potentially interesting nearby sellers, based on the current position of a buyer. The floor plan is considered a *service*: an active element supporting the functioning of agents without itself being an agent, as the floor plan cannot buy or sell books. The cash register's functionality is a web-service, one that can e.g. initiate credit card payments for the associated book-selling companies. Note that this distinction between agents, objects, and (web)services is *conceptual*; when programming a software agent often object-oriented programming languages are used (Java is a well-known example). A software agent does not differ in this respect from other software on a computer: when it "runs", it is considered to be a process (Tanenbaum and Steen 2002) with code (programming instructions), data (comparable to a human's "long-term memory") and execution state (comparable to a human's "working memory").

2.2. WHAT IS AN AGENT PLATFORM?

Middleware in general, and agent platforms in specific, provide the means to hide differences in underlying computers (and, e.g., their operating systems such as Unix, Windows XP, Mac OS X, Linux, Palm OS, etc.) by offering a more standardised interface to agents (and their developers). Examples of agent platforms include FIPA (FIPA, 2001), Jade (Bellifemine et al. 2001), ZEUS (Nwana et al. 1998), and AOS (Wijngaards et al. 2002). The FIPA interface is currently being standardised as a "de facto" standard in the form of prevalent Jade implementations (AgentCities 2003). However, interoperability between different types of agent platforms is *not* simple, even with the FIPA/Jade grassroots standardisation.

When discussing agent platforms, a number of aspects need to be addressed. First of all, an agent platform typically manages agents (Section 2.2.1), provides communication and interaction services (Section 2.2.2) and offers migration services (Section 2.2.3). Finally, deployment of an agent platform in the real world needs to be addressed (Section 2.2.4).

2.2.1. Managing agents

An agent platform manages the allocation of resources to agents, e.g. such as which agent "runs" on which computer (if a choice exists) with what speed and which services (e.g., http-access to remote web-sites, or access to a local database). In addition to hosting agents, an agent platform creates agents, deletes agents, searches for agents, etc. An agent platform provides mechanisms for security, including access control for migrating agents as well as defence against possible malicious, or faulty, agents. In the example marketplace, seller agents commonly reside on companies' agent platforms, and buyer agents are more dynamic; they can enter the marketplace, and possibly migrate to and from a company's agent platform. This implies that the marketplace agent platform needs to manage migration of buyer agents together with access policies for buyer agents entering the marketplace, manage address information of buyer and seller agents (for inter-agent communication purposes), providing computer resources to buyer agents, etc.

Policies for access control, i.e. deciding whether an agent is to be hosted by a specific agent platform, are usually based on the agent's identity, its owner, or other characteristics. In addition, information on reputation or gossip, and observations of the (expected) load of the agent platform itself may influence these policies. Trust, a multi-facetted concept (McKinght and Chervany 2001), is often based on reputation and gossip information. Trust plays an important role in agent systems, involving trust between and among agents and agent platforms (Wong and Sycara 1999). Access control for agents may involve the exchange of credentials; for example, the Trustbuilder system (Winslett et al. 2002) is used to negotiate which resources agents may access, based on both specific agent's and Trustbuilder systems' policies and credentials. In the example marketplace, access policies for entering the marketplace for the first time can be based on a buyer agent's credentials including producer of the agent, and the human user's reputation for payment and rebates. In addition, the marketplace may use experiences of another marketplace, e.g. in which CD's are sold, about trustworthiness and (mis)behaviour of users.

Additional policies are needed, for example to govern agent creation: to determine whether an agent may create another agent or clone itself. Issues concerning ownership, identification and authentication as well as liability still need to be addressed.

Anonymity is another issue: are anonymous agents allowed? Complete anonymity may hamper communication facilities (no addressee or recipient are known, bulletin-board communication approaches may be an alternative option) and pseudonyms may be confusing when agents migrate to other agent platforms where different pseudonyms are used. The subject of anonymity and software agents is discussed in more detail elsewhere in this special issue, based on (Brazier et al. 2003a). In the example marketplace, anonymity is not an option: buyer and seller agents in the marketplace need to be identifiable for communication, management and migration purposes. However, the identity of the human user is only needed (if at all) when completing a transaction at the cashier service. A user can choose to reuse their buyer agent, or have new buyer agents (with different identities) enter the marketplace.

2.2.2. Agent communication and environment interaction

An agent platform provides services to agents, including communication, interaction, and directory services. Communication services may differ in their "quality of service". For example, e-mail, as used by humans, is an unreliable service as messages may get lost or delivered late, because a guarantee of delivery is missing. Similarly, communication protocols for agents may have different costs (in terms of bandwidth, number of messages sent, time taken, etc.) and characteristics such as reliability and security.

In a similar vein, interaction protocols are defined with which agents can interact with objects and (web-)services. Protocols may include message-based protocols and streaming-protocols (e.g., a continuous flow of information from a sensor or audio station). Standardisation is currently not widely researched. In the example marketplace, a service offered to buyer agents may consist of an information stream of advertisements (similar to stock-tickers, e.g. via RSS) for a buyer agent. Likewise, seller agents may wish to have a continuous update of which buyer agents are currently interested in specific issues, e.g., by monitoring the behaviour of buyer agents browsing through book catalogues.

Agents and services can be found with directory services. Directory services couple names of agents and services to characteristics, such as abilities to perform certain tasks. In the example marketplace, yellow pages can be used to find seller agents with specific abilities, such as a seller agent specialised in literature on agent technology and law. A directory service can be used to provide such yellow page functionality. Note that directory services in large distributed systems in which many information updates occur, is currently a topic of research (Tanenbaum and Steen 2002).

Agent platforms need to extend communication services to agents on other agent platforms, as communication is usually an inter-agent-platform service. For this to work, agent platforms also need inter-agent-platform protocols and languages. FIPA provides some standardisation (FIPA 2001) but additional standardisation is required. Whether agents can interact with objects and services which are hosted on different agent platforms, usually depends on the configuration of both the local agent platform and remote agent platforms. If no remote access is allowed, an agent may migrate to the other agent platform, or have other agents migrate (as "helper" agents). In the example marketplace, the agent platforms involved in supporting agents participating in the marketplace facilitate inter-agent communication. This is similar to humans using cell-phones: cell-phone usage is (usually) transparent with respect to the specific telecommunication companies involved as they resolve the underlying connections for relaying speech.

2.2.3. Agent migration

Agents, objects and services may be distributed over a large number of agent platforms. Agents may migrate from one agent platform to another: they are mobile. A common reason for mobility involves the quality of access to resources such as large volumes of data and security constraints (Brazier et al. 2002b). An agent may gain faster access to data when an agent resides on the same agent platform as the data, instead of a remote agent platform with restricted (inter)network capacity (e.g., a modem connection offers less bandwidth than an Ethernet or DSL connection).

In the example marketplace, a compelling reason for buyer agents to migrate to the marketplace's agent platform is security: the migration policy involves stringent checks of, e.g., owner reputation, agent software reliability, and protocol compatibility. Agents which successfully entered the marketplace, can then more easily migrate to a company's agent platform, as all stringent checks have been performed earlier. A reason to migrate to a company's agent platform may be to do a full text search on books for sale at that company. A full text search is an expensive operation, especially if the buyer's agent contains specific evaluation criteria or algorithms to assess a book's usefulness. By migrating this buyer's agent to the company's agent platform, digitally encoded books need not leave the company's computers and network, thereby preventing possible misappropriation and reducing the amount of data shared over the Internet.

