Autonomous Agents and Multi-Agent Systems, 2, 87–103 (1999)

Liability for Autonomous Agent Design

CAREY HECKMAN ceh@stanford.edu
Stanford Law School, Crown Quadrangle, Stanford, CA 94305-8610, (650) 725-7788

JACOB O. WOBBROCK jakeow@leland.stanford.edu
Stanford Symbolic Systems Program, PO Box 5264, Stanford, CA 94309

Abstract. Though exciting scientifically, autonomous agent design can result in legal liability. This paper surveys those legal concerns, focusing on issues arising from the unique qualities of agents not found in conventional software. Informed designers can more effectively reduce their liability exposure and influence emerging agent liability law and policies.

Keywords: legal liability, design, adaptivity, anthropomorphism, autonomy, adaptivity, strict liability, negligence, misrepresentation

1. Introduction

Autonomous software agents differ qualitatively from conventional software in many ways. Designers must be aware of the legal issues that agents raise, as this awareness will aid in thoughtful design and encourage the ability to anticipate the sources of liability. Informed designers can reduce their liability exposure and more effectively influence emerging agent liability laws and policies.

Consider the following scenarios:

Scenario 1. As a consultant for network servers, you employ a number of mobile agents to help you with your work. Once you have been contracted to work on a system, the agents travel across the network, locate the server, and gather information about the system. Eventually they return with a profile of the system information, assembling a report that will enable you to prepare for your work. However, on one occasion an agent causes damage to a remote system. The client sues you for damages to its system, but you contend that it was out of your hands and the fault of the agent developers.

Scenario 2. A high-profile technology company releases a financial application suite. Private citizens and professional financiers alike adopt the software. As a sort of "master of ceremonies" for the suite, Stockbroker Stanley is released as an agent that interacts with the user across all of the applications. Stanley serves as a user-interface agent; users channel their input through Stanley. His graphical representation is elaborate: he bears an honest facade, appears appropriately in a fine suit, exhibits a wide variety of gestures, voice outputs, and appropriately timed

pseudo-conversations with the user, and is convincing as an intelligent financier. Stanley is used to observe the stock market and create trend analyses, offering predictions and advice on which stocks to buy and sell. He can even be asked to buy the stocks himself via an on-line service.

Over time, however, it becomes apparent that Stanley's financial advice is abysmal, and thousands of people who took it too seriously end up losing a great deal of money. Stanley's "clients" bring a class action lawsuit.

Scenario 3.[1] A simple desktop agent performs UNIX background operations for you, removing old files, backing up important documents, and so forth. The agent has an interface that allows you to pass it shell command strings such as `rm*.sav`. On one such occasion, a hacker intercepts one of these messages and changes it to `rm*.*`. Fortunately the agent protects against executing this command and refuses the operation. The hacker realizes this and introduces a new shell program called `save` such that save `*.dat` has the same effect of `rm*.*`. This time the agent accepts the command and executes it, and your files are lost. The hacker is held accountable for the intrusion, but you also consider holding the corporation that created the agent liable, since it did not protect against such actions.

This paper discusses the legal concerns that agents raise. Some of these issues pertain equally to conventional software, but as we shall see, agents raise them with more urgency and in a new light. In this paper we will show that certain characteristics of agents make them even more susceptible to liability issues than conventional software. These issues are *unique* to agents, and designers should be wary of them.

2. General Principles of Agent Liability

Creating a software agent creates potential legal responsibility for any harm inflicted by the agent. This responsibility may require paying money to repair the damage. In severe situations, the agent's creator may be convicted as a criminal.

The creator's liability has limits. An agent's creator may not always have to reimburse others for the agent's harm. Society's rules accept that those involved in an accident sometimes should share responsibility, and sometimes a victim should bear all or at least part of the injuries he or she has suffered.

On the other hand, society's rules also recognize that certain behavior creates responsibility for the harm it causes. Under some circumstances, making the victim bear the burden of the harm would be unjust. The victim would be tempted to seek retribution on his or her own, increasing rather than avoiding civil disorder. Fairness and keeping the peace, society has concluded, require imposing responsibility on people other than the victim. Society's rules may also reflect an attempt to encourage injury prevention, or if that is impractical, at least to spread the costs more efficiently so unpreventable injuries will not be as individually disruptive or even devastating.

Rules differ among societies for deciding when someone has committed a crime that must be punished or fined. International uniformity is the exception, although

the growth of regional trading regions and multilateral trade relationships have harmonized some differences. The rules may vary among provinces or states of one nation. In the United States, for example, national law says comparatively little about the liability of a software agent designer. The individual states play a far greater role, and the differences between states can be significant.

On the whole, however, states within the United States have based their rules on similar general principles. These general principles underpin our discussion in this paper. While this cannot deliver an answer for any specific set of facts, the general principles indicate where the designer of an agent should pay particular attention.

