THE LAW AND POLICY OF AUTONOMOUS SOFTWARE AGENTS

SUBMITTED FOR THE DISTINCTION OF

B.S. WITH HONORS IN SYMBOLIC SYSTEMS

Jacob Otto Wobbrock

Symbolic Systems Program

Stanford University

Spring 1998

TO MY FATHER

For his love of justice and his work to achieve it

TABLE OF CONTENTS

INDEX OF FIGURES

AUTHOR'S NOTE

The subject of this thesis fits nicely into the realm of Symbolic Systems. Evidence for this is simply the wide variety of references at the end. Among these are twenty-one sources on law, eleven on artificial intelligence, eleven on social studies, nine on autonomous agents, eight on philosophy, and a smattering of books and articles on human-computer interaction, software engineering, and cognitive psychology. This interdisciplinary approach to knowledge is what motivates the study of Symbolic Systems. It also is a key point of this thesis: The united efforts of experts from all of these disciplines will be required to make autonomous agent technology safe and effective. My hope as author is that an interdisciplinary perspective is adopted towards solving these problems.

1
INTRODUCTION

In the mean time I worked on, and my labor was already considerably advanced. I looked towards its completion with a tremulous and eager hope, which I dared not trust myself to question, but which was intermixed with obscure forebodings of evil, that made my heart sicken in my bosom.

– *Mary Shelley's* Frankenstein

New technology appears almost daily. Often the effects of the technology on society and the legal system are unpredictable, and discernible only through hindsight after notable change. And once in a great while a new technology appears that changes even the way humans think and live. However, a power that produces such unfettered change is also a power that can cause damage. The automobile made homo sapiens the most mobile creatures on earth, but at the expense of the environment. Atomic and hydrogen bombs changed the collective self-perception: Humankind had the power to entirely destroy itself. Genetic engineering promises almost divine control over life, but some argue at the expense of the collective morality. Computers have given us reign over information, and some think eventually cognition, but have led to over dependence on automation. (Consider the Year 2000 Problem.) In all these technologies there lies a double-edged sword. None of them have come for free, without consequences to the environment, society, social conscience, and human relationships. The more powerful technologies send ripples through all areas of life, from individual psyches to societal self-images. But many worry as Mary Shelley did that as humans continue to dominate the universe, technologies come closer to dominating humans.

Agents have the potential to be as big as any technology in human history. If agent design reaches its ultimate goals, humanity will have to share the throne of cognition with an artificial intelligence, perhaps even relinquish the throne altogether, as the genus of "intelligent things" will increase in population from one to two. But agents

will not escape the double-edged sword accompanying all powerful technologies. *Agents are an inherently dangerous technology and their design and regulation is crucial.* The dangers are commensurate to the potential: both great. Donald Norman, a founder of cognitive science, entertains "images of human-like automatons, working without supervision on tasks thought to be for our benefit, but not necessarily to our liking." However, to this utopian vision, Norman adds, "The main difficulties [are] social, not technical."[40:68] As soon as agents assume traditionally human roles of thinking and acting, even desiring and motivating, the cost of agent malfunctions skyrockets. And judging from the past, humanity's over-reliance on agents will be as commonplace as the dependence on electricity and computerization.

Despite the potential for disaster, agent technology can be safely developed if designers and policy makers take an active role in building the infrastructure necessary for this technology to flourish. The legal system, in particular, will have to adapt to consider autonomous technology. These pages provide a look at the legal issues that agents raise and some solutions to the dangers. Many issues explored are unique to agents and not raised with such urgency in other computer technologies. The hope is to offer a positive outlook on the future of agents, one which is characterized by careful design, engineering, and regulation so that the benefits of this emerging technology can be freely relished.

1.1 VICTOR AND DR. QUINCY

Victor threw his briefcase into the back seat of his sporty convertible and dropped into the driver's seat, exhausted from a long day at the office. He endured an onslaught of new information and procedures each day. It wasn't easy acclimating to his new working environment as the newest lawyer in the most prestigious law firm in LA. The coming night would not provide respite from his responsibilities, however, so he relished the time out that was his drive home…

The engine grumbled and surged one last time when Victor turned the key over and stepped onto his cold garage floor. He snatched his briefcase from the back seat and unlocked the house with a thumbprint.

"Welcome home!" exclaimed a voice. A monitor in the entryway of Victor's house lit up. A small gray-haired scientist rushed on to the screen. "At your service," he said.

"Dr. Quincy, what did you find today in the way of a new water-ski boat?"

"Here are the results of my search," the little man in the monitor replied. He was about three inches tall, wore trousers and a collared shirt, and had frizzy hair as if he had been electrocuted, a visage somewhat reminiscent of Albert Einstein. The big screen television in the living room automatically activated and a list of internet addresses materialized as text. The little scientist rushed off the side of his monitor in the entryway, seemingly passed through the empty space that was between there and the living room, and appeared on the large television screen next to the list.

"What is the lowest price you found?" Victor asked as he walked into the kitchen, all the while keeping his eyes fixed on the television across the counter into the next room.

"This site," the little character began, pointing to the third on the list, "listed a Mastercraft GS20 for $18,240."

"I don't think that's low enough. Redo the search tomorrow and search all sites, including international."

"Okay," the eccentric-looking man replied. The list vanished and the scientist stood waiting for his next request, rocking back and forth on his feet.

Victor fixed himself a drink. After a nip of scotch, he sighed and asked, "How 'bout some music, Dr. Quincy?"

Dr. Quincy snapped his fingers and the stereo sprung to life. Smooth jazz oozed through the speakers.

Victor ran over in his mind the things he had left to do before he could relax and watch television. As he enumerated the briefs, pleadings, and cases he had to review, his face dropped as if his mind hit a snag. "Oh, darnit!" he blurted. "I forgot the Rosenthal memo." Victor thought for a second and then looked up from his drink to the television, where Dr. Quincy was still patiently waiting.

"Dr. Quincy," Victor prompted.

"Yes, sir?" the little man replied.

"I forgot the Rosenthal memorandum at work. It is located on the hard disk in my office. Could you please retrieve it for me? If you want to, ask Lawrence where it is."

"I'll be right back," Dr. Quincy replied, and rushed off the screen to the left.

Victor carried his drink from the kitchen across the carpet and sank into a sofa chair near the television. He opened his briefcase, but before he could finish pulling his work from under the leather cover, the little gray-haired man rushed back onto the television screen.

"Retrieval successful!" the scientist chimed.

"Excellent," Victor replied. "Please print a copy for me."

"Printing Rosenthal memorandum…" the man replied. After a few seconds he spoke, "printing complete. You have a notice from Lawrence."

"Let me hear it," Victor asked.

"Lawrence reminds you to finish the brief by 2:00 p.m. tomorrow."

"Thank you, Dr. Quincy." Victor reached underneath the television and lifted the printout from its tray. "Now to get to work," he mumbled to himself.

After a few minutes a light chime rang once from the television. Often Victor ignored news notifications, but he realized he had not read it since lunch that afternoon. "Dr. Quincy, let me see the news for today."

"Retrieving news stories search…" the little man said, taking a pair of glasses and a notepad from his pocket, as if to read a list. "Here is the list of the stories you're following." Again, a list emerged for Victor to read.

"Let me hear the first story," Victor requested, still half focused on his work.

"I'll be right back," Dr. Quincy said. His list vanished and he ran off into the distance. The screen cleared and an attractive news broadcaster materialized. "Today's top story…" she began. Victor half-listened to the rest of her report. After she had finished she vaporized, and a second passed before the little figure ran back onto the screen. "Oh? My coffee break's over!" Dr. Quincy exclaimed, and threw a ceramic mug off to the side of the television screen.

"Dr. Quincy, I'll see the rest of the news later. Do me a favor. Show me the email from today."

"Here is your email," the little man said, and a list, much like the first, materialized against a deep blue background. "The email is prioritized, and some solicitations have been cached," he said. The little character rocked back and forth on his feet next to the top of the list.

"That's fine. I do not have time for all of it. Just read me the high priority ones," Victor ordered, looking back down at his work.