Figure 3 depicts a number of the steps involved in migrating an agent. Suppose a user has acquired a buyer agent on his or her mobile phone (Figure 3a). This buyer agent is instructed to enter the book marketplace and to buy a specific book. The user's mobile phone contains a (small) agent platform, which starts negotiations with the marketplace's agent platform (Figure 3b). These negotiations may include checking of credentials and authentication steps. When all is in order, the user's agent platform suspends the user's buyer agent, and places the transportable format of the agent in an "agent container" (see section 3.2.1.): this basically is a data-structure (such as a ZIP-file) containing relevant files for an agent and its associated process(es) (Figure 3c). The buyer's agent is currently not functioning; policies may be provided, e.g., to store messages for this agent until the agent is "re-activated". The user's agent platform transports the suspended buyer agent's container to the marketplace's agent platform (Figure 3d). The marketplace's agent platform is responsible for resuming ("awakening") the buyer agent on the basis of information in its agent container. The buyer

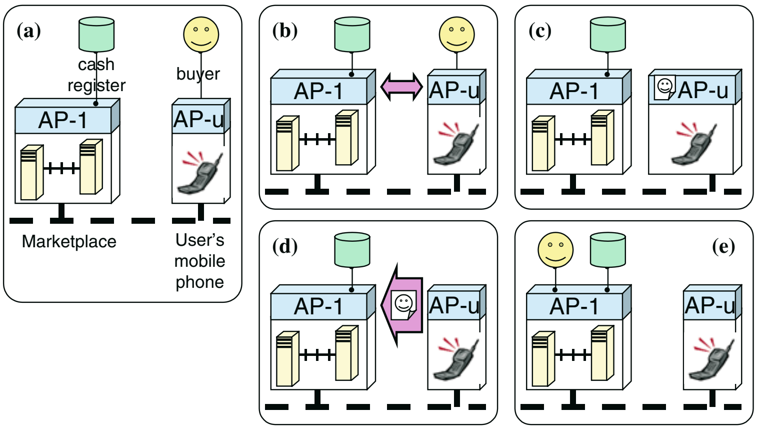

Figure 3. Example of migrating an agent; see the text for an explanation.

agent is now hosted by the marketplace agent platform and resumes functioning (Figure 3e). Both the user's agent platform and the marketplace's agent platform have to make certain that the buyer agent is also registered as being hosted at the marketplace's agent platform, e.g. with respect to routing communication messages. After completion of the migration protocol it no longer necessary to maintain a connection between the marketplace's agent platform and the user's agent platform.

Migrating an agent entails migrating its process involving code, data and execution state. A number of options exist (Fugetta et al. 1998). In one approach, an agent may not need to be aware of migration, e.g., to store important information in its agent container. The agent may continue with its functioning and it typically does not need to be "restarted" at its destination. This type of migration, transparent to the agent, is termed *strong mobility*; the agent's code, data and execution state are migrated. Strong mobility as found in NOMADS (Suri et al. 2000), Ara (Peine and Stolpmann 1997), and D'Agents (Gray et al. 2001), requires that an entire agent, including its execution state and other details, is saved before an agent is migrated to a new location: this requires homogeneity of operating systems, agent platforms and programming languages.

The opposite is *weak mobility*, in which only an agent's code and data are migrated, but its execution state is discarded. In this alternative, an agent is always "restarted" at its destination, and it needs to store important information from its execution state in its data to be able to resume its functioning from the moment it migrated. Many agent systems support weak mobility (like Ajanta (Tripathi et al. 1999) and Aglets (Lange et al. 1997)). Most of the agent systems are implemented using the Java programming language and the Java Virtual Machine (JVM), which provides object serialization as a basic mechanism to store execution state information as data.

2.2.4. Deployment

Security is of key importance for deployment of agent systems. Agents need to be protected during migration and execution. Agent platforms, however, also need to be protected from malicious and malfunctioning agents. Current research on secure agent systems concentrates mainly on protecting agent platforms (and the computers on which they run) against hostile mobile agents. Only very few systems also provide facilities for protecting mobile agents against hostile agent platforms (Karnik and Tripathi 2001). The example marketplace spans agent platforms from multiple companies and possible users. Their underlying computers, operating systems and other resources may be heterogeneous, yet security and reliability needs to be assured otherwise companies and users will not participate. Section 2.3 outlines possible risks of the use of agents and agent platforms.

An agent platform may support agents from different multi-agent systems, although this is currently not commonplace. Existing agent platforms such as ZEUS (Nwana et al. 1998), NOMADS (Suri et al. 2000), Sensible Agents (Barber et al. 2000), RETSINA (Sycara et al. 2003), JADE (Bellifemine et al. 2001), OAA (Martin et al. 1999), MASIF (Milojicic et al. 1998) and Ajanta (Tripathi et al. 1999) are approaches in which tools for developing agents are combined with agent platforms. Basic facilities such as communication, creation and deletion of agents are provided by most agent platforms; mobility is often limited. AgentScape (Wijngaards et al. 2002) is a mobile agent platform designed to support large numbers of heterogeneous agents and locations, securely and reliably. Aspects such as interoperability, efficiency and performance, but also security, are part of the current research to which the OpenNet project (formerly the AgentCities (2003) project) contributes by providing an infrastructure for testing agent systems. An open collection of FIPA compliant agent platforms is created on which multiple multi-agent systems can reside. An agent platform thus needs to manage multiple concerns, e.g. from local agents (and their users) as well as unknown external agents (and their users).

Successful deployment of agent systems requires scalability: very large numbers of agents, services and resources require adequate support by agent platforms. Large-scale agent systems are often heterogeneous systems: heterogeneous with respect to the computer hardware (e.g., Sun SPARC, Intel x86/i64), the computer operating system supported (e.g., Solaris, Linux, Windows XP), the agent programming languages support (e.g., C++, Java, Python, and Prolog) and the communication infrastructure (e.g., available qualities of service). An agent platform should provide a homogeneous, transparent interface to agents. A major technological challenge is to build scalable, secure, and fault tolerant agent platforms, that support multiple distributed applications, heterogeneity, and multiple qualities of services (e.g., De Wilde et al. 1999; Turner and Jennings 2000; Brazier et al. 2001).

2.3. WHAT ARE KNOWN RISKS?

Application of any technology has consequences, some of which are foreseen and wanted, others may be unwanted. Agent technology is not different in this respect: its application may involve certain risks. Below, first risks of using agents are described, then risks of using agent platforms. Section 3 introduces two relevant cases from Dutch and American law to highlight that these risks are real, and can already happen. The first case (XS4All versus ABFAB) is about safeguarding users and maintaining system integrity, while the second case (eBay, Inc. versus Bidders' Edge, Inc.) is about access policies.

2.3.1. Risks when using agents

As mentioned, using agents is not without risk. Agents are expected to function ‘‘autonomously’’, without a user's constant supervision. Agents run on an agent platform, communicate with multiple agents on the Internet and interact with numerous objects and (web)services. It is most often unknown to the user which agents, objects and services are involved. Mobility increases the risks an agent faces: agents may be hosted by a potentially malicious agent platform, outside the control of their user and thus become more vulnerable (e.g., Brazier et al. 2003b). The underlying problem is that other parties, such as agent platforms, other software agents, humans, virus applications, etc., can change or copy the code and/or data of an agent, either intentionally or accidentally. This may have far-reaching consequences and entail great risks to users. For example, an unwanted modification of an agent's code and/or data may influence an agent's behaviour significantly. Changing data especially confidential information (e.g., bank balance, bank accounts) or data with an economic value may be sufficient to cause significant damage. In addition, instructions and goals of an agent can be modified. An agent may then exhibit unwanted behaviour, for example by buying goods for exorbitant prices, buying multiple items, engaging in malicious behaviour, etc.

In the marketplace example, a buyer agent may be modified, e.g., by a book-selling company, by raising the buyer's agent maximum price increasing the likeliness of the agent purchasing one or more books. Similarly, a seller agent may be modified to provide large discounts to specific buyer agents. Catalogues may be tampered with to display higher prices for competitor seller agents, thereby stealing their (potential) business. More severe consequences for the entire marketplace can, e.g., be caused by a change in the code of a buyer agent which may cause the agent to engage in many bogus interactions to the cashier service, frustrating the functioning of the marketplace for other agents by means a kind of "denial of service" attack.