2.1. Liability for Intentional Acts

Writing a software agent intentionally for use in harming someone or stealing their property would make the agent's creator as subject to criminal prosecution as the person who used the agent in the crime. Federal and state criminal conspiracy statutes may also apply. The Credit Card Act of 1984, for example, makes it a U.S. federal crime to produce any counterfeit or unauthorized means of accessing an account that can be used to obtain money, goods, services, or anything else of value. Writing a software agent to collect, without authorization, ATM personal identification numbers from a bank network would likely violate this statute. Note that use of the agent is not required; creating the agent is sufficient for criminal prosecution.

Creating a software agent that is substantially certain to cause others pain, suffering, or loss of privacy can bring about liability in a private civil lawsuit. The agent's creator may be held responsible for all the damages caused by the agent, even beyond those the creator could have anticipated. It is not even necessary that the agent's creator intended to cause any damage, only that damages from the agent were probable.

Suppose the hacker in scenario three created an agent that deleted files. The legal consequences would be as if the hacker had taken a hatchet to someone's storage devices.

Or suppose the consultant in scenario one designed an agent to retrieve (without consent) information from the personally-owned home computers of the company's employees when they dial into the corporate computer network. An affected employee could claim that the consultant committed an intentional and highly offensive intrusion into an area of her life that she could reasonably expect would not be intruded upon. If a court agreed, the consultant would have to pay damages resulting from the intrusion. Additionally, punitive damages might also be awarded if the circumstances justify further discouraging the consultant from behaving this way again.

Finally, suppose Stockbroker Stanley is an unauthorized audio or visual portrayal of a celebrity. Many states have a specific personality appropriation statute or otherwise allow lawsuits to recover damages caused by this kind of privacy invasion.

2.2. Liability for Negligent Acts

Careless creation of a software agent can also make the agent's creator liable for damages. The agent's creator may have had the best possible intentions but nevertheless designed the agent without fulfilling a duty to take sufficient precautions to ensure that the agent would not damage anyone or anything.

According to the Second Restatement of Torts (an influential treatise that strives to summarize the liability laws of the U.S. states), negligence is "conduct which falls below the standard established by law for the protection of others against unreasonable risk of harm." To create legal liability for negligent agent design, the injured party must prove that:

- the agent's designer failed to use reasonable care;
- the failure to use reasonable care caused harm;
- the agent's designer has legal responsibility for that harm; and
- the agent's designer has no recognized defenses for liability.

2.2.1. Failure to Use Reasonable Care. In general, a software agent designer has a duty to act as a prudent and reasonable person would under the same or similar circumstances. The designer's best intentions or lack of awareness of doing anything wrong are of no consequence. Instead, the designer's behavior must conform to what others would do in the same situation. Community custom may be one indication. But especially in a fast changing technology, a designer cannot rely on a community custom that is no longer reasonable. A degree of care sufficient when agent design was more primitive is likely to be inadequate with today's more sophisticated design principles, and today's standards will become inadequate as agent design advances in the future. Another frequently used formulation for "reasonable care" balances the probability of the injury occurring and the degree of injury that would occur against the burden of preventing the injury from occurring.

If it can be shown that agent design is a profession, the law imposes a more demanding standard of care: the care that similar professionals exercise in the same or similar communities. Whether this tougher standard applies remains undecided. The wider recognition of agent design as involving special knowledge and skills, the general understanding that an academic degree in agent design is needed to work in the field, and the growing significance of quality programming to personal and property safety, all point towards a "reasonable agent designer" standard rather than the broader "reasonable person."

The software agent designer owes this duty to use reasonable care only to those the designer could have reasonably foreseen as being endangered by the designer's failure to exercise that care. Consider a prototype agent that a designer stored on a computer in an access-restricted office. However, one night someone breaks into the designer's office and, despite the "DO NOT TOUCH" sign, executes the agent. The agent malfunctions and causes considerable damage. A court would probably determine that the designer owed no duty of care because she could not have reasonably foreseen someone breaking into her office and ignoring her sign.

2.2.2. Causation of the Harm. The failure to use reasonable care must have caused the harm. In other words, *but for* the designer's failure to use reasonable care, the injury would not have occurred. Thus, in the example above, if the thief had introduced the error that caused the agent to malfunction, the designer's actions would have had nothing to do with the resulting harm. The designer would not have acted negligently with respect to that harm.

If two or more people jointly act negligently, each will be held responsible for causing the negligence even though only one person could have caused the injury. If the harm results from successive acts of different people, it is up to these people, and not those who suffered the resulting harm, to prove who was responsible.

2.2.3. Harm Within Scope of Legal Responsibility. Third, the harm has to fall within the zone of responsibility society has decided to impose on its members. Society's rules usually limit responsibility to reasonably *foreseeable* injuries caused by reasonably *foreseeable* events.