The little man read the first three emails out loud. The first, from Victor's wife, reminded him about their date tonight. Before the little man had finished, Victor looked up. "Uh oh! I forgot Marcy and I were supposed to go out tonight. Do me a favor, Dr. Quincy. Find theater tickets in one of the playhouses for *A Midsummer Night's Dream* and reserve us two seats, front row."

"Certainly!" Dr. Quincy tirelessly exclaimed. Once again he scurried off the side of the screen…

1.2 Discussion

If one had to place the above scenario on a timeline what year would be assigned? The turn of the millennium? 2020? 2050? While no single system today has all the capabilities expressed by the system in the scenario, the technology does exist in separate pieces—and the pieces are *agents*.

All of the technological feats performed in the scenario—except flexible natural language understanding—are realized today by a new software genre that emerged in the late 1980s and early '90s. The technology is called *autonomous software agents*. (Such agents are also called *softbots*, for "software robots.") Agent hype is charged with claims like the following: "Software agents will be accepted as a design paradigm like object-oriented programming or client/server computing."[7:19] However, many fear agents are a source of potential disaster, and their statements are equally charged: "Along with their beliefs and capabilities come the potential for social mischief, for systems that run amok, for a loss of privacy, and for further alienation of society from technology through increasing loss of the sense of control."[40:71] Should agents be heralded or feared? No one is sure what to expect.

Agents are a loosely and variously defined type, ranging from on-screen graphical personas to mobile sojourners that zip across the internet and gather information. Many agents exhibit intelligence, aiding users by suggesting courses of action or working on their behalf. These intelligent agents often have the ability to learn from their own actions, their user's actions, and even the actions of other agents with whom they collaborate. The paramount characteristic of many agents, which itself encompasses many of the above features, is *autonomy*. The ability for agents to act without user instruction, determine possible courses of action, observe users and implement steps to achieve intimated goals, and assimilate new information via learning is what makes agents autonomous.

The current state of agent technology could come close to implementing a system like that in the above story. The Microsoft Office suite contains an on-screen graphical agent, "The Genius," that was the source of inspiration for the description of Dr. Quincy. NetBot, Inc. released an agent named "Jango" that scans through hundreds of web sites to find the lowest price for new appliances and other consumer goods. "Ringo," developed in the Media Lab at MIT, has the ability to find and recommend films and music based on

the tastes and listening habits of the user. General Magic Corp. is fully invested in developing mobile agents that can travel across networks and retrieve information, and the Media Lab is working on protocols for agent-agent interaction. In combining these abilities, mobile agents will travel across networks and confer with other agents, sharing information and returning with the results to their respective users. “Hoover” is a news filtering agent that sifts through large databases and finds articles of interest for a particular user. Pattie Maes, an agent designer at the MIT Media Lab, created “Maxims,” an email filtering agent that prioritizes emails after observing user reading habits. And “Night on the Town,” by General Magic Corp., travels across the network and purchases theater tickets, makes dinner reservations, and orders flowers. The state of agent technology today is clearly impressive, and once agent designers have a history from which they can learn, the technology will undoubtedly become even more inspiring. As the network load capacity increases, microprocessors cycle faster, and protocol is established, agents will be as influential a technology as the GUI of the ‘80s.

But what if the scenario were different? What if something went wrong? A number of problems could have occurred during the story: The agent could have recommended a motorboat that turned out to be a stolen commodity; in selecting music, some audio files could have been erased or damaged; while traversing the network the agent might have entered a forbidden system and obtained classified information; the memorandum could have been damaged during transport; there may have been an unlawful disclosure of information between agents when collaborating; the agent may have accidentally deleted an important email or failed to prioritize accurately; and finally, when purchasing theater tickets the agent might have improperly handled the monetary transaction or bought fraudulent tickets. Certainly more malfunctions are conceivable. The potential for catastrophe deserves consideration.

Malfunctions have the potential to cause damage in all technologies from household tools to nuclear reactors. Regular “conventional” software is more dangerous

than most technologies, and "some have suggested that a unique feature of software is its inherent unreliability. This unreliability arises from complexity."[22:127] Agents suffer from this unreliability because of their nature as software. Agents, like no technology ever before, pose new threats to society because of their autonomous nature. No other technology has the ability to impersonate human beings; to think, act, be acted upon, and conduct itself in the *real* world.

Steven J. Frank sums up the inquiry well. He notices that with unintelligent technologies "[w]hen the agent of injury is a tangible device or product, attention is currently directed toward three sets of possible culprits." These are manufacturers, sellers, and purchasers. He asks a question of each of them: Was the product designed or manufactured defective? Was the product sold in a defective condition? Was the product used improperly? Frank points out that "[a]bsent from the lineup is the injury-producing item itself" and says that its "existence is relevant only insofar as it pertains to the conduct of human beings." But artificially intelligent devices paint a different picture. Frank worries that as technological devices "come to include electronic system capable of judgment and behavior, they too will become objects of direct inquiry." He makes the bold conclusion that "the standards by which humans and machines are judged will begin to merge as the tasks they perform grow similar."[13:110]

Deborah Johnson calls this "the unusual character of artificial intelligence. No invention before has promised to take over the thinking and decision-making functions of human beings."[22:10] Hydrogen bombs and nuclear submarines may destroy entire nations, but even these technologies require human operation. Agents, on the other hand, are subtly active, behind the scenes, across the network, and eventually in every computerized system. "Defining characteristics of agents include their autonomy and their ability to deal with imprecisely defined situations at the boundaries of their competence and awareness—exactly the combination that can get them, or users, into trouble."[7:257]

Agents will have to be corralled by technological and legal safety mechanisms. The policy maker will be as involved as the computer scientist in ensuring adequate protections. Technological devices can help agents achieve their potential, but technical solutions can only be successful for technical problems. The legal, social, and policy aspects of agents will have to be judiciously established so that the technology can safely flourish.

1.3 Objectives

Exploring the legal ramifications of autonomous software agents is the aim of this paper. These ramifications are felt scientifically, technologically, politically, and legally, and an interdisciplinary approach to the subject is required. The central thesis will become clear: Agent technology is potentially very dangerous and must be carefully designed and regulated. Computer scientists and policy makers will be equally vital in ensuring the safe implementation and operation of agents.

In Chapter Two the reader will learn what agents are. A history of agents is presented along with two agent architectures from the past. Modern softbots evolved from the classical architectures of the 1960s, ‘70s, and ‘80s into their current forms, and this evolution will be discussed.

Chapter Three is an exposition of the philosophy underlying agency, culpability, and ethics. This discussion will provide the philosophical foundation for the legal theories discussed later. The philosophical arguments will shed light on many of the areas of liability.

Chapter Four is devoted to agent liability. The discussion will first overview the law on the subject, and then discusses issues specific to agents. Much of the background in the section on agent technologies will be useful in the legal discussion, and the more

intimate the reader is with the abilities of agents the more pressing the legal issues will seem.

The designer and policy maker are both presented in Chapter Five with solutions for reducing the threat of agents. Abstinence from agent development is not the goal. The goal is to heighten awareness to the dangers of agent technology and suggest an infrastructure that must be in place before agents become ubiquitous. This section highlights this infrastructure and its technological and legal components.

This future of agents is foreboding unless the legal system is adequately prepared. But the potential of agents is magnificent. The challenge for designers and policy makers is to usher in such magnificence safely.

2

FROM AI AGENTS TO HCI AGENTS

Most people still believe that no machine could ever be conscious, or feel ambition, jealousy, humor, or have any other mental life-experience. To be sure, we are still far from being able to create machines that do all the things people do. But this only means that we need better theories about how thinking works.

– Marvin Minsky

As the vastness of information in the computerized era increases, the ability of an individual person to accomplish all their informational needs diminishes. The days of sitting in front of a computer, performing a singular task with singular attention are giving way to a new interaction modality. Computer usage is moving from *manipulation* to *delegation*. "[D]irect manipulation will have to give way to some form of delegation. Researchers and software companies have set high hopes on so-called software agents, which 'know' users' interests and can act autonomously on their behalf."[32:84]

Agents manage complexity. The utopian vision of autonomous servants performing tasks humans are unable or unwilling to do is embodied in the autonomous software agent. "Ideally, they will mimic just what an intelligent human would do, except with greater stamina and accuracy."[63:110] This ideal has been approached by AI researchers with varying degrees of success. One problem has been that "[c]ommon sense is not a simple thing. Instead, it is an immense society of hard-earned practical ideas—of multitudes of life-learned rules and exceptions, dispositions and tendencies, balances and checks."[36:22] Defining common sense is hard enough; building it is even more difficult. Still, even if artificial intelligence (AI) has not achieved the ultimate goal of crafting general intelligence, techniques have been developed which enable systems to reason and perform actions intelligently within restricted domains.