Another issue is stealing information, including private information such as addresses, bank accounts, credit card numbers, and an agent's internal (strategic) aims and instructions. This information can then be used for malicious purposes not necessarily related to the agent's application. For example, electronic banking information in buyer agents may be used to acquire cash advances or goods for third parties in a different country, thereby causing financial losses to users. Alternatively, a seller agent may make use of knowing the instructions of a buyer agent, to predict the buyer agent's behaviour and attempt to steer the buyer agent in certain directions. For example, by knowing the criteria for establishing the maximum acceptable price, the seller agent knows the "optimal" price to aim for.

In addition, an agent may disappear temporarily or permanently, possibly representing a loss of investments in training and acquisition of the agent as well as frustrating the acquisition of results from the agent. Depending on the severity of the damages caused by a misbehaving agent, the agent's reputation may be jeopardised, as well as their user's or owner's reputation. In addition, a user or owner may face litigation. The extent of possible damages depends on the agent and the specifics of the context. For example, corruption of a single agent in a swarm application (i.e. an application with large numbers relatively simple agents used to examine emergent behaviour) may have a negligible effect. In the example marketplace, the user of a corrupted, misbehaving, buyer agent may be banned, although the user was not (directly) responsible for the agent's misbehaviour. The functioning of the marketplace is not frustrated when one user and associated buyer agents are removed. Whether such policies are fair to users and constitute viable business practices remains to be seen.

2.3.2. Risks when using agent platforms

Using agent platforms is also not without risk. An agent platform hosts an agent: the code and data of which validity, reliability and trustworthiness cannot be easily determined automatically. Agents compete for an agent platform's resources (such as processor cycles, bandwidth, disk space) and services (such as (reliable and secure) communication, mobility, automated updates, streaming connections). Malicious agents may succeed in migrating to a trustworthy system.

Code and/or data of an agent platform can be modified or stolen, by mistake or by design. Possible consequences include modifying or stealing information on agents or conditions for (dis)allowing access to migrating agents, or management policies, etc. An agent platform may then exhibit unwanted behaviour, e.g. by allowing unknown agents to enter, assign or remove resources to agents, or delete agents.

In the example marketplace, stealing information on buyer agents, as well information about buyer agents such as their human users, characteristics, etc., can be used to corrupt the marketplace by modifying internal data of agent platforms, e.g. causing buyer agents to appear as seller agents, etc. In addition, stolen information can be used for other malicious purposes, including attacking related multi-agent applications which trust the book-marketplace or trust specific human users. By changing information in agent platforms in the marketplace also rather different behaviour may appear. For example, specific agents may be killed entirely, or migrated to non-trustworthy, non-affiliated systems of companies out to gain access to the actual buyer agents and their private information.

An agent platform may also cease to function, with possible financial losses. Damages to agents can be extensive and difficult to determine. Depending on the severity, the reputation of the agent platform, and its users and/or owners, can be jeopardised. In addition, a user or owner may face litigation. If the example marketplace is attacked, and causes severe damage, subsequent usage of the marketplace may cease, causing large (financial) damage to involved parties.

3. Agent and agent platform owners living together uncomfortably

The previous sections describe the key specifics of agents and agent platforms. In summarising these risks it is clear that the key concerns relate to unwanted modification of code and data, stealing of code and data, and interfering with functioning of agents and agent platforms. Whether this occurs by design, by mistake during functioning of agents and agent platforms, or by malicious intent, the integrity and availability of agents and agent platforms is at stake. Given this, Section 3.1 discusses the concepts of availability and integrity with respect to agents and agent platforms. Section 3.2 addresses relevant technical issues, and Section 3.3 subsequently addresses relevant legal issues. Finally, Section 3.4 analyses two relevant legal cases with respect to availability and integrity of agents and agent platforms.

3.1. AVAILABILITY AND INTEGRITY IN AGENT SYSTEMS

In computer systems in general, availability and integrity are important features. From a technical perspective, integrity is strongly related to preventing improper alteration of a system (and its assets), where an alteration should be detectable and recoverable, whereas availability is strongly related to fault-tolerance and reliability (Anderson 2001; Tanenbaum and Steen 2002).

From a legal perspective, the integrity of a computer system is protected under both criminal law and civil law. Civil ownership provides a basis to act against infractions of integrity. Criminally, a computer system is considered to be a physical object, the damaging of which is a criminal offence if it is done intentionally and without right (see art. 350 Dutch Criminal Code). For an object to be damaged, it is sufficient that it has ceased to function (Dutch Supreme Court 19 October 1971, NJ 1972, 33); it need not be physically damaged. For example, erasing relevant parts of a computer's operating system damages the system; it may no longer function properly. Erasing a single Word-document from a computer's hard disk does not.

The foregoing suggests it is possible to affect the integrity of data without affecting the integrity of the physical carrier of the data. This raises the question of the integrity of data as such is protected in law. The integrity of data is protected separately in both criminal and civil law. In civil law the affection of the integrity of data may constitute an unlawful act. In criminal law the integrity of data as such is the subject of specific criminal offenses (see art. 350a and 350b Dutch Criminal Code). The Cybercrime Convention of the Council of Europe provides an example of the statutory provision that is representative of the national legal provisions protecting availability and integrity of data.

Article 4.1 Convention of Cybercrime (CoC) deals with data interference:

> Article 4 – Data interference
>
> 1. Each Party shall adopt such legislative and other measures as may be necessary to establish as criminal offences under its domestic law, when committed intentionally, the damaging, deletion, deterioration, alteration or suppression of computer data without right.

In the marketplace example, it is obvious that the availability and integrity of the code of agents and agent platforms is of major importance. Deficiencies in the availability and integrity of the code of an agent or agent platform may lead to many different problems. Messages may not be received or they may be received multiple times. Deadlines – e.g. for accepting an offer – may not be met. Potential buyers or sellers may not be visible to other participants in the marketplace. The list of potential problems is almost innumerable. It is also clear that both agents and agent platforms are in this respect interdependent. The owner of an agent may go a long way to ensure integrity of his agent, but this is to no avail if the agent must function on an agent platform that suffers from deficiencies. The reverse also holds for ensuring integrity of agent platforms and potentially deficient agents. Apart from the integrity of code also integrity of data is relevant. This may regard both data as part of a message being sent between different entities as well as information stored in databases. Since data are directly relevant for the contents of the transactions that take place on the marketplace, it is not difficult to envisage damages that could result from deficiencies in the integrity and availability of data.

3.2. TECHNICAL ISSUES

When considering the relevant technical issues, the same two issues are of importance: integrity and availability of both agents and agent platforms.

3.2.1. Availability and integrity of agents

An agent must be protected from unwanted alterations, caused by system malfunctions or malicious entities, e.g. other agents or agent platforms. From a legal perspective it is desirable that the facts can be reconstructed "afterwards", i.e. after something has happened that influenced the availability and integrity of an agent. For this purpose, an agent may be obliged to keep a trace of information (including orders) received from users and possibly from other agents, together with its responses.

Fortunately, possible (counter)measures exist against integrity violations. From a technical perspective, encryption can be used to secure information, although transportation of encryption keys by migrating agents still poses a dilemma to researchers. Another technique is to use secure change logs in agents: digitally signed entries in a log file containing the signatures of all involved parties (Noordende et al. 2002). Such change logs can be used to determine what happened to an agent; an important aspect of litigation cases.