An agent designer's lack of care that causes an injury would not be negligence if she could not have reasonably foreseen the nature of the injury (rather than its extent). For example, it might be concluded that an agent designer could not have anticipated that the error in the agent that causes a sleeping laptop to make a loud sound would cause an avalanche in the Swiss Alps.

An intervening act also complicates the responsibility analysis. An agent designer remains responsible if the intervening act is a normal response to the situation created by the designer's negligence. So if the laptop's unexpected beep caused someone carrying the laptop to drop it, a court could determine that the designer should have reasonably foreseen this would occur.

2.2.4. Absence of Defenses to Negligence. Even if the designer's behavior has met the other requirements for negligence liability, the injured party's conduct may negate or decrease the designer's liability.

For example, a test version of an agent includes negligently designed code but is accompanied by a booklet warning the user not to install the agent on a KTel computer. The victim ignores the booklet, installs the agent on a KTel computer, and suffers an erased hard drive as a result. In a very few states, the user's behavior would be called *contributory negligence*, and eliminate his ability to recover anything at all from the agent designer. In the rest of the states, the liability would be calculated by comparing the designer's fault with the user's. In a *pure comparative fault* state, if the designer were 10% at fault and the user 90%, the user would still recover 10% of his damages. In a *partial comparative negligence* state, the user only recovers if the user's own fault is less than some cutoff level. If the applicable state law sets a 50% threshold, the user who is 90% at fault would recover nothing.

Another potential defense for a designer is *assumption of risk*. Assume that the user read the booklet accompanying the agent, including the large, bold type warning that use on a KTel computer could result in an erased hard drive. The user nevertheless operates the agent on a KTel computer and suffers an erased hard drive. Because the user recognized and understood the danger but voluntarily

chose to take the risk anyway, most states would relieve the designer from negligence liability.

2.3. Strict Products Liability

Creating a software agent may result in liability without any designer fault. Strict products liability recognizes that with modern technology and mass production, injuries will occur without intentional misdeed and despite reasonable care. Individual consumers would find it difficult to prove negligence. Producers, on the other hand, can absorb or insure against a loss more easily and are better able to take measures to reduce the occurrence of injuries.

Section 402A of the Second Restatement of Torts provides for strict products liability for physical harm caused by the sale of a "product in a defective condition unreasonably dangerous to the user or consumer." The seller must be one who is in the business of selling the product. A product may be in "defective condition" because of defective manufacturing, defective warnings to the purchaser concerning the product's dangers, or because of defective design.

An agent has a manufacturing defect if it leaves the manufacturing facility in a condition other than that the manufacturer intended. Thus if a disk duplicator generates a flawed copy of a software agent that as a result is more dangerous than the ordinary consumer would expect, and the flawed copy causes a physical harm, strict products liability would apply.

An agent has a warning defect if the product lacks adequate warnings of danger and the product is unexpectedly dangerous or if the product is unavoidably unsafe and the danger is not reasonably apparent. The required warnings depend on the normal expectations of consumers, the product's complexity, the potential magnitude of the danger, the likelihood of injury, and the feasibility and effect of including a warning.

An agent has a design defect if the agent's design presents an undue risk of harm in normal use. In most U.S. states, a design defect exists if the risk of harm could have been eliminated without a serious adverse affect on the agent's utility. In some U.S. states, however, a design defect exists if the agent did not perform as safely as an ordinary consumer would expect. In a few U.S. states, a design defect exists if either a feasible design alternative existed or if ordinary consumer safety expectations were not met. No matter which test applies, an injured consumer could have a difficult time making the necessary proof.

Strict products liability is restricted. It only applies to tangible products. It does not apply to services. Thus a custom designed agent or an agent used to deliver a service may not fall within strict products liability protection. On the other hand, an agent licensed in mass market distribution would probably qualify as a product. Strict products liability also does not apply to instructions, information in books, or other intangibles. Courts have been reluctant to let liability of this kind chill freedom of expression. Although a 1991 decision[2] suggested a willingness to treat computer software differently from recipes and guidebooks [1], so far no court has followed the suggestion. Finally, most states do not allow recovery under strict

products liability for purely economic losses. This limits the applicability of the doctrine to situations in which a software agent inflicted a personal injury or property damage.

These general principles of liability are raised by agents as well as other products and services. We now explore the specific features of agents that raise liability concerns.

3. Liability Unique to Software Agents

It is important to explore the differences between agents and conventional software. This difference has been succinctly stated as agents having a "sense of themselves as independent entities" [9], but we should like to explore this more deeply. For in these differences we find new issues formerly clandestine in non-agent-based applications.

"*Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed*" [8].

Pattie Maes' definition reveals a great deal about what distinguishes an agent from conventional software. In examining it, we will describe what makes agents special, and why these distinctions are potentially prone to liability.