Knowing agents' past as well as their present and future is necessary in order to understand their purpose, function, and dangers. Although modern agents emerged in the

early ‘90s, their roots go all the way back to the “classical” era of artificial intelligence, beginning in the 1960s and continuing until the early ‘80s. The research done in this era produced the intelligence necessary for agents. Now designers are incorporating this intelligence into agents that interact with people, making applications easier to use and computer time more efficient. Cognitive scientists in the field of human-computer interaction (HCI) study and design agents to increase the usability of the human-computer interface. Agents began as stand-alone robots or applications designed for their own sake by AI researchers. Now agents are a tool, part of larger applications instead of solitary, and assist users in their tasks. HCI researchers use and study agents as a means rather than an end.

The contribution that AI has made to the modern HCI agent is worth appreciating. Since much of the technology for intelligence is the same, an understanding of agent architectures from AI will also be an understanding of the intelligence in the autonomous software agent. The intelligence of the agent is what gives it special abilities to carry out human-like actions, but it is also a tenuous notion, since misdirected “intelligence” can wreak havoc.

2.1 Classical AI Agents and Architectures

The longtime goal of AI has been to produce a system as intelligent and flexible as a human mind. But researchers in the field of AI and artificial life (A-life) have disagreed about the best way to accomplish this task. Different approaches have been tried, some with more success than others have. The history of approaches to agent design, from symbolic logic to bottom-up approaches and eventually to artificial life, is a fascinating study in its own right.

Approaches to Agent Design

Traditional theorists advocated a top-down approach to agent design, where high-level knowledge is programmed directly, often in a symbolic language like first-order logic. John McCarthy and Nils Nilsson of Stanford University are the primary advocates of this tactic, which "holds that thinking can be accomplished through the formal language of 'first-order predicate calculus.'"[14:38] But this approach has only limited success, mainly because "the quirky world of humans simply doesn't readily translate into the language of logic."[14:39] People do not perform decision making by logical proof when interacting with the real world. Advocates of the symbolic approach are still tenaciously conducting research, however. Douglas Lenat, also of Stanford, is developing Cyc, a huge database of facts and knowledge accompanied by symbolic rules for reasoning. Lenat is using a broader range of symbols and concepts than found strictly in first-order logic, but his approach is still top-down in nature. "Cyc is the boldest test yet of one of artificial intelligence's most cherished assumptions: that an ordinary computer can be hand-primed not merely with data and number-crunching abilities, but with knowledge and reason."[14:34] Most of the field is skeptical of his efforts, largely because they are merely a supped-up version of the symbolic approach from McCarthy and Nilsson.

The central problem with the symbolic approach is that logic is extremely formal, unrealistically tidy, and is unambiguous. The real world, by contrast, is messy, dynamic, and operates more on probabilities, such as "it might rain tonight." "Interaction with the physical world involves dealing with conflicting and contradictory information in a way that does not fall within the scope of decision making by logical proof."[19:241] "[I]t has become clear that it is precisely such features as everyday conversation, common sense, and creativity that have proven practically impossible to formalize satisfactorily."[11:54] Faced with these obstacles, AI researchers (Lenat excluded) have taken a step away from top-down approaches and searched for a model besides logic. They found it: nature.

The model AI researchers were looking for was right before them. Nature had provided a way to build intelligence from the bottom-up in human beings. The nature-based approach to intelligence observed that "[t]he brain alone is intricate beyond mapping, powerful beyond imitation, rich in diversity, self-protecting, and self-renewing. The secret is that it is grown, not built."[5:18] Researchers reasoned that they should be able to grow software as nature had grown the brain, though they could not afford the same leisurely billion-year pace! The nature-based approach to AI was advocated early by Marvin Minsky in *The Society of Mind*[36] which says that intelligence can be built from smaller pieces, unintelligent by themselves.[14:44] This approach believes intelligence is an *emergent* property of systems: If the right pieces are assembled the right way (*i.e.*, neurons in the human brain), intelligence simply emerges.[14:27] Rodney Brooks also advocated this approach in his AI robots. He developed a subsumption architecture upon which he built robot animals designed to move about their environment without hitting things, something symbolicists like McCarthy and Nilsson had done but with less success. "Instead of employing a 'top down' approach of explicitly programming in intelligence, Brooks insists that intelligence emerge on its own, in a 'bottom-up' fashion, through the interaction of relatively simple independent elements—as it apparently does in nature."[14:16] This approach is the hallmark of the later half of the classical era of AI, and led to the field of artificial life, or A-life.

A-life considers nature the ultimate model for developing intelligence. Researchers in A-life, or alifers, have taken the intuition of Rodney Brooks and Marvin Minsky to heart. Claus Emmeche, a Danish A-life researcher, shares the intuition that one should not "construct intelligent machines by means of programs made from the top down," which is the "dominant programming principle within AI." He says the top-down approach builds behavior *a priori* by "dividing it into strictly defined subsequences of behavior, which are in turn divided into precise subroutines, smaller subroutines, etc., all

the way down." The conviction Emmeche has is to instead use the bottom-up method which "simulates processes in nature that organize themselves."[11:19]

A-life programs often follow evolutionary patterns, where a thousand tiny programs "mate" with other programs and continue to solve a problem. Those that get closer to solving the problem survive, while those that are farther from solving the problem die off. Eventually after enough "generations" have passed, an A-life program will emerge capable of solving the problem, and the breeding ceases.

However, in both AI and A-life, the aim is the same: To create intelligence capable of operating in any domain, from "toy problems" like the Tower of Hanoi[50] to real world situations like flying jets.[60] In fact, despite the heated debates in which top-down and bottom-up theorists engage, many researchers believe the two approaches—AI and A-life—should overlap. "Intelligence, after all, is a product of evolution; and any behavior that enhances the prospects of survival or reproduction could be considered a form of intelligence. In a sense, biological life and intelligence depend on one another, and A-life and AI may prove to be equally codependent."[14:158] Claus Emmeche agrees by saying, "once life has emerged, intelligence may follow."[11:31]

The implications of emergence, as a design methodology, are frightening from a legal standpoint. Emmeche finds that the "most interesting examples of artificial life exhibit 'emergent behavior.'" He believes that it is "the bottom-up method that allows for the emergence of new, *unforeseen* phenomena on the superordinate level."[11:20] (Italics mine.)

The words "unforeseen phenomena" are used, and these are particularly frightening. The law defines a concept called foreseeability that is complicated by such unforeseen phenomena. (Foreseeability is discussed in Chapter Four.) It is sufficient here to flag this characteristic of agents, especially bottom-up agents, as one of great concern from a legal standpoint.

Whether agents today are built from the old top-down angle or the nature-inspired bottom-up approach, the legal issues they raise are potent. Certainly the liability concerns are intensified the closer the agent is to operating in the real world. An agent is unlikely to cause damage if it fails at a toy problem; the agent designed to run the Shinkansen bullet train in Japan could kill hundreds of people. Thus, as AI and A-life get closer to achieving their goals, the legal system becomes more crucial.

WHAT IS AN ARCHITECTURE?

AI agents are not singular programs built from scratch. Agents are built within an *architecture* that provides the underlying functionality. Architectures come in various forms that have been classified into different genres. Blackboard architectures,[17] reactive and subsumption architectures,[6] and cognitive architectures[1,50] are all recognized agent frameworks that emphasize different aspects of intelligence. Agents built within cognitive architectures, like Soar and Act, attempt to simulate human cognition and problem-solving. These two architectures are canonical examples and worth considering.