Techniques to detect improper modifications are mainly based on tracing alterations to an agent. A mobile agent may need to carry these traces during its life span if agent platforms "en route" cannot be trusted. One technique is to use an encrypted agent container (Karnik and Tripathi 2001; Noordende et al. 2002) as the unit for transportation of an agent and its associated information, as alterations are securely logged. Alternatively, an agent may regularly visit a trusted third party for an integrity check or frequently send messages back to its owner reporting on its status. This may ensure continuing a user's trust in agents. Trust and reputation play an important role for agents, with respect to both other agents (whom to buy a book from?) and agent platforms (which will treat me fairly?). Techniques to manage trust in large open multi-agent systems are being researched (e.g., Wong and Sycara 1999; McKnight and Chervany 2001).

With respect to availability, techniques to recover from improper alterations are usually based on restoring information from a copy (a "backup") at a secure location. Techniques differ in their guarantees with respect to the relevance of the back-upped information (e.g., the more recent, the better) and whether other agents may notice that a specific agent is recovering. An agent container, mentioned before, is one means for an agent to frequently backup its data (and code if changed). In other techniques, agents may have fault-tolerance (e.g., via backups) built-in which may better guarantee their availability as they may recover more easily albeit at a significantly higher cost in resource consumption (Marin et al. 2003). If no (recent) backup is available, then information may possibly be restored by replaying past behaviour (and consulting relevant agents and agent platforms). This may be unfeasible as vast amounts of information may need to be shared and highly dynamic environments may prevent retrieval of past behaviours; it also is a possible means for denial-of-service attacks.

An agent also needs protection from an agent platform. Unfortunately, an agent platform has enormous power over an agent which cannot easily be restrained. Some solutions advocate special purpose hardware, others have developed cryptographic techniques (e.g. Sander and Tschudin 1998). Protocols requiring agent platforms to provide a rationale for their modifications to agents may also be needed. In addition, standards for protocols and auditing logs concerning agent integrity need to be developed. Clearly in all these areas more research is required, in particular with respect to heterogeneous large-scale open systems.

3.2.2. Availability and integrity of agent platforms

To preserve the availability and integrity of an agent platform, an agent platform needs to provide mechanisms for access control (discussed in Section 2.2.1), prevention, monitoring and corrective action. Usually these mechanisms are provided by a system administrator. Earlier in Section 2.3.2. a number of risks for agent platforms have been identified. Existing agent platforms have started to address these and other security issues (including FIPA-OS, e.g. Poslad and Calisti 2000, and AOS (Wijngaards et al. 2002)). Below a number of such possible measures are discussed, including examples of corrective measures.

3.2.2.1. Preventive measures. To limit possible damages, preventive measures may be taken involving access policies and restricting the possible actions of agents. Access policies, for agents, can be based on reputation of owner, user and/or producer of an agent, combined with experiences from other, trusted, agent platforms. An agent platform may assign different levels of "trust" to agents it hosts. One preventive measure is to check an agent's code for known viruses and other malignant behaviour: a costly endeavour if possible at all. A safer approach is to have an agent only use code which is known to be valid, by accepting or downloading the code from known, trusted sites or parties. Another measure is to place the agent in a restricted execution environment, known as a "sandbox", e.g. the JVM, with which unwanted code commands can be intercepted. A third measure is, when deciding to host an agent, to impose constraints, including permissions (read/write to specific information, execution of other programmes) and resource consumption constraints (e.g., Jansen 2001). Information in both agents and agent platforms, including code and data, can be protected by encryption techniques, of which a number exist (see, e.g., Brazier et al. (2003a) in this issue for a brief overview). A complex aspect is whether mobile agents can safely carry their own decryption keys, while traversing potentially malicious agent platforms.

3.2.2.2. Monitoring. To provide a reliable environment for each of its agents, an agent platform needs to monitor itself to detect unwanted behaviour. Often, unexpected behaviour, however, is caused by correctly functioning agents: e.g. when all buyer agents in the example marketplace access the same catalogue congestion and overloading may occur. Monitoring resource consumption on agent platforms for management purposes (Mobach et al. 2003) to limit the resources allocated to each agent is a possible preventive measure. It is technically possible to digitally and securely record all actions of an agent platform together with its agreements with agents, for the purpose of furnishing proof in litigation cases. Unfortunately, this leads to extremely large volumes of recorded data, with associated costs, problems of storage and retrieval as well as legal questions as regards the lawfulness of keeping the data. Another source of information is from outside entities such as system administrators, agents, trusted third parties, and other agent platforms. This information needs to be assigned a level of trust, before subsequent action is taken.

3.2.2.3. Corrective measures. An agent platform requires guidelines on which automated corrective actions it can take in which situation, as too many potential situations may emerge to involve a human system administrator. A system administrator's agent may be treated differently from someone else's untrusted agent. An agent platform may be required to maintain a trace of its corrective actions possibly with a rationale, and may be required to employ a standard protocol for notifying agents or their users on corrective actions. Possible corrective actions include:

- slowing down an agent by allocating fewer processor-cycles,
- adjusting an agent's permissions and resource allocations, e.g. to limit message sending to once per large-time-unit,
- placing an agent into a sandbox (a restricted environment in which an agent's code is "run"),
- forcing an agent to migrate to another agent platform (possibly with the agent's latest data, otherwise causing the agent to revert to its last saved "state" based on the latest data),
- returning the agent, by stopping the agent and sending the agent (together with last saved data) to human user(s),
- killing an agent (including deleting its identity from directory and location services),
- updating and distributing reputation and trust information with regard to this agent,
- consulting a system administrator,
- informing an agent's owner on (appropriateness of) corrective actions taken.

The use of these, and other, corrective actions is related to a number of legal issues, of which we address the key problems in the next section.

3.3. LEGAL ISSUES

How do the aforementioned statutory provisions with respect to availability and integrity relate to the following question: may a system administrator remove or modify agent processes without the consent of their owners? In analyzing the issue of removal or modification of agent processes, two situations must be distinguished. On the one hand, as stated earlier, it is possible that a system administrator and an owner of a software agent have a contractual agreement detailing the specifics and conditions of the use of the platform by an agent. In this case, their mutual relation is governed by their individual contract as well as the general terms and conditions. We do not pursue this topic any further in this article, including the interesting question whether software agents can considered to act as legal persona (e.g., Wettig and Zehendner 2003) and thus be able to close contracts (e.g., Weitzenböck 2001; Brazier et al. 2003c; Cevenini 2003; Grijpink and Prins 2003; Schafer 2003).

With respect to the issue of removal or modification of agent processes, the relevant question here is whether the contract contains a provision on a possible removal or modification. When such a provision is included, the parties have to act according to what has been agreed upon.

It is, however, possible that no contractual agreement exists. In this case, the applicable legal regime must provide the answer. As discussed above, statutory law protects the availability and integrity of data. Data are however not processes, but it seems reasonable to assume that processes cannot be terminated without removing or altering data and that comparable considerations apply with respect to processes. This stance is also grounded on the insight that a "process" on a computer constitutes information in memory chips and CPU's (central processor unit), which interpret and modify this information (involving both code, data and execution state of a process).

From the provision of art. 4 CoC, it becomes clear that breaching the integrity of data is only a criminal offense if it is done "without right". It is thus relevant to determine whether a system administrator acts without right (or not) if he removes an agent's process from his platform or modifies it. On the one hand, the system administrator is the owner of his platform, or he acts on behalf of the owner. An owner may, in general, remove or alter data and processes that reside or run on his platform. On the other hand, the system administrator has allowed the software agent to commence running on his platform at some point in the past. Therewith he may or may not have raised certain expectations. Furthermore, there is the question whether he may exercise his rights as an owner without regard for the interests of other people, viz. the "owner" of an agent's process running on his platform.

3.4. TWO RELEVANT CASES

To illustrate the above in more detail and to identify relevant circumstances two court cases appear relevant. Both cases, one from Dutch law and one from American law, are analysed below to illustrate the types of circumstances that play a role in dealing with integrity problems of software agents. Each case is analysed from a legal and a technical perspective. The first case, XS4all versus ABFAB is used to introduce issues concerning the rights and obligations of system administrators. The second case, Bidder's Edge versus eBay, concerns the rights of (non-mobile) software agents. The analyses of both cases offer indications for several concluding remarks.