3.1. Autonomy

The first word of Maes' definition tells the most: agents are *autonomous*. This means that they somewhat control their own actions and do not depend on constant human feedback [2]. Contrast this with conventional software that operates synchronously with its user, accepts input when the user supplies it, and produces output that is causally related to the input. "Computers currently respond only to what interface designers call direct manipulation. Nothing happens unless a person gives commands from a keyboard, mouse, or touch screen" [9]. Direct manipulation has also been referred to as *explicit responsiveness* [5]. Agents, on the other hand, perform *asynchronously*, meaning they can assume a task and continue to operate without constant feedback from a user [2]. This translates into greater freedom for agents, as a human need not constantly supervise them. It also raises issues with respect to reliability, since a user will not be present to arrogate control should something go wrong. Autonomy raises more concerns when coupled with mobility, because then "not only can you not see what the agents are doing, they may be off doing it on the other side of the planet" [2]. Asynchronous behavior is not present in the direct manipulation paradigm of conventional software.

Autonomy poses unique concerns for designers with respect to *causality*. In using conventional software, the user is seen as operating a tool to achieve his goals. If something goes awry, the damages can often be easily traced to a user error. If a car is driven straight into a wall, assuming no extenuating circumstances, it can only be the driver's fault. The autonomous nature of agents, however, complicates

assigning responsibility to the user. If an agent takes actions that result in damages, it is unclear who is liable. The user is not directly in control over the agent's actions and cannot be expected to have insight into all of the actions of their agent. A tempting defense for a user whose agent caused damage might be "but my agent did it."

Legal defenses of this kind have not succeeded [17].[3] The rationale Steven Miller states curtly, is that "there is no such thing as a 'computer mistake.' Microchips are too dumb to do anything except follow instructions" [10].

In light of the unfeasibility of such a defense, users whose agents have caused damage will likely blame designers. The potency of this defense will only be heightened if the agent in question is autonomous because this absolves the user of direct responsibility for the agent's actions. In the need to assign blame, the court may be pressed to consider inadequacies in design. Moreover, as noted in the discussion of strict liability, designers can be of good intention and adequately test their products, yet still be found strictly liable if an agent has taken actions that caused damages [4].

The causation for any agent malfunction lies with a human, and the autonomous nature of agents may implicate designers more often than users.

3.2. Mobility

Maes' definition says that agents "inhabit some complex dynamic environment." This is of great concern from a liability standpoint. The ability for software to travel to a remote host and execute some sequence of actions brings forward two considerable problems: *security* and *privacy*. The former determines who has access to a system and when this access is granted. The latter concerns what someone can see and do once they get there.

3.2.1. Security. Mobile behavior, while exhilarating from a computational viewpoint, holds perhaps the greatest potential for security problems. "Security is a significant concern with mobile agent-based computing, as a server receiving a mobile agent for execution may require strong assurances about the agent's intentions" [2]. How can a machine that receives a remote agent be assured it is not a virus in disguise? Are designers responsible for everywhere their agent goes, and everything it does when it gets there? These are tough questions for which there is no precedent. The main danger with mobile agents is that often they perform actions the user cannot observe, and agents that are out on the network may have no means by which they can be recalled.

Some agent designers have resolved security problems by building safeguards into their systems. Telescript (of General Magic Corp.) places lifespan limits on agents in hopes of curbing a mutation that might turn an agent into a disastrous virus on the Internet [3]. But safeguards can reduce the flexibility of an agent, in Telescript's case by destroying it after a predetermined amount of time. Ensuring security is further complicated because mobile agents often must have all of the access rights of their user in order to accomplish their goals [2]. This leads to a

process called *delegation*: passing the user's identity to the agent. This is usually accomplished by giving the agent the user's identification certificate. However, these certificates are valid for finite periods, and a remote agent whose certificate suddenly expires can cause unexpected problems [2].

Hackers take advantage of insecure systems by breaking in from a remote locale. We would not want to consider users of mobile agents in the same category as hackers, even if an agent caused damage. However, despite the "good intentions" of everyday users, statutes in many states cover *any* form of unauthorized access, malicious or benign [17]. Users may be held accountable if their agents gain access to systems that they should not; and if damage is caused, it is possible that both the user and the designer may be implicated. In any case, agent accessibility protocols should try to incorporate some form of user-acknowledgment. This way the user can have more control over his agent's accesses. Unfortunately, requesting permission from the user every time his agent attempts to access a remote host is impractical for agents that zip across multiple servers.

Controlling agent access to remote hosts also means controlling access to system resources. Mobile agents can transport across networks and execute on remote machines. The system resources (memory and processes) of the remote host can be consumed by mobile agents when they run. In large networked servers, where these resources are in abundance, the negligible amount used by a mobile agent seems inconsequential. However, as more and more individuals connect to the network, smaller machines may play host to mobile agents. If a small machine is forced to lodge a high number of mobile "patrons" the availability of its system resources may become an issue. Determining who has the right to execute on remote machines will become a problem for designers and policy makers alike.