Before describing Soar and Act it is first necessary to be clear on what a general architecture provides. Architectures themselves are not agents. They provide a framework in which agents can be constructed. Agents built within a particular architecture leverage the functionality of the system. Reactive agents built with Brooks's subsumption architecture, for example, will exhibit the qualities of the architecture in being dynamic, interruptable, yet untaskable. However, agents built within the same architecture can be quite different. An analogy can be drawn to animals and their species. A gopher is similar in many ways to other gophers, but different in its particular behavior. Each instance of an agent is different but will be similar to other agents created in the same architecture. Russel and Norvig[51:36] capture this distinction in their equation:

agent = architecture + program.

The functionality of a complete agent is founded on the architecture and realized by the program. A creature that crawls along surfaces avoiding objects and a mechanical fly might be both built with a reactive architecture, but certainly the crawler program is different from the flier. The TacAir-Soar agents[60] were built within the Soar architecture and simulated automated pilots for use in air combat scenarios. These agents, while based on Soar, are different than the Soar agents used to solve the toy AI problems like the eight-puzzle.[50:495]

The architecture describes the "species" of an agent and its core capabilities. The actualization of the agent's behavior is dependent upon the program and will vary from agent to agent. Before describing Soar and Act, let us look at the components of a general AI agent architecture.

Six components are indispensable to an intelligent agent architecture. Each is explained below. The first four are aspects of the architecture's construction, whereas the final two are processes that the architecture supports. A graphical representation of a general architecture is shown below,[51:32] and arrows connect parts that share information.

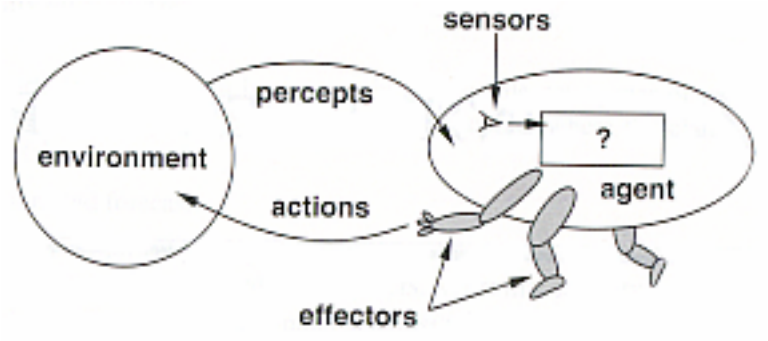

Figure 2.1. A general agent architecture. An agent interacts with its environment through sensors and effectors.

SENSORS AND EFFECTORS

The environment an agent inhabits, whether the real world, a network, the world wide web, or a simulation environment, plays a large role in determining the agent's behavior.[51:46] Agents may operate in an environment where all data is not fully observable at once. The environment is said to be *inaccessible*. The environment may be unpredictable, also known as *non-deterministic*. In a *non-episodic* environment, agents receive inputs continuously, rather than in episodes. *Dynamic* environments change while the agent is deliberating, as opposed to *static* environments that do not change while the agent is thinking. Finally, *discrete* environments have a limited number of percepts and actions, whereas *continuous* environments do not.

Environments that are inaccessible, non-deterministic, non-episodic, dynamic, and continuous are the most difficult for an agent to negotiate. These are also the most dangerous for agents to operate in and will be a source of liability. The real world is one such environment. Chess, on the other hand, is less difficult for an agent to master, as it is fully accessible, episodic, static, and discrete. The only difficult characteristic of chess is that it is non-deterministic, because the agent cannot know what its opponent will do.

The more flexible agents can survive in difficult environments; the more restricted agents can only function in specific circumstances for which they are designed. So chess playing agents would be difficult to adapt to taxi-driving.

Sensors are the mechanisms with which agents receive information from the environment. For robotic agents, sensors are often mounted cameras. Softbot agents sense the environment by receiving a data stream or parsing a text file. Most architectures support sensors that enable the architecture to receive environmental information. Accompanying the sensors is usually a mechanism that interprets the perceived information and translates it into a form the architecture interprets. The sensors and translators that are designed to operate in inaccessible, non-episodic, dynamic, and

continuous environments must be much more sophisticated than sensors designed for less challenging environments.

Effectors enable the agent to affect its environment. A robotic arm is an effector, or an agent's ability to write out data to a file or create information on a web page.[29] Without effectors an agent could only reason about the environment but not act in it to affect it.

Liability can occur in two places with respect to the environment. In the first, the sensors misinterpret what they see, and the agent incorrectly represents the environment internally. In the second, an agent miscalculates an action to perform with its effectors and causes damage. An ideal agent would reflect the environment consistently with its sensors and plan beneficial actions with its effectors.

WORKING MEMORY

After information from sensors and effectors is interpreted (*e.g.*, camera images are analyzed with a Gaussian or Lapacian method) it is sent to *working memory*. This is the active area of memory in which computation takes place. In blackboard architectures like BB1,[17] various knowledge sources share information by depositing it in the blackboard area. This blackboard area is like working memory, where current information is processed and handled. Information from the environment is integrated with the architecture's own information to arrive at the agent's next action. Cognitive agent architectures like Soar and Act also have areas of working memory, and will be discussed shortly.

Cognitive psychology has studied short-term memory in humans where information is quickly accessible but ephemeral, and limited to seven or eight items.[2:171] This is similar to *working memory* in cognitive architectures. The information is deposited only a short time while it is being used, and then is either discarded or stored

somewhere else. Working memory is similar to RAM in common computers. The capacity of working memory in cognitive architectures is limited, though the limit is much larger than the seven- or eight-item capacity of humans. Much like in humans, working memory in cognitive architectures is active. Almost all areas of an architecture affect the working memory.

Long-term Memory

Architectures must have some way to retain information about the world in a more permanent fashion. *Long-term memory* is a generic name for any declarative knowledge the architecture retains about the environment, past actions the agent has taken, and information gained from learning or inferring new facts. Like working memory, long-term memory is an meaningful concept in cognitive psychology,[2:180] and the ability of humans to retain large amounts of information in long-term memory is impressive. For example, the world record for reciting the most digits of pi (π) from memory is approximately 45,000! Amazing feats such as these seem impossible until one remembers that approximately 100 billion (10^{11}) neurons fill the human brain, connected by a million billion (10^{15}) synapses, and collectively these neurons fire 10 million billion (10^{16}) times a second—all while consuming less power than an ordinary light bulb.[14:98] So perhaps retaining 45K digits of pi isn't all *that* impressive…

Often in artificially intelligent agent architectures long-term memory may be encoded, at least in part, with *production rules*. These are formulations of conditions followed by actions, or *condition-action pairs*. An example of a production rule is, "*If* the sensors receive red light, *then* play the beep sound." The general form of a production rule is $p \rightarrow q$, read "*if p then q*." Each architecture handles production rules differently, but usually they are searched and the conditions are compared with the current state of the working memory or environment. The productions that have matching conditions are

selected. The architecture will resolve how to choose which one to execute, sometimes executing more than one or even all of them. The results of the executed productions are usually used to update the working memory, long-term memory, or other elements in the architecture (perhaps the learning procedure). The sensors observe the current state of the world after productions fire and the process repeats, matching productions in long-term memory and processing them in working memory.

CONTROL PROCEDURE

The architecture's *control procedure*, sometimes called a decision procedure, is often the most complicated feature of the architecture. The control procedure is responsible for manipulating working memory and solving the *control problem*: "Which of its potential actions should an AI system perform at each point in the problem-solving process?"[17:251] Barbara Hayes-Roth, creator of BB1, refers to the control problem as the quintessential hurdle an AI architecture must overcome. Part of the concern over liability lies in the realization that the control problem is not solved adequately in many situations, often because "maintaining a logically consistent rule is very difficult" within a larger rule base.[7:157] (Solving the control problem is *not* the most important task of an HCI agent, but instead effectively communicating and collaborating with the user.)

Cognitive architectures try to solve the control problem in different ways. Finding the best match with production rules is one way, then performing the action in the consequent of the production rule. If an architecture cannot solve the control problem it is said to be at an *impasse*. Soar has a unique method for resolving impasses and learning from them.[50:490]

The control procedure interacts with working memory and with long-term memory to determine the next action for the agent (*i.e.*, solve the control problem). By examining the working memory and state of the world, the control procedure searches

declarative long-term memory for relevant knowledge. This knowledge may be in the form of production rules or data structures. These are usually retrieved into working memory where further computation occurs.