3.4.1. Case 1: XS4all / ABFAB

The direct marketing company ABFAB (HR 12 maart 2004, LJN-nummer: AN8483, Zaaknr: C02/264HR) sent commercial e-mails (spam) to the subscribers of XS4all, a famous Dutch ISP. In order to protect its subscribers from spam sent by ABFAB and in order to protect its system integrity, XS4all wanted to forbid ABFAB to send spam to its system. XS4all founded its claim inter alia on its property right in its system. Basically XS4all's reasoning is sound. As the proprietor of an object one can freely enjoy his or her property and fend off third parties that interfere with one's unencumbered enjoyment of the property. There may, however, be circumstances in which an appeal to one's property right alone is not enough to fend off others that interfere. This may, e.g., be the case with very unsubstantial interferences with a property right or if the object is used for purposes that reflect a certain public interest (such as an e-mail service) and the interference involves this public interest.

The question then becomes: under what circumstances can a proprietor fend off interferers? In Dutch legal doctrine two schools of thought exist. According to the first school a proprietor can fend off third parties if he or she can show that he or she has a good reason to do so, justifying his actions. According to the second theory he or she can only act if the third party acts unlawfully towards him or her. The Dutch Supreme Court has chosen the former theory. It is clear that this school leaves more room for "enforcement" by the proprietor. After all, it is for XS4all much easier to indicate that it has good reasons to fend off spam from ABFAB (protection against spam, protection of system integrity) than is to indicate that ABFAB acts unlawfully when it sends commercial e-mail (it is after all questionable whether sending commercial mail is unlawful *per se*).

The Advocaat-Generaal (i.e. the advisor of the Hoge Raad) basically concurred. According to the Advocaat-Generaal, an owner can fend off the use of his or her system if he or she has sufficiently weighty reasons to do so and his or her decision is not based on an unfair weighing of the interests of the parties involved.

3.4.2. Case 2: Bidder's Edge/eBay

eBay is an Internet-based, person-to-person trading site (eBay, Inc. versus Bidder's Edge, Inc. 100 F. Supp. 2d 1058, N.D. Cal. 2000). eBay offers sellers the ability to list items for sale and prospective buyers the ability to search those listings and bid on items. Bidder's Edge (BE) is an auction aggregation site designed to offer on-line auction buyers the ability to search for items across numerous on-line auctions without having to search each auction individually. The information that is made available on the BE site is stored in a database of information that BE compiles through access to auction sites such as eBay.

eBay demanded that BE ceased accessing the eBay site, because BE's queries for filling its database compromised the capacity of eBay's system. The Court found that BE's access to the site constituted a trespass to chattels. A trespass to chattels is defined by the Court as an intentional interference with the possession of personal property has proximately caused injury. See Thrifty-Tel versus Beznik, 46 Cal. App. 4th 1559, 1566 (1996). If this trespass consists of accessing a computer system, the plaintiff must establish that: (1)

defendant intentionally and without authorization interfered with plaintiff's interest as a possessor in the computer system; and (2) that defendant's unauthorized use proximately resulted in damage to plaintiff. See Thrifty-Tel, 46 Cal. App. 4th at 1566; see also Itano versus Colonial Yacht Anchorage, 267 Cal. App. 2d 84, 90 (1968).

With respect to the first requirement, the Court concluded that BE's activity is sufficiently outside the range of the use permitted by eBay that it is unauthorized for the purposes of establishing a trespass. Although the website is publicly accessible, eBay does not grant the public unconditional access and does not generally permit the type of automated access made by BE. In fact, eBay explicitly notifies automated visitors that their access is not permitted. BE continued however to "crawl" eBay's web site even after eBay demanded BE to terminate this activity. The Court decided that even if BE's web crawlers were authorized to make individual queries of eBay's system, BE's web crawlers exceeded the scope of any such consent when they begin acting like robots by making repeated queries. With respect to the second requirement the Court held that eBay suffered damage by the mere fact that BE's use is appropriating eBay's personal property by using valuable bandwidth and capacity, and necessarily compromising eBay's ability to use that capacity for its own purposes.

3.4.3. Conclusion

What do both cases reveal when considering issues related to integrity and availability of agents and agent platforms? The XS4all/ABFAB case indicates that a system administrator intending to modify an agent must first refer to his (employer's) ownership of the system. In addition he or she must balance his or her own interests against those of the agent–owner and such balancing must not be unfair.

The BE/eBay case identifies issues in the case of a system administrator removing unwanted agent processes. In the first place, the conditions under which the system administrator has allowed a software agent to use his or her system are of importance. In this respect, it is relevant that the software agent or its user can recognize conditions and the nature of these conditions imposed by a system administrator. In the second place, it is relevant to what extent the software agent uses the capacity of the system administrator's system. In the above case, it wasn't necessary that alternative uses were frustrated by the fact that BE used the system. It was sufficient that capacity was consumed that could have been used by eBay or its customers.

Finally, the eBay/BE case shows that problems with software agents are already a reality. In general, the owner of an agent platform has some leeway in dealing with software agents. Nonetheless the agent platform owner will always have to adduce a good reason for acting against software agents and the balancing of his or her own interests against those of the owner of the software agent must not be unfair. In this respect, it is helpful if the agent platform owner has communicated beforehand what the conditions are under which a software agent may make use of the platform.

4. Adressing future problems

The current state of the art of both technology and law, and especially its interrelation, does not provide for easy-to-use cookbooks detailing guidelines and standards for the development of law-abiding agents. The worldwide deployment of (open and closed) agent systems, however, hinges on social and commercial acceptance, which is fostered by meeting explicit legal standards. Guidelines are needed to describe standard techniques to employ in agent systems, and standard measures to take in specific situations. Development of guidelines is a time-consuming endeavour, as understanding of the interplay between technological aspects of agents and agent platforms is still developing. Important technical aspects of agents and agent platforms, interoperability, scalability and security, are being researched, the results of which may influence legal possibilities. On the basis of experiences with closed systems and theory on (semi-)open systems, four recommendations are listed below.

4.1. PRE-EMPTIVELY CONSIDER LEGAL RISKS

The analysis of the case of the system administrator wishing to remove an agent, in Sections 3.3 and 3.4.2, from his or her system clearly demonstrates the added value that can be gained by a timely appreciation of legal aspects imposing conditions of use upon software agents and their users, for example by means of a contractual provision. If during an earlier stage protocols are made to request or to force an agent to migrate, then the system administrator's problem seems more manageable and may even not be perceived as a "problem": (standardised) and well-thought protocols minimise integrity and availability problems.

4.2. EXPLICIT AGREEMENT ON USAGE AND ACCESS CONDITIONS

The conditions under which access to a computer system is granted are relevant. Nevertheless, a number of outstanding questions with respect to these conditions remain. The conditions for use of an agent platform must, e.g., be clearly communicated to the software agent (and possibly its user) by the system administrator. The current practice with which website maintainers can constrain automated access by providing a file with conditions, can easily be ignored by external parties: an explicit agreement is not reached. A straightforward way to enforce agreement on conditions may be a mechanism that is able to "physically" prohibit any use of an agent platform when the usage violates conditions. Tools and techniques need to be developed for an agent platform to monitor usage and check for violation (and agreement) of conditions.

Technical measures may have an all or nothing character: allowing either too much or too little. In the eBay versus BE case, the former blocked at one point the latter's IP-addresses in an attempt to prohibit the acts that violated the conditions. This measure did not, however, differentiate between acts in compliance or violation of the conditions. This case also highlights a second drawback of physical measures: BE easily robbed the measure of its effectiveness by using a proxy-server, thus hiding her IP-address and nullifying the effect of eBay's IP-blocking. In addition, technical measures may, on their own, make the precise conditions for use insufficiently explicit: was e.g. BE's access through the proxy-server allowed since it was not blocked or must IP-blocking be understood to be a general denial of access?