Consuming a machine's time and resources without explicit permission could be considered theft of property. One might argue that, if a person is networked, they have implicitly consented to outsiders using their resources. However, no one would argue that the networked user has also given implicit consent to network viruses landing on his machine. The line separating which mobile "sojourners" are permitted to use remote resources and which are not is unclear. The courts have dealt with unauthorized use of a computer's resources before. In *United States v. Sampson*,[4] the court ruled that "the utilization of a computer's time and capacities is inseparable from possession of the value of the physical computer itself" [17]. Theft of computer time is therefore treated as a "theft of property" under federal criminal law.

Designers must implement a technological infrastructure that allows mobile agents to access remote sites without unreasonably compromising security. People must be comfortable when an agent from afar pays a visit.

3.2.2. Privacy. If a mobile agent suddenly lands on your machine, how are you to know it can be trusted? Perhaps it is relaying information back to someplace without your consent. Agent-based espionage may become a new form of hacking. Additionally, if agents confer together and share information about their users, how can users be assured their agent does not reveal sensitive information [7]? Mobile agents raise these concerns.

We cannot expect agents to share the same kind of common sense and ethical judgment that we as humans try to exhibit. Steven Miller points out that the Internet is a place where good intentions must prevail. "The Internet works best when people obey the established 'netiquette'" [10].

Whether agents can be built with this concept of netiquette in all situations is a challenge for designers. Unfortunately trusting peoples' good intentions is not enough. Computers magnify privacy concerns because of the ease with which data can be accumulated, transferred, and copied. Some have found privacy so scarce a commodity on the network that they inveigh against keeping any data meant to be private on the network at all [12].

The law recognizes invasion of privacy as unauthorized access to or disclosure of *private* information [17]. No privacy right is violated by intruding into or disclosing information that is already public. Further, for information to be private, reasonably appropriate measures must be taken to protect information from becoming public.

Information revealed on a publicly-accessible web page is public. Other information stored on a publicly-accessible web server is likely to be considered public unless significant security measures are taken to block access. Allowing agents to retrieve information from your web site is like opening your home to public tours. You cannot complain about a visitor who, once given free access to a bedroom, looks inside a doorless closet.

Information stored on a server may not be the only private information agent technology puts at risk of becoming publicly known. If information on a publicly-accessible network server could be considered public, and thus retrievable by others without violating legal privacy rights, then perhaps unprotected mobile agents and any unprotected information they carry are similarly exposed.

The owner of a server has ample reason to design the server so it inspects any visiting agent as thoroughly as possible to assess the agent's design integrity and intentions. But the visiting agent may contain sensitive private data. Agent designer Alper Caglayan observes: "If the agent carries state around with it, how does it know that the server will not spy on that state or modify it?" [2] Caglayan worries that "a mobile agent which has previously visited other servers may contain private information, such as their prices for a good or service." If it is a bargain finder, for example, the agent "might contain a negotiating algorithm which it uses to determine which is the best offer it has received from the set of servers it visits."

Ensuring privacy becomes even more difficult if agents communicate with other agents about their users [7]. Mobile agents use a communications protocol — for example, Agent Tool Command Language (TCL), Knowledge Interface Format (KIF), or Agent Communication Language (ACL) — that allows agents to collaborate and share information [2]. "Collaboration can make it possible for collections of agents to act in sophisticated, apparently intelligent, ways even though any single agent is quite simple" [9]. Thus collaborative agents may result in the disclosure of private information. For example, if a collaborative agent which has access to its user's credit information "told" another agent that information without the user having been aware this could occur, the user might be able to recover from the disclosing agent's designer the damages the unauthorized disclosure caused.

Mobile agents need defenses against other agents, as well as humans. A "persuasive" agent could be built to try and extract information from other agents or copy any state a mobile agent has with it.

Users must know what information about them is seen by the agent, and what information a mobile agent might take with it across the network. Despite attempts to make agent-user communication less intrusive, it may be necessary to require agents to inform the user of their actions insofar as they pertain to information about the user. As Peter Neumann points out, "laws that make privacy violations illegal cannot prevent violations; they can only possibly deter violations" [12]. It will be a job for designers to make agents honor privacy.

3.3. Indeterminacy

Maes' definition also states that agents "sense and act autonomously in [their] environment." This section focuses on the environments in which agents operate and the role adaptivity plays in creating unpredictable actions. Two factors — indeterminate environments and indeterminate users — raise the legal issue of foreseeability.

3.3.1. Indeterminate Environments. Conventional software operates in a fairly restricted domain, accepting only a limited set of *predictable* inputs. Word processors, for example, accept text, graphs, charts, and tables, while the more advanced of these accept images, videos, and sounds. They accept mouse clicks on their toolbars and selections from their pull-down menus. Their behavior in response to these inputs is predictable.