Agents must determine *goals* for action. "An ideal agent knows what is goal is and will strive to achieve it."[32:85] The result of solving the control procedure is that a goal be formulated which the agent can attempt to carry out. Goal formulation is the output of the control procedure. The architecture must have some form of goal so that preferences for different actions can be gauged. For example, a mobile robot may have a high-level goal of traveling across the room without hitting obstacles. This high-level goal will be decomposed into subgoals as the robot proceeds: *move forward*, *detect object*, *turn right*, *move forward*, *turn left*, etc. Each of these subgoals, in turn, is decomposed into even smaller goals like *turn wheels* and *analyze camera image*. This subgoaling procedure is part of the control procedure and architectures handle it in different ways. When all subgoals are satisfied, the highest-level goal is satisfied. A danger with agent subgoaling, and one that can result in liability, is if the malfunction of a subgoal results in a malfunction in the entire goal structure. A safe approach is to design the agent so that the overall goal remains intact despite the corruption of a subgoal.

LEARNING PROCEDURE

Most intelligent agent architectures, and certainly cognitive agent architectures, have some form of *learning procedure*. This procedure enables the architecture to retain the results of its past actions in hopes of using that information in future decisions. "The basic set of agent functions is to combine existing knowledge content with new knowledge represented by events and to apply the reasoning or learning machinery to those to determine a desirable action or to extend the knowledge base."[7:136] The manner in which learning is implemented within an architecture is highly architecture-specific.

One thing learning procedures have in common, however, is that the learning mechanism interacts with the production rules and long-term memory. This interaction creates two capabilities. First, the learning procedure can observe which productions are matched and the results of those matches. Second, the learning procedure can create new production rules to reflect the learned information. As patterns develop and successful sequences of productions are executed repeatedly to resolve impasses, new productions are created that collapse these patterns into a single step. Learning procedures work with production rules enabling the architecture to perform better over time.

2.2 The Soar and Act Architectures

Soar and Act are two well-known cognitive agent architectures intended to model human problem solving. Soar and Act are designed in the spirit of the symbolic approach to AI, though they do not use first-order logic. Both have their own instantiations of the components described above, and their exact functioning is unique.

The mechanisms outlined above in the general architecture will now be explained with respect to Soar and Act. Sensors and effectors are omitted because they are often agent-specific and their existence can be taken for granted. The concern is more with the manner in which the two architectures solve the control problem and represent knowledge. For convenience, diagrams of Act[1:19] and Soar[50:474] are reproduced below.

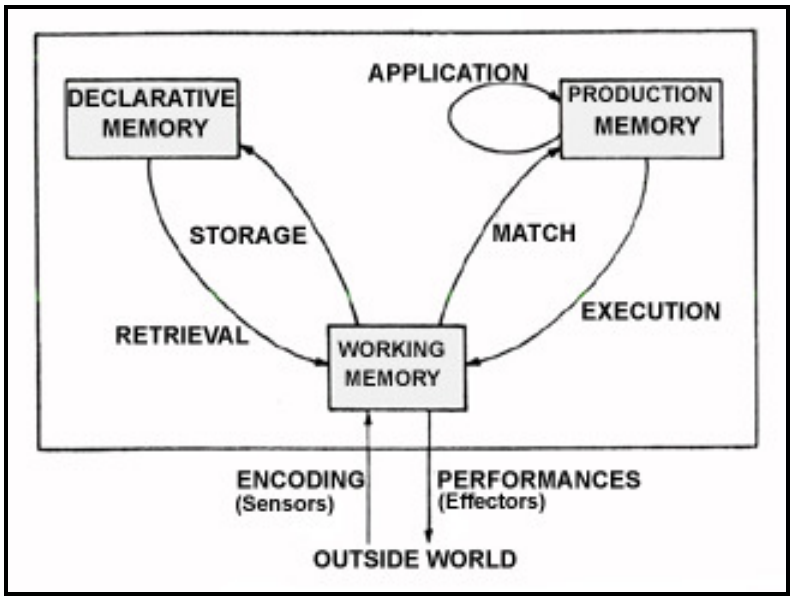

Figure 2.2. The Act architecture diagram.

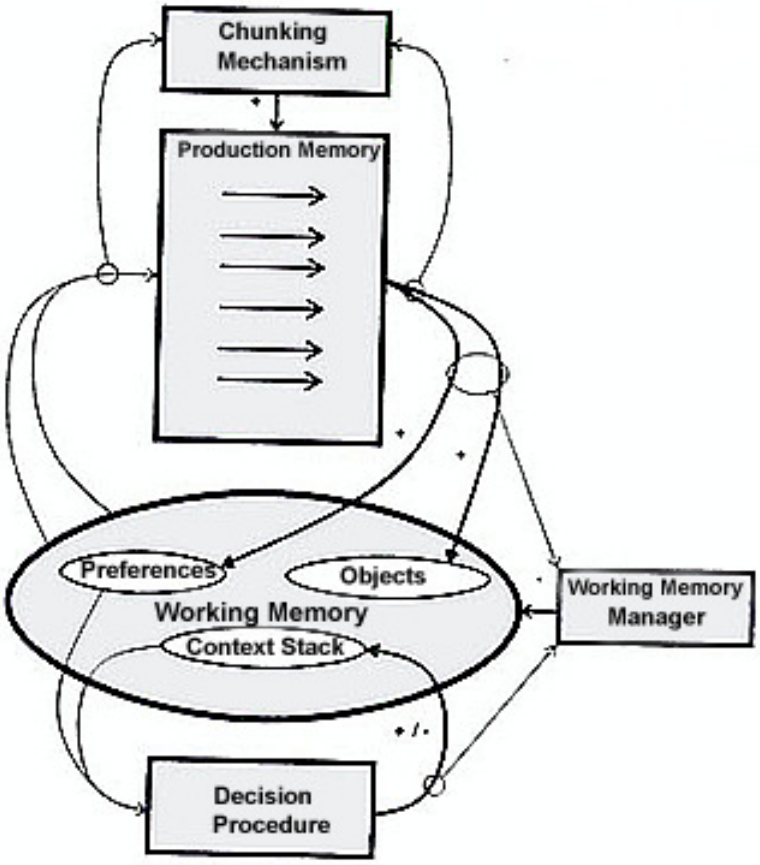

Figure 2.3. The Soar architecture diagram.

The computation occurs in the working memory in Soar.[50:475] Declarative memory elements—*preferences*, *objects*, and the *context stack*—reside here as well. Preferences aid in the control process by encoding knowledge that helps the architecture choose which operator to apply. (Operators are discussed below.) Objects encode the architecture goals and states that compose the problem space to be searched. Both preferences and objects can be deposited in working memory by the control procedure from long-term memory. Finally, the context stack contains the goals on which the architecture is currently working. Subgoals are more recent and therefore higher on the context stack, and the current goal is the top element on the stack. Links from the goals in the context stack connect to their corresponding goals in objects. The goals in the context stack are only one of four elements in the context stack. Four *slots* are present for each entry in the context stack, one containing the goal, the others describe the problem space, state, and operator.[50:481] These slots provide a description of the system at each point in its operation. Soar is designed to operate on a single goal hierarchy[24:202] (*i.e.*, it concentrates on solving one goal or subgoal at a time). The working memory encodes this goal hierarchy.

The working memory in Act is also used to contain relevant information about the current operation of the architecture.[1:20] Working memory contains elements from long-term memory. These elements encode declarative knowledge: facts about the environment, past actions, and the current goal. Production rules from long-term memory examine the contents of working memory and operate on it. Working memory in Act is also modified by incoming information from the environment. Sensors retrieve environmental information and deposit it in the working memory. All of these elements will be discussed more thoroughly below, after discussing long-term memory.