It is indispensable to have a protocol that facilitates communication of the conditions of use much more clearly and explicitly than is the case with "coercive technical measures". A distinction needs to be made with respect to the recipient of the conditions of use. If they are to be communicated to the software agent, a protocol is required that can express that conditions are being communicated and specify these conditions, in such a way that this is meaningful for the software agent, e.g., by using an annotation language such as XrML (Gelati et al. 2003). In addition, it is desirable for communication of conditions to guarantee non-repudiation of receipt, i.e. it must be provable that the software agent has undeniably received and understood the conditions. An appropriate protocol could, e.g., require a software agent to acknowledge receipt of the conditions. If the conditions are to be communicated to the owner of the software agent, the owner must be identifiable and have a contact address. Again, a protocol is necessary, including non-repudiation of receipt.

4.3. EXPLICIT AGREEMENT ON RIGHTS, OBLIGATIONS AND SANCTIONS

Not only usage and access conditions needs to be specified, but also rights, obligations and sanctions need to be explicitly agreed upon by the involved parties, which can include terms of notification, number of warnings issued and possible normative deliberations. Both an agent platform and a software agent have obligations to each other, e.g. to respect their respective (private) data, accommodate fair usage, etc. Such an explicit agreement alleviates potential (legal) problems when, e.g., a system administrator employs certain measures affecting the performance and functioning of software agents. Especially when agent platforms automatically act according to policies and take these agreements into account, without involving a system administrator in every decision. A predictable agent platform may be better trusted and valued by software agents and their human users.

In the XS4all versus ABFAB case, a system administrator has a number of options. For a system administrator to regulate message volume, management policies may be a used, e.g., to dynamically constrain resource usage, i.e. the number of messages to be send per time-unit. A rationale may have been to grant every user the opportunity to send and receive email and at the same time-constrain large-scale senders. Additionally, security (and access control) policies may have been put into place whereby anti-virus and anti-spam measures temporarily block sources of vast amounts of emails (or other messages). To stop distributed denial of service attacks, in which multiple computers (e.g., more than 1,00,000) are involved in sending messages to a target computer, a system administrator may enlist the help of other system administrators, ISPs and network administrators, who together may stench the message flood.

In the eBay versus Bidder's Edge case the agent platform could consider isolating the suspicious Bidder's Edge agents in such a way that is not harmful to eBay's system, possibly while storing enough data in order to allow the agents to resume on another system (this was not an option for eBay). A more drastic option which eBay's system administrator could consider is killing Bidder's Edge agents. In all situations eBay's system administrator could consider notifying the owner of the agents (Bidder's Edge) of their illegal presence and their potential termination.

Unique identities of agents facilitate notification of stakeholders. Software robots were used to search, amongst others, eBay's database to update the Bidder's Edge database. eBay's system administrator could identify software agents such as the Bidder's Edge robots, e.g. by analysis of the IP-addresses from which these agents originated or their co-ordinated behaviour. In the case of eBay, preventive measures were not of relevance, as far as can be deduced from the case. Possible preventive measures include constraining resource usage when an agent is accepted. In case of eBay for example, it was technically possible for Bidder's Edge agents to extensively access eBay, far more than is "normally" required by clients.

4.4. DESIGN FOR CHANGE

As technology progresses, guidelines need to be frequently updated to reflect current state-of-the-art technologies and measures. This is increasingly important in a world in which security patches are frequently issued (e.g., cf. Microsoft Windows Update) and new hardware is developed (e.g., optical computing, quantum computing). Mechanisms may need to be developed to automatically update existing agent systems to current legal, technical and security requirements.

4.5. DISCUSSION

In sum, a system administrator, as the owner of a computer system, decides whether or not to allow software agents to use his or her agent platform. A system administrator may or may not bind the use by the software agents to certain conditions. At a later point in time a system administrator may want to disallow the use of his agent platform. What is he or she to do with running agent processes? Can these processes only be prematurely removed from the agent platform with the consent of the agent's user or can the system administrator remove agent processes from his or her agent platform without consent of the agent's user or even against their will? A preliminary analysis of case law has been performed in order to find the arguments pro and contra interference by the system administrator. This analysis made clear that the following circumstances are relevant:

- Has the owner of an agent platform stated conditions to the use of this system? Has a specific software agent or its "owner" violated these conditions? Does the absence of conditions raise expectations about the usability of the agent platform for use by software agents? With respect to the conditions other issues may be relevant, such as: can an agent or its owner recognise that conditions are being imposed? Can the nature of these conditions be recognised by the software agent or its user?
- What interest has the owner of an agent platform to disallow further use of his or her system? Does he or she suffer damage? Is the agent platform's integrity affected? Are – real or potential – other uses of system capacity precluded by a specific software agent?
- What is the nature and seriousness of the measures taken against a specific software agent? Requirements of proportionality and subsidiary role – i.e. the least harmful remedy has been chosen – decrease the system administrator's leeway.
- What is the nature of the service provided to the users of software agents? Is it a public service? Is the service important for society,

such as an e-mail service? To what extent must the interests of the "owner" of the agent process in its unencumbered functioning be protected?

- What are consequences of the use of "agreements" between agent platforms and software agents, when these agreements resemble contracts? Existing research on management of distributed systems, including agents, includes research on leases: a simple form of such an agreement.

For practical purposes it is of course wise to log the pertinent situations and actions in order to (be able to) prove the facts that underpin one's position in court.

5. Discussion

Agent technology is progressing at a fairly steady pace neglecting reflections on legal aspects and implications on the use of software agents and agent platforms, potentially affecting their future usability. The extent of possible damages is difficult to predict; yet it is "better to be safe then to be sorry". Numerous complex legal issues need to be resolved before agent technology can be safely, and legally, applied (e.g., Brazier et al. 2002a). A multi-disciplinary approach is advocated, as each of the research areas of Law, Computer Systems and Artificial Intelligence cannot research these legal issues on their own. Agent developers need to know how to design software agents that comply with known legal requirements (corresponding with the intended use of the agents). In addition, developers of agent platforms (which support running or active software agents) need to know which legal issues play a role.

For the time being, system administrators may still be confronted with the presence of "suspicious" agent's processes on their systems. The technical analysis has shown that there is a whole range of measures a system administrator can take against software agents violating conditions for use. These measures range from reducing resources for the software agent to removing the software agent from the system. The traditional and well-known legal criteria of proportionality and subsidiary role imply that the least damaging measures that are effective must be chosen. The question is, however, which measures meet this description. As there are, as yet, no legal standards available norms, protocols and other standards need to be developed by market parties. This, however, requires further insight into the specific implications and problems of software agent deployment. Thus, experiences have to be collected and discussed. The outcome of such collection and discussion can be laid down in self-regulatory instruments that can guide system administrators in their approach to agent processes that go beyond the latter's allowed use.

The legal and technical communities need to embark on interdisciplinary research to deal with the issues addressed in this article. One option is to first analyse a complex closed-system application to gain additional insight before analysing (semi-)open systems such as marketplaces. A real-world application being considered is the case of using agents to support the use of digital dossiers in court: addressing specific questions such as demarcation of dossier access (e.g., create, read, write, modify, delete of the whole of parts) by judges, lawyers, and other parties (e.g., Oskamp and Brazier 2001). The relevance of a number of aspects of software agents and agent platforms is being examined, including transaction capabilities, identity, liability, traceability, privacy, copyright, autonomy, and legal identity. This study involves multiple perspectives, such as the user's perspective, a software agent's perspective, an agent platform's perspective, an Artificial Intelligence perspective (e.g. regarding knowledge and capabilities), a Computer Systems perspective (concerning security, reliability, deployment, etc.) and a Legal perspective (including integrity, availability, etc.). Any interdisciplinary research encounters cultural and language problems including ambiguity of interpretations of concepts: our aim is to engage in discussions to develop a shared framework, with explicit mappings to frameworks used in individual disciplines (Apistola et al. 2002). Our joint research in this challenging and rewarding domain continues.