Software agents, however, may operate in *indeterminate environments*. By this we mean to say that the environments in which agents operate can be dynamic and unpredictable, and the possible data that an agent might encounter is varied. The Internet, where the possible data encountered is immense and varied, is an example of an indeterminate environment. An agent used to navigate this environment, like Henry Lieberman's Letizia, must be flexible enough to interact with all the information confinable to a web page [6]. Ted Selker, in his development of the COACH Lisp tutor, faced the challenge of creating an agent that could offer help in a dynamic language of over twenty-five thousand commands [15]. Both agents were created to operate in complex environments that changed, where the possible data and input encountered is not altogether foreseeable. An agent that confers with other agents can also be considered operating in an indeterminate environment since there is no way to forecast the *exact* nature of all the agents one may encounter.

In such indeterminate environments, how can we be sure of the behavior of our agents? Is there any way to protect against all possibly unforeseen inputs? Unfortunately, "most intelligent applications are extremely fragile at the boundaries of their capabilities; we need to provide safety mechanisms that can detect failures of reasoning or negotiation" [2]. Certainly in some cases, protections would limit the flexibility of agents, disallowing their interaction in certain environments or with

certain types of data. A balance is needed that limits agents' exposure to uncertain environments yet allows them to flexibly operate in a variety of domains.

Agents are not only designed to inhabit indeterminate environments but also, as Maes' definition states, to "sense and act" in them. That is, agents not only receive inputs from these environments but produce outputs as well. Contrast this with conventional software that usually creates (or *is*) its own environment. Agents inhabit *other* environments that are exterior to themselves, such as the web or a database. The fact that agents sense and act in environments typifies their agency: unlike conventional software, they have the ability to affect changes outside themselves to their surroundings.

This ability to affect indeterminate environments raises the concern that some actions may cause unforeseen results. Peter Neumann observes that "in a complex system, it is essentially impossible to predict all the sources of catastrophic failure" [12]. In the first introduction scenario, an agent inadvertently caused damage to the environment in which it was a guest. It is important to take note that a mutated or malfunctioning agent is nothing more than a virus. "We might call viruses 'agents with attitude'" [2]. We might also, then, call agents viruses with good intentions. Agents affect their environment, and because environments are varied and complex, the outcomes of such effects are not always predictable.

We have shown that indeterminate environments exacerbate liability concerns because predicting how inputs will affect agents and how agent actions may affect their surroundings is difficult. After we discuss indeterminate users in the next section, we will show how indeterminacy in general is related to legal foreseeability.

3.3.2. Indeterminate Users. Users are part of an agent's "complex dynamic environment." They behave very differently and may use their agents for varying purposes. Many agents are designed to be *adaptive* so that they will self-customize to their user's patterns of behavior. As more information is gathered by the agent about the user — either through observation or explicit user feedback — the agent updates its model of the user and the heuristics by which it performs, makes judgments, etc. Adaptivity is a highly prized quality in an effective agent, and it has been cited as the key characteristic that will embody the future of agents [3].

However, the more adaptive an agent gets, the more indeterminate its actions and the effects of those actions. Artificial life agents, for example, are built with the ability to modify their own code. It can be very difficult to predict exactly how they may end up behaving, especially after many generations of "evolution." An agent's ability to adapt may result in extremely variant behavior not conceived at the agent's inception.

Adaptive behavior is made possible when the agent creates a model of the user. This model has been referred to as an *adaptive user model* (AUM),[5] which is maintained by the agent, and is constantly changing as the agent learns [15]. The user model enables the agent to make judgments about the goals of the user, then take steps proactively to achieve them. Two agents may be initially identical but over time may behave quite differently because their users' patterns and goals are distinct.

Contrast this adaptive learning with conventional software that has no knowledge of different users. A word processor treats each user equally, and the application is completely blind to each user's goals and patterns of behavior. Adaptive behavior is thus promising and concerning. We recognize its power, but it comes with the unfortunate tag of indeterminacy.

3.3.3. Foreseeability. Recall from the discussion in 2.2.3 that designers may be liable for the foreseeable results of their agent's use. And the indeterminacy we have discussed makes determining what is causally foreseeable more difficult.

The *indeterminacy of environments* complicates foreseeability. The fact that agents (especially mobile ones) operate in indeterminate environments means that the possible inputs and outputs are virtually infinite, constantly changing, and unforeseeable. Unforeseen data may enter a system and cause failure even though error protections were in place [17]. It may be hard to hold a designer responsible for failing to account for *all* of the myriad circumstances into which an agent may wander.