Long-term Memory in Soar and Act

All long-term memory in Soar is encoded in the form of production rules. The production rules are condition-action pairs that, when matched, will modify working memory by placing new objects and preferences there. Productions can never be removed from production memory, only added through a learning method called *chunking* that will be discussed below. Production memory in Soar is opaque, meaning that productions cannot observe and modify other productions in long-term memory. Production rules only support exact matching. That is, the entire conditional must be satisfied in order for a production to be retrieved. Once retrieved, all selected productions are fired in parallel. The working memory in Soar is distinct from the long-term production memory. That is, production rules are not explicitly retrieved into working memory, but instead able to modify the elements of working memory such as objects and preferences.

The structure of Act's long-term memory is different from Soar. Rather than have a single, localized bank of long-term memory, Act makes the distinction between declarative memory and production memory. Declarative memory in Act is encoded as *chunks* (do not be confused with Soar's learning method, *chunking*). These may be propositions, strings, or even spatial images.[1:23] The chunks encode a relationship between data elements. These elements can contain subelements in a recursive manner so in principle an unlimited amount of information can be contained in a chunk. Knowledge chunks are retrieved from declarative long-term memory into the working memory when their activation is high. This will be discussed more in a moment.

Act attempts to model spreading activation in human cognition by interconnecting declarative knowledge structures so that "activation" of one structure can flow to another. Thus, related declarative knowledge elements will be retrieved into working memory because activation will spread between them. This is based on the cognitive psychology principle of *priming*.[2:183-4] Each declarative memory structure has a strength associated

with it, and, like in human cognitive theories, activation flows from highly activated nodes to weaker ones. Declarative elements with higher activation have a better chance of being matched by production rules in working memory.

Production memory in Act is separate and serves a different function from declarative memory. Act's production memory is encoded as production rules. The rules can be matched with the current state of working memory and then executed.

The decoupling of declarative memory and production memory is integral to the Act cognitive architecture. Conflict resolution is made easier because, as Anderson puts it, "the process of retrieving data from declarative memory does not have to compete with the productions that perform the task."[1:21] Another advantage is that the process of storing declarative memory and storing production memory is separate, so storing declarative memory can be made faster.[1:23] Soar does not decouple its declarative memory from procedural production memory, and this represents a major architectural difference between it and Act.

Control Procedure

In both Soar and Act, the primary function of the architecture is to process a goal hierarchy and exhibit successful behavior. However, the manner in which the control sequence flows in each architecture is different.

Soar's operation is searching a problem space. The problem space is formulated by a goal and start state interconnected by a series of intermediate states. Soar cognition moves from state to state in the problem space, firing *all* productions that match. After productions are fired, preferences are created for operators. Operators in Soar are what change from state to state. In Act, it is the productions themselves that change state. The most preferred operator is selected and then applied to the current state in the problem space.

The description of the general architecture mentioned impasses. Soar contains a procedure for moving past impasses. An impasse occurs in Soar when more than one operator has been preferred and the architecture cannot decide which one to choose. In this event, Soar has a unique feature of creating a new subgoal on the context stack designed to solve the impasse. This new subgoal is processed until a solution is found or another impasse is reached. New subgoals will form automatically to resolve any impasses that occur. The name for this procedure in Soar is *universal subgoaling*.[50:491] Soar performs conflict resolution in this way by subgoaling when unclear which operator to apply.

Act does not have operators but instead looks directly for the best production to execute. When productions are matched and fired, they modify the working memory and the declarative memory structures there. These structures, when their activation falls below threshold, are returned to long-term declarative memory, possibly changed from the time they were retrieved.

The cycle of execution in Act discovers all productions in procedural memory that match the conditions in working memory, namely the declarative memory elements. Declarative memory elements in working memory have activation levels, and more strongly activated items are addressed first by the production rules. Notice this is different from Soar, where all matching productions are fired simultaneously. Conflict resolution must occur in this second step so that only one production is chosen to fire.

Act works to satisfy a goal hierarchy like Soar. Goals in Act are elements from declarative memory that are stored in a goal stack,[24:203] much like the context stack in Soar's working memory. Act performs conflict resolution by a process called *satisficing*. This process refers to the way Act chooses which production rule to fire. Rather than gather all relevant productions like Soar, Act looks for one that is "good enough." That is, while Act is searching productions for a match, it does not perform an exhaustive search.

As soon as one is found whose result is above a certain threshold, it is deemed good enough and selected for execution. This resolves the control problem in Act.

The contents of working memory change when new declarative elements are retrieved or stored. Productions can only retrieve elements to working memory, and once their activation drops low enough, they are returned to declarative memory by the architecture. Spreading activation, as mentioned above, has an effect on which declarative structures are retrieved. Those items that are related to active structures are more likely to be retrieved.

Learning Procedure

An integral aspect of human cognition is the ability to learn. Cognitive agent architectures also address this feature of cognition. Soar learns through a process called *chunking*.[50:497] Chunking is derived from cognitive psychology. Humans perform chunking when smaller procedures are familiarized and chunked together to form a single process.[2:124] For example, reading is a chunking behavior. Though humans must recognize individual letters in order to understand a word, humans are not aware of processing on that level. Instead, people chunk the letters together and recognize a word as a whole. When especially engrossed with a novel people may even chunk words into sentences, becoming only aware of ideas and not individual words.

Soar's chunking methodology is similar. After an impasse arises and Soar resolves a subgoal, the results of this resolution are complied into a single production, placed in long-term procedural knowledge, and thus retained for use again in a similar situation. The original impasse cannot arise again because Soar will merely invoke the new production and resolve the problem immediately without subgoaling.

Act learns through a process called *knowledge compilation*.[1:34] Through this process, new productions are created as they are in Soar. When productions are used

frequently, they are collapsed into single productions and added as new to procedural memory. Anderson argues this process is consistent with the manner in which humans learn. He says, "For example, rather than verbally rehearsing the side-angle-side rule in geometry and figuring out how it applies to a problem, a student eventually has a production that directly recognizes the application of side-angle-side."[1:34]

Act also learns when declarative memory elements are modified by production rules and restored. The state of the environment is updated within the architecture this way and thus allows Act to more accurately reason about its actions.

Soar and Act both perform similar actions and have common elements in their architecture. But differences are also evident, for example, in long-term memory: Act's is segregated into procedural and declarative compartments whereas Soar's is unified. The contents of working memory are also different in Soar and Act, and the control procedure that modifies working memory operates differently. Additionally, Soar has no concept of spreading activation, and Act has no universal subgoaling mechanism.

Despite these differences, however, the emergent cognitive behavior in Soar agents and Act agents is similar. Agents created with both architectures have the ability to learn and adapt, plan and choose the best actions, and resolve conflicts in reasoning. Agents in both architectures are taskable and can function reasonably well in difficult environments. Taskable agents are able to assume a user-provided goal and work to achieve it.

From the exploration of the intelligent agent architecture one thing is clear: agent technology is complex, highly complex. Peter Neumann points out, "In a complex system, it is essentially impossible to predict all the sources of catastrophic failure."[39:271] The amount of software required is in the hundreds of kilobytes and is often very dense, such as LISP code. Furthermore, the same architecture will underlie various agents and be expected to perform in myriad environments. For a designer to foresee all the

environments the architecture could encounter is impossible. This issue will be raised again in the discussion of negligence.

2.3 The Modern Autonomous Agent

The massive agent architectures fell short of performing at human-level cognition. If they had not, characters like Data from *Star Trek: The Next Generation* might be with us today. Many researchers felt the problem was in the goal itself (*i.e.*, to produce flexible general intelligence). "Artificial intelligence researchers have long pursued a vastly more complex approach to building agents. Knowledge engineers endow programs with information about the tasks to be performed in a specific domain, and the program infers the proper response to a given situation."[32:85] But this high-level focus on knowledge and reasoning for generalized intelligence did not yield the flexible cognitive abilities in humans, and so AI researchers began to look elsewhere. "The approach that AI researchers had generally used—the deliberative thinking paradigm—had not yielded serviceable autonomous agents, so the promise of servile bots was never followed by the real thing."[20:237] Researchers looked for new methods at the same time that applications became more complex and the result was the invention of the autonomous software agent.