Acknowledgements

The authors thank the anonymous reviewers for their comments, which led to a number of improvements of the article. The ALIAS project is supported by NLnet Foundation, http://www.nlnet.nl. The ALIAS project is an interdisciplinary project specifically aimed at exploring the legal status of agents and the implications of their use. The authors acknowledge the contributions made by the participants of the ALIAS project: Martin Apistola, Onno Kubbe, Erik Schreuders and Marten Voulon; http://www.iids.org/alias/.

References

AgentCities (2003). http://www.agentcities.org/

Anderson, R. (2001). Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley Computer Publishing: New York.

Apistola, M., Brazier, F. M. T., Kubbe, O., Oskamp, A., Schellekens, M. H. M., and Voulon, M. B. (2002). Legal aspects of agent technology. In Proceedings of 17th Bileta conference, Amsterdam, March 2002, 11 pp. Bileta: Edinburgh, Scotland. http://www.bileta.ac.uk/.

Barber, K. S., McKay, R., Goel, A., Han, D., Kim, J., Liu, T. H., and Martin, C. E. (2000). Sensible Agents: The Distributed Architecture and Testbed. IEICE Transactions on Communications 5: 951–960. IECIA/IEEE Joint Special Issue on Autonomous Decentralized Systems, E83-B.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing Multi Agent Systems with a FIPA-Compliant Agent Framework. Software - Practice And Experience 31(2): 103–128.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. Scientific American 184(5): 34–43.

Biasiotti, M. A., Romano, F., and Sagri, M. -T. (2003). The Software Agent's Liability for Harm Suffered by Third Parties, In Oskamp A. and Weitzenböck E. (eds.), In Proceedings of the Law and Electronic Agents workshop (LEA '03), 47–56. Norwegian Research Center for Computers and Law: Oslo, Norway.

Bing, J. (2003). Copymarks: A Suggestion for Simple Management of Copyrighted Material, In Bing J. and Sartor G. (eds.), The Law of Electronic Agents 2003. Legal Contributions to ALFEBIITE - A Logical Framework for Ethical Behaviour Between Infohabitanats in the Information Trading Economy of the Universal Information Ecosystem 185–222. Norwegian Research Center for Computers and Law: Osly, Norway.

Bradshaw, J. M. (ed.), (1997). Software Agents. AAAI Press / MIT Press: Menlo Park, CA.

Brazier, F. M. T., Kubbe, O., Oskamp, A., and Wijngaards, N. J. E. (2002a). Are Law-Abiding Agents Realistic? In Sartor, G. and Cevenini, C. (eds.), In Proceedings of the workshop on the Law of Electronic Agents (LEA '02), 151–155. CIRSFID, University of Bologna. Bologna, Italy.

Brazier, F. M. T., Oskamp, A., Prins, J. E. J., Schellekens, M. H. M., and Wijngaards, N. J. E. (2003a). Are Anonymous Agents Realistic? In Oskamp, A. and Weitzenboeck, E. (eds.), Proceedings of the LEA 2003: The Law and Electronic Agents, 69–79. Norwegian Research Center for Computers and Law. See also their contribution in this special issue of AI & Law: Oslo, Norway.

Brazier, F. M. T., Oskamp, A., Schellekens, M. H. M. and Wijngaards, N. J. E. (2003b). Are Mobile Agents Outlawed Processes? In Oskamp, A. and Weitzenboeck, E. (eds.), In Proceedings of the LEA 2003: The Law and Electronic Agents, 127–139. Norwegian Research Center for Computers and Law: Oslo, Norway.

Brazier, F. M. T., Oskamp, A., Schellekens, M. H. M. and Wijngaards, N. J. E. (2003c). Can agents close contracts? In Oskamp, A. and Weitzenboeck, E. (eds.), In Proceedings of the LEA 2003: The Law and Electronic Agents, 9–20. Norwegian Research Center for Computers and Law: Oslo, Norway.

Brazier, F. M. T., Overeinder, B. J., van Steen, M., and Wijngaards, N. J. E. (2002b). Agent Factory: Generative Migration of Mobile Agents in Heterogeneous Environments. In Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002), 101–106. ACM Press: Madrid, Spain.

Brazier, F. M. T., van Steen, M., and Wijngaards, N. J. E. (2001). On MAS scalability. In Wagner T. and Rana, O. (eds.), Proceedings of Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, 121–126, Montreal, Canada.

Castelfranchi, C. (2001). Again on Agents? Autonomy: A Homage to Alan Turing. In Castelfranchi, C. and Lespérance Y. (eds.), Intelligent Agents VII, Lecture Notes in Artificial Intelligence 1986, 339–342. Springer-Verlag: Berlin, Heidelberg.

Cevenini, C. (2003). Contracts for the Use of Software Agents in Virtual Enterprises. In Oskamp, A. and Weitzenböck E. (eds.), Proceedings of the Law and Electronic Agents workshop (LEA '03), 21–32. Norwegian Research Center for Computers and Law: Oslo, Norway.

Dale, J. and Mamdani, E. (2001), Open Standards for Interoperating Agent-based Systems. Software Focus 2(1): 1–8, (Spring).

De Wilde, P., Nwana, H. S., and Lee, L. C. (1999). Stability, Fairness and Scalability of Multiagent Systems. International Journal of Knowledge-Based Intelligent Engineering Systems, 3(2): 84–91.

Ding, Y., Fensel, D., Klein, M., and Omelayenko, B. (2002). The Semantic Web: Yet Another Hip? Data and Knowledge Engineering 41(2–3): 205–227.

Finin, T., Labrou, Y., and Mayfield, J. (1997). KQML as an Agent Communication Language. In Bradshaw, J. (ed.), Software Agents 291–316. MIT Press: Cambridge.

FIPA (2001), FIPA Agent Platform, http://www.fipa.org.

Fuggetta, A., Picco, G. P. and Vigna, G. (1998). Understanding Code Mobility. IEEE Transactions on Software Engineering. 24(5): 342–361.

Gelati, J., Rotolo, A., and Sartor, G. (2003). A Logic-based Analysis of XrML. In Oskamp A. and Weitzenböck, E. (eds.), Proceedings of the Law and Electronic Agents workshop (LEA'03), 57 – 68. Norwegian Research Center for Computers and Law: Oslo, Norway. See also their contribution in this special issue of AI and Law.

Geurts R. (2002). Legal Aspects of Software Agents. In Prins, Ribbers, Van Tilborg, Veth, Van der Wees (eds.), Trust in Electronic Commerce. The Role of Trust from a Legal, an Organizational and a Technical Point of View, 231–270. Kluwer Law International: The Hague, Boston.

Gray, R. S., Cybenko, G., Kotz, D., Peterson, R. A. and Rus, D. (2001). D'Agents: Applications and Performance of a Mobile-agent System. Software: Practice and Experience 32(6): 543–573.

Grijpink J. and Prins J. E. J. (2003). New Rules for Anonymous Electronic Transactions? An Exploration of the Private Law Implications of Digital Anonymity. In Nicoll, C. Prins, J. E. J. and van Dellen, M. J. M. (eds.), Digital Anonymity and the Law. Tensions and Dimensions 249–270. T.M.C. Asser Press, The Hague.

Jansen, W. (2001). A Privilege Management Scheme for Mobile Agent Systems. First International Workshop on Security of Mobile Multiagent Systems, Autonomous Agents Conference. ACM Press: Montreal, Canada.

Jennings, N. R. (2000). On Agent-based Software Engineering. Artificial Intelligence 117(2): 277–296.