The *indeterminacy of users* causes similar problems for foreseeability. The liability for adaptive agent failures could be difficult to assess, especially if the agent is used over a long time, a shadow of its initial skeleton. The user has little control over how the agent learns, as the adaptive nature of an agent is a function of its design. However, this does not mean the designer is necessarily negligent. It may be unreasonable to expect her to foresee all possible adaptations the agent could make. Furthermore, myriad users mean myriad agents, all of which may form goals very differently. The fact that adaptive agents learn may be enough to excuse the designer for negligence since all adaptations to all possible users are not reasonably foreseeable. The ultimate behavior of the agent lies beyond that foreseen at its inception.

It remains to be seen how the indeterminacy associated with agents will play out in the legal arena. The laws that exist now cannot be expected to judiciously govern the idiosyncrasies of the computer age [12]. But as we have already stated, it is dangerous to adopt a mind set that an agent could be the culprit. Ultimately people are responsible, and it is likely that a combination of people will be involved should an agent-related problem occur. By being aware of more than just the foreseeable environments, users, adaptations, and behaviors, designers can better anticipate problems and protect against them.

3.4. Anthropomorphic Representation

Anthropomorphic representation is another common attribute associated with some agents and not found in conventional software. By *anthropomorphic representation* we mean to include a wide variety of agent traits: the ability for agents to interact with people on their own terms through natural language, graphical interfaces, gestures, personality, and generally anything which agents exhibit that is human-like. Stockbroker Stanley from scenario two exemplifies a high degree of anthropomorphic representation. Whether agents should be given graphical on-

screen representations is hotly debated.[6] People attribute intelligence and personality to media, especially on-screen characters [13]. While competence and trust are two characteristics that must be built into agents [7], graphical agents can be imputed with greater competence than they deserve [13]. This danger of attributing undeserved competence and trust to graphical agents is the source of criticism against anthropomorphizing agents [14].

A human is guilty of misrepresentation if they falsely present themselves and their expertise. Moreover, if a customer relies on the misrepresented information for a product or service, the deceiver will be held liable. However, studies have shown that people relate to media as they do other people [13]. Therefore, one might argue, anthropomorphized agents should be subject to misrepresentation laws. Especially when an agent is portrayed as a specialist in some field (e.g., Stockbroker Stanley), people will attribute a high degree of competence to the agent [13]. Should people come to rely on that competence for weighty decisions, the agent's representation could have serious repercussions. Of course, since we cannot hold the agent itself liable for misrepresentation, we would likely point to the designer for misleading customers.

Interfaces should not be considered less prone to liability than underlying code simply because of their unquantifiable nature. Many cases have occurred in which an interface was responsible for disastrous consequences.[7] Interfaces have been the subject of major court cases as well, notably the "look and feel" lawsuits in the 80s. It is unlikely that the advent of computer agents will receive any less attention. In light of the possible interactions people will have with increasingly embellished graphical agents, "designers of human interfaces should spend much more time anticipating human foibles" [12].

4. Design Solutions

Thoughtful agent design can reduce much, perhaps most, of the potential legal liability. Agent design is very young. It presently lacks the traditions, conventions, and community of the more established technologies. Designers must act more proactively to establish these standards and make agents safer.

A definitive study of measures to limit designer legal liability for agents is beyond the scope of this paper. But a few examples will illustrate the kinds of measures possible and how preventative design can limit the liability exposure a designer must face.

4.1. Autonomy

Although technology may permit delegating far-ranging power to agents, agent designers must consider the appropriate allocation between agents and their users. Allowing agents to make crucial decisions without human intervention can be unreasonably dangerous. "[O]ne clear lesson that can be learned from [computer malfunctions] is that, in view of the enormous harm which may result from it, total

reliance on computers probably does not constitute an exercise of reasonable care" [17]. An agent should not be the ultimate authority for any crucial decision.

Autonomous agents should be built so that they only make the decisions possible without the benefit of human reasoning. Jim Moor distinguishes decisions made under clear standards from those made under fuzzy standards [11]. A clear standard is one that can be applied solely based on objective facts. Thus "shut off power if speed exceeds 120 km/hour" states a clear standard. Agents can make intelligent decisions under clear standards. A fuzzy standard, on the other hand, cannot be applied without justifications as well as facts. An agent should not be entrusted to make a decision based on a fuzzy standard such as "purchase the fabric if it will look attractive." The agent would not be capable of making the necessary justifications.

The agent designer must critique and weigh as appropriate or not every decision to be made by an agent. The agent's decisions should only be accepted after its competence is rigorously tested. And humans should never allow an agent or any other machine to decide basic goals and values [11].

Designers also must ensure that the agents they design do not make decisions or modify their behavior too rapidly for reasonable human supervision. Agents are undoubtedly capable of deciding and adapting at great speed. Doing so with absolute precision is almost as certainly unlikely. Human oversight must be preserved even at the price of slowing agents to less than their maximum speed. Slowing the speed of agent adaptation reduces the possibility of drastic swings in agent behavior that could produce surprising and potentially dangerous outcomes.