Why Softbots are Unique

The massive AI engines like Soar and Act described above are created with the goal of achieving general intelligence capable of functioning in myriad domains under strenuous circumstances. The AI research community has now turned its energies toward different endeavors. Though some researchers still pursue general human-level intelligence, many

look for ways to utilize the useful techniques the AI field had developed in smaller domains. The result is the new agents of today.

Rather than attempt human-level performance across unlimited domains, agent researchers have adopted a new directive. "The solution is to focus on task-specific agents for narrow domains."[7:17] The massive architectures are systems unto themselves, rarely embedded within other frameworks and used to power physical robots or robot simulations. The agents of the '90s are just the opposite. These agents rarely have any physical component and instead are often referred to as *software robots*, or *softbots* for short. Softbots exist not as stand-alone applications but as additions to existing systems. Legacy applications have been retrofitted to accommodate agents, and new applications are written with agent capabilities in mind.

Thus the focus has gone from standalone AI agents to agents that help users in applications or provide functionality to applications behind the scenes: the HCI agent. HCI, as a discipline, studies how "to make non-absorbent interfaces that take the tools out of the task."[54:69] The fusion between the user and the computer is made through the user interface, and HCI has as its goal to make this fusion as effective as possible. The softbot agent promises to aide users in their work by "simplifying the clutter in HCI screens and scenarios."[54:69] They "assist users in a range of different ways: they hide the complexity of difficult tasks, they perform tasks on the user's behalf, they can train or teach the user, they help different users collaborate, and they monitor events and procedures."[30:31] Softbots are an HCI designer's dream. Whereas AI agents are developed by cognitive scientists, the origins of HCI agent technology are in artificial intelligence, software engineering, and human interface.[7:6] That is, the HCI agent is a product of three disciplines, as shown below.

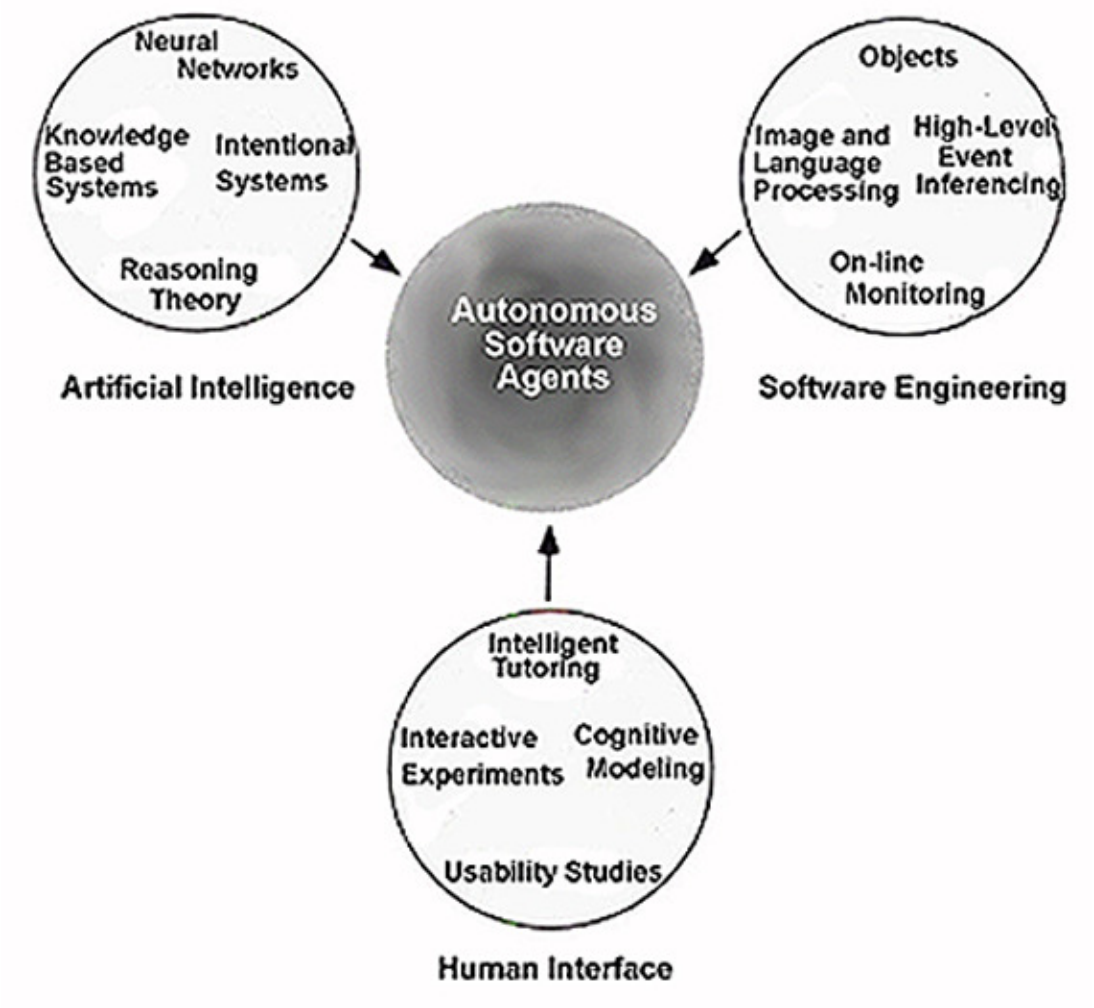

Figure 2.4. The interdisciplinary nature of agent design.

Researchers have had more success with softbot agents than AI agents because they have been able to focus more intently on a single domain of performance. "Instead of trying to model high-level functions like logic and program for every contingency, they would instead aim for interaction and learning. They set out to use this bottom-up approach, building robots, or 'animats,' that could perform simple tasks."[20:239] Specific softbots were eventually developed, and now exist for email, news filtering, web navigation, networking, E-commerce, databases, user interfaces, education, and secretarial work (see Appendix A). These agents are restricted to fairly narrow domains

of operation, and thus do not face the insurmountable challenges of the general intelligence architectures like those described above. As a result, the power of this new computing paradigm is being realized in industry and academic research.

AI agent architectures like Soar and Act have essentially two parameters: *machinery* and *content*. Machinery refers to the AI techniques used to power the architectures. This includes all the internals of the architecture, namely memory, procedures, and the organization of the architecture's components. Content is the data that the machinery utilizes. The data supplied both from the environment through sensors and effectors, along with new information from learning procedures, is the content for the agent.

The new agents have these parameters as well as two others not formerly present. These two new parameters are *access* and *security*.[7:129] Access is the domain in which the agent is allowed to function and the data it can see. "Access refers to the ability of the agent to observe and control its environment ... [T]he agent needs to be 'aware' of its world and to 'participate' in it."[7:201] The reason cognitive agents built within the architectures do not have this dimension is that they are generally stand-alone entities such as robots, simulations, or toy AI problem-solvers. They are applications unto themselves. Softbot agents inhabit other applications external to themselves and can affect those environments, possibly destructively.

Security is closely related to access. Security refers to the safety mechanisms the agent owns so that it will not cause damage to systems it inhabits. Security mechanisms are technological components: verifiers, filters, and other safeguards.

To sum the difference, then, between cognitive agents and softbot agents: Cognitive agents are themselves the application, but softbot agents are guests in programs external to themselves. This is why their construction is somewhat different. It is also why new agents raise potential legal and policy questions that older agents did not. If an agent malfunctions, it could cause damage to the system in which it is a guest.

The Technology of Softbot Agents

Modern agents of today share a lot in common with their massive architectural relatives and it would be repetitive to explain these features again. Structures for working memory, long-term memory, control procedures, and learning mechanisms are still present. Production rules still compose the knowledge base for agents, and usually the techniques for conflict resolution and inferencing strategies are much less complicated than architectures like Soar and Act. The reason softbots are less complicated than architectures is because their domains of operation are simpler and restricted. The agents do not have to be programmed for general intelligence in every domain.

Sensors and effectors still exist that allow the softbot to interact with its environment. However, a distinction exists here with physical robot agents. Softbots' sensors and effectors are purely *data interactive*, meaning that they simply interact with bytes rather than the physical world. Finally, softbot agents do not often exist on top of a framework architecture, but are programs unto themselves that exhibit intelligence, agency, and learning. That is to say, agents are not defined by the architecture they rest upon but simply by the qualities they exhibit. No longer is it appropriate to use the equation *agent = architecture + program*. The correct formulation now is *agent ISA program*. That is because softbots are restricted in their behavior and are much smaller than architectures like Soar and Act.