Jennings, N. R. and Wooldridge, W. J. (eds.), (1998). Agent Technology: Foundations, Application, and Markets. Springer-Verlag, Berlin: Germany.

Karnik, N. and Tripathi, A. (2001). Security in the Ajanta Mobile Agent System. Software: Practice and Experience 31(4): 301–329.

Karnow, C. E. A. (1996). Liability for Distributed Artificial Intelligence. Berkeley Technology Law Journal 11: 161–162.

Lange, D. B., Oshima, M., Karjoth, G., and Kosaka, K. (1997). Aglets: Programming mobile agents in Java. In Worldwide Computing and Its Applications, Lecture Notes in Computer Science 1274, 253–266. Springer-Verlag: Berlin, Germany.

Luck, M., McBurney, P., and Preist, C. (2003). Agent Technology: Enabling Next Generation Computing. AgentLink, ISBN 0854 327886, available from http://www.agentlink.org/roadmap/.

Marin, O., Bertier, M., and Sens, P. (2003). DARX - A Framework for the Fault-Tolerant Support of Agent Software. In: Proceedings of the 14th IEEE International Symposium on Software Reliability Engineering (ISSRE 2003), 406–417. IEEE Computer Society: Denver, Colorado, USA.

Martin, D., Cheyer, A., and Moran, D. (1999). The Open Agent Architecture: A Framework for Building Distributed Software Systems. Applied Artificial Intelligence 13(1/2): 91–128.

McKnight, D. H. and Chervany, N. L. (2001). Trust and Distrust Definitions: One Bite at a Time. In Falcone, R., Singh, M. P., and Tan, Y.-H. (eds.), Trust in Cyber-Societies: Integrating the Human and Artificial Perspectives, Lecture Notes in Computer Science 2246, 27–54. Springer: Berlin, Heidelberg.

Milojicic, D. Breugst, M., Busse, I., Campbell, J., Covaci, S., Friedman, B., Kosaka, K., Lange, D. B., Ono, K., Oshima, M., Tham, C., Virdhagriswaran, S., and White, J. (1998). MASIF: The OMG Mobile Agent System Interoperability Facility. In Proceedings of the 2nd International

Workshop on Mobile Agents, 1477 of Lecture Notes in Computer Science 50–67. Springer-Verlag: Berlin, Germany.

Mobach, D. G. A., Overeinder, B. J., Wijngaards, N. J. E. and Brazier, F. M. T. (2003). Managing Agent Life Cycles in Open Distributed Systems, In Proceedings of the 18th ACM Symposium on Applied Computing, 61–65. ACM Press: Melbourne, Florida, USA.

Noordende, G. van ‘t, Brazier, F. M. T. and Tanenbaum, A. S. (2002). A Security Framework for a Mobile Agent System, In Fischer, K. and Hutter D. (eds.), Proceedings of the 2nd International Workshop on Security in Mobile Multi-agent Systems (SEMAS 2002), associated with AAMAS-2002, Bologna, Italy, DFKI Research Report RR-02-03, Deutsches Forschungszentrum fur Künstliche Intelligenz, 43–50.

Nwana, H. S. (1996). Software Agents: An overview. The Knowledge Engineering Review 11(3): 205–244.

Nwana, H. S., Ndumu, D. T., Lee, L. C. (1998). ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems, Applied AI 13(1–2): 129–185.

Oskamp, A. and Brazier, F. M. T. (2001). Intelligent Agents for Lawyers, In Proceedings of the Workshop Legal Knowledge Systems in Action: Practical AI in Today’s Law Offices, 5 pp. Saint Louis, Missouri.

Peine, H. and Stolpmann, T. (1997). The Architecture of the Ara Platform for Mobile Agents. In Proceedings of the First International Workshop on Mobile Agents (MA ’97), Lecture Notes in Computer Science 1219, 50–61, Springer-Verlag: Berlin, Germany.

Poslad, S. and Calisti, M. (2000). Towards Improved Trust and Security in FIPA Agent Platforms, In Proceedings of the Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies, 87–90. ACM Press: Barcelona, Spain.

Sander, T. and Tschudin, C. F. (1998). Protecting Mobile Agents Against Malicious Hosts, In G. Vigna (ed.), Mobile Agents and Security, Lecture Notes in Computer Science 1419, 44–60. Springer-Verlag.

Schafer, B. (2003). It’s Just not Cricket - RoboCup and Fair Dealing in Contract, In Oskamp, A. and Weitzenböck E. (eds.), Proceedings of the Law and Electronic Agents workshop (LEA ’03), 33–46. Norwegian Research Center for Computers and Law: Oslo, Norway.

Shoham, Y. (1993). Agent-oriented programming. Artificial Intelligence 60(1): 51–92.

Stuurman, K. and Wijnands, H. (2001). Intelligent Agents: A Curse or a Blessing? A Survey of the Legal Aspects of the Application of Intelligent Software Agents. Computer Law and Security Report 17(2): 92–100.

Suri, N., Bradshaw, J. M., Breedy, M. R., Groth, P. T., Hill, G. A., Jeffers, R., Mitrovich, T. S., Pouliot, B. R., and Smith, D. S. (2000). Nomads: Toward a Strong and Safe Mobile Agent System. In Proceedings of the Fourth International Conference on Autonomous Agents, 163–164. IEEE Computer Society: Boston, Massachusetts, USA.

Sycara, K., Paolucci, M., van Velsen, M., and Giampapa, J. (2003). The RETSINA MAS infrastructure. Autonomous Agents and Multi-Agent Systems 7(1–2): 29–48.

Tanenbaum, A. S. and Van Steen, M. (2002). Distributed Systems: Principles and Paradigms, Prentice Hall: New Jersey.

Tripathi, A., Karnik, N., Vora, M., Ahmed, T., and Singh, R. (1999). Mobile agent programming in Ajanta. In Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS ’99), 190–197. IEEE Computer Society: Austin, Texas, USA.

Turner, P. J. and Jennings, N. R. (2000). Improving the scalability of multi-agent systems. In Wagner, T. and Rana, O. (eds.), Proceedings of the First International Workshop on Infrastructure for Scalable Multi-Agent Systems, 246–262. Barcelone, Spain.

van Klink, B. M. J. and Prins, J. E. J. (2002). Law and Regulation: Scenarios for the Information Age. IOS Press: Amsterdam, Berlin, Oxford.

Weitzenboeck, E. M. (2001). Electronic Agents and the Formation of Contracts, International Journal of Law and Information Technology 9(3): 204–234.

Wettig, S. and Zehendner, E. (2003). The Electronic Agent: A Legal Personality under German Law? In Oskamp, A. and Weitzenböck, E. (eds.), Proceedings of the Law and Electronic Agents workshop (LEA '03), 97–112. Norwegian Research Center for Computers and Law: Oslo, Norway. See also their contribution in this special issue of AI and Law.

Wijngaards, N. J. E., Overeinder, B. J., van Steen, M., and Brazier, F. M. T. (2002). Supporting Internet-Scale Multi-Agent Systems. Data and Knowledge Engineering 41(2–3): 229–245.

Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., and Yu, L. (2002). Negotiating Trust on the Web. IEEE Internet Computing 6(6): 30–37.

Wong, H. C. and Sycara, K. (1999). Adding Security and Trust to Multi-Agent Systems, In Proceedings of Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies, 149–161. IEEE Computer Society: Boston, Massachusetts, USA.

Wooldridge, M. J. and Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. The Knowledge Engineering Review 10(2): 115–152.

Yip, A. and Cunningham, J. (2003). Ontological Issues in Agent Ownership (Reasoning about Agent Ownership), In Oskamp, A. and Weitzenböck, E. (eds.), Proceedings of the Law and Electronic Agents workshop (LEA '03), 113 – 126. Norwegian Research Center for Computers and Law: Oslo, Norway. See also their contribution in this special issue of AI and Law.