Human oversight also requires agents designed so as to be functionally transparent to their users. A user cannot make proper decisions about when and how to use an agent (and when to terminate the agent) unless the user can understand what the agent can do and how it will decide.

To the extent a user lacks this essential transparency, the law is likely to assign the designer responsibility for agent misconduct.

4.2. Mobility

Allowing an agent to travel the Internet introduces special security and privacy concerns. But here again, appropriate design measures should help minimize the legal risks.

One agent behavior that can increase security is the use of feedback mechanisms. A feedback mechanism passes information from a mobile agent back to its user. For example, password passing is a feedback mechanism that guarantees human authority be given before entering a site. When a mobile agent encounters a site with security precautions, it passes back the password for the system to the user. If the user is authorized on that site, the user can provide the password and the agent can proceed. A password-passing feedback mechanism trades agent autonomy for increased security.

If feedback mechanisms are to work, mobile agent sites must be classified for privacy. Many network firewalls already include some kind of classification capabil-

ity. Mobile agents must be built with the ability to recognize the security level of a server and to respond accordingly.

Because a mobile agent is itself a source of privacy vulnerability, designers should take measures to protect information contained within or acquired by a mobile agent. Encryption technology can be used for this purpose.

Finally, recall mechanisms and a D-Day capability provides additional protection. A recall mechanism forces the agent to return to the user immediately after receiving its user's order to do so. A D-Day or timeout capability sets the mobile agent to self-destruct after some predetermined time or if the agent had not had contact with its user for some period of time. The D-Day or timeout capability reduces the prospect of a mobile agent wandering from site to site out of human control.

5. Conclusion

Humans are ultimately responsible for the actions of their creations. Heightened sensitivity to the predicaments posed will better enable designers to build trustworthy agents that respect their environments and users, as well as avoid costly litigation. We do not envision a future where agents are on trial for their mistakes, facing "death by formatting" if convicted.

Making agents a viable technology involves more than just technical prowess. Legal liability standards must remain appropriate in light of the evolving art of agent design. Informed designers therefore must devote attention not only to agent design itself, but to shaping the emerging policies which govern their art.

Notes

1. This scenario is adapted from [2].
2. Winter v. G.P. Putnam's Sons, 938 F.2d 1033, 1035 (9th Cir. 1993).
3. See, e.g., Walters v. First Tennessee Bank, 855 F.2d 267 (6th Cir. 1988).
4. 6 Comp. L. Serv. Rep. (Callaghan) 879 (N.D. Cal. 1978).
5. The COACH user model, and user models in general, is discussed in [16].
6. An article defending anthropomorphizing agents is [5]. For the opposing view, see [14].
7. See pp. 206–208 in [12]. This section details numerous catastrophes traced directly to interface design problems.

References

1. L. Burgunder, *Legal Aspects of Managing Technology*. South-Western Publishing: Cincinnati, Ohio, 1995.
2. A. Caglayan, and C. Harrison, *Agent Sourcebook*. John Wiley & Sons: New York, NY, 1997.
3. S. Ditlea, "Silent Partners." *PC Computing*, pp. 160–171, 1994.
4. D. Johnson, *Computer Ethics*, 2ed. Prentice Hall: New Jersey, 1994.
5. B. Laurel, Interface Agents: Metaphors with Character. In B. Laurel, ed., *The Art of Human-Computer Interface Design*. Addison-Wesley, Reading, Mass., 1990.

6. H. Lieberman, Letizia: An Agent that Assists Web Browsing. Proc.IJCAI 95, 1995.
7. P. Maes, Agents that Reduce Work and Information Overload. *Communications of the ACM 37*, 7 (July 1994), 31–40, 146.
8. P. Maes, Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Communications of the ACM 38*, 11 (November 1995), 108–114.
9. P. Maes, Intelligent Software. *Scientific American*, September 1995, 84–86.
10. S. E. Miller, *Civilizing Cyberspace*. ACM Press, New York, NY, 1996.
11. J. H. Moor, Are There Decisions Computers Should Never Make? *Nature and Systems 1* (1979), 217–229.
12. P. Neumann, *Computer-Related Risks*. Addison-Wesley, Reading, Mass., 1995.
13. B. Reeves, and C. Nass, *The Media Equation*. Cambridge University Press, New York, 1996.
14. B. Shneiderman, Beyond Intelligent Machines: Just Do It! *IEEE Software 10*, 1 (January 1993), 100–103.
15. T. Selker, COACH: A Teaching Agent that Learns. *Communications of the ACM 37*, 7 (July 1994), 92–99.
16. T. Selker, ed. New Paradigms for Using Computers. *Communications of the ACM 39*, 8 (August 1996), 60–69.
17. J. Vergari, and V. Shue, *Fundamentals of Computer-High Technology Law*. American Law Institute-American Bar Association, Philadelphia, PA, 1991.