The diagram below gives an architectural view of the modern softbot agent.[7:153] The main differences to note are that all input from the environment and output to the environment must pass through access controls. Also, the learning procedure, which used to only modify long-term memory, now must also share information with access controls so that the agent can learn what is acceptable information and what is not.

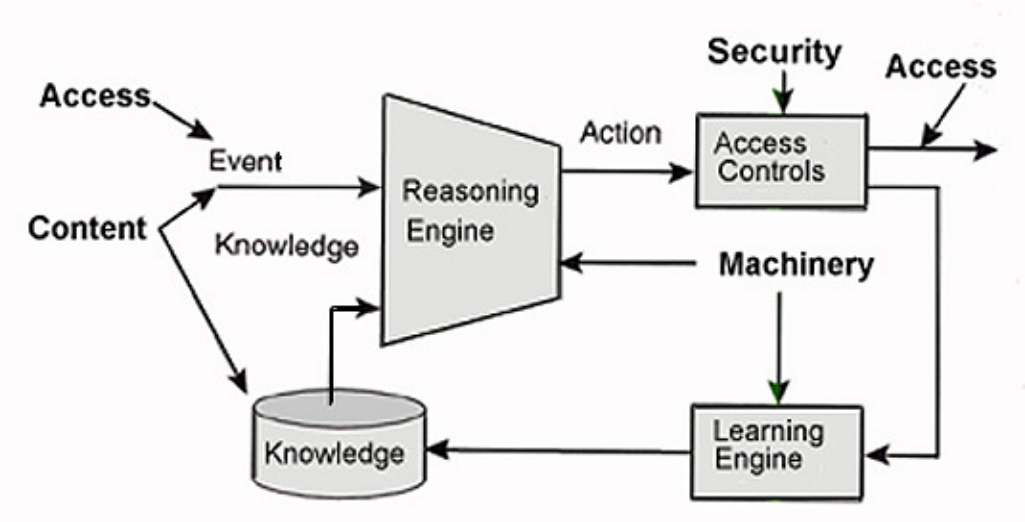

Figure 2.5. A general software agent architecture. Notice *Access* and *Security*, two components not present in AI architectures that do not inhabit other applications.

This architecture supports the new parameters of agents that distinguish them from the architectures. The access controls module ensures that sensors only allow certain information in, and effectors ensure the agent only affects the environment in appropriate ways.

The working memory and long-term memory are much like their counterparts in the large architectures, though the production rule base is much smaller because the agent is restricted in its domain of operation. The control procedure is also often simpler in a softbot agent than an agent built within an architecture. A simpler way to perform conflict resolution (*i.e.*, resolve the control problem) is probabilistic or Bayesian reasoning, which is "very helpful in dealing with ambiguity, where the result of reasoning is not a single, clear decision.."[7:155] The agent merely calculates probabilities for successful action and carries out the one with the highest score. The learning procedure for softbot agents is also simpler than Soar's sophisticated chunking procedure or Act's knowledge compilation. Statistical analysis, discriminant analysis, data mining, and non-symbolic learning are all techniques that achieve learning which do not involve massive

computational architectures.[7:157] Neural networks can also be used in softbot agents, for example Open Sesame! (see Appendix A).

While it seems convenient that softbots are less complicated than AI agents, and for researchers it undoubtedly is, from a liability standpoint the issue is not so clear. While complicated programs do contain more places for hidden failures to lurk undetected, the hope exists that intelligent behavior will be produced. Simple programs may be more easily verified as correct, but there is a limit to their sophistication. The simpler the agent, the less likely it will handle a complicated situation in a highly intelligent manner. Simplicity is not therefore a sole benefit. It can exacerbate the concern over liability.

THE "BIG FOUR"

Four characteristics are found in many softbots and typify the autonomous software agent. These are autonomy, adaptivity, mobility, and anthropomorphic representation. All four are not necessarily present in every agent, and only autonomy is necessary for a program to be considered an agent.

Software agents are coined by the term *autonomous*. Sometimes this is referred to as *proactive*.[53:93] The concept is simple. Agents are "proactive because they decide to do something without your asking for it."[9:170] Agents enjoy some control over their own actions and are not dependent upon constant human feedback. They perform asynchronously, meaning they can assume a task and continue to operate without constant feedback from a user.[7:125]

Autonomy is a characteristic that creates potential for liability. Agents are not manipulated directly by a user, so the user is not directly responsible for any damage. Consider the following characterization of "conventional" machines, like automobiles:

> If a machine can ever be said to have caused a problem, the machine is at most a victim of circumstances—people designed the machine, people built it, people chose to use it, and people controlled it. When a car crashes and causes serious damage, we may blame the driver, we may blame some other person at the scene, and we may blame the manufacturer, but we don't seriously blame the car.[58:161]

If the car had a mind, however, it might blameworthy. Agents do not fit nicely into the above commonsense characterization of machines. The autonomous nature of agents as a cause of liability is a major theme of this paper.

Softbot agents often learn. Learning allows them to be *adaptive*. Rather than exhibit the same functionality from one instant to the next, agent behavior can be self-modified based on past experience. For example, an email agent may observe that its user consistently saves email from Professor Chomsky. The agent creates a rule in its long-term memory for automating that process. Agents implement their learning using different methods. Firefly and Open Sesame! are both examples of agents that learn; Firefly does so using statistical techniques and Open Sesame! uses neural networks.[7:156] Depending on the agent's implementation, it may prompt the user before making such an inference or simply carry it out. "The assistant becomes gradually more effective as it learns the user's interests, habits, and preferences."[30:31] On the other hand, Soar, Act and other AI agents learn from their own experience rather than from direct user-interaction or feedback. This is because their purpose is not to assist a user, but to solve an isolated problem, such as navigating a robot through a room. HCI agents are adaptive so that they can better help humans with their computing needs.

Adaptive agents create a user representation, a model of the user's goals and needs.[7:125] Ted Selker, a researcher at IBM and creator of the COACH adaptive tutor, calls this user representation an *adaptive user model*, or AUM.[53:92] The AUM is built from the user's past actions and responses to agent behavior. If an agent observes a pattern in

user behavior, it will update its AUM and modify its own behavior accordingly. For example, the COACH tutor maintains a model of the mistakes students make and prepares customized examples. The power of adaptive agents is their ability to intelligently improve their ability to assist their user over time.

Adaptivity greatly exacerbates the potential for liability. An agent that has the ability to modify its own behavior is less predictable. Designers who develop adaptive agents may not foresee all the adaptations their agent could make. "Often we cannot test complex computer systems under every possible condition in which they will operate. This calls, some would argue, for an entirely new way of thinking about liability."[22:10] In fact, it is not altogether clear *who* the designer of an adaptive agent ultimately is, because so much of the agent's behavior may come after its release. If an adaptive agent causes damage, then, the designer could be exonerated because he is not the sole creator of the agent. Consider this point made about chess programs by Deborah Johnson.

> Admittedly, there are cases in which no human can fully understand what a computer system does … It may happen when computer systems, in some sense or another, learn. For example, a person who designs a chess program that learns from each game that it plays may not be responsible (in some sense) for all the program's wins and losses, since the program comes to know more than its designer.[22:139]

But the user is often unaware of the inner workings of the agent, so it seems just as inappropriate to blame her as it does the designer. Who is left? The agent itself? This is a legal conundrum that will be discussed later in greater detail.

Researchers in artificial life often cite autonomous movement as a key characteristic of many life forms. "One criterion for classifying artificial life as genuine life—for determining that it is just as alive as real, natural life—could be this very

autonomy, this capacity for self-movement. Inanimate entities like stones or rivers may also move, but they do not decide for themselves how to move."[11:23] Agents often have, as a defining characteristic, the ability to move. *Mobility* is the ability for an agent to transport itself, state data and all, from one server to another, executing and performing operations on a remote host. "[Mobile agents] are … capable of migrating themselves, including process and instance data, between the user's computer and one or more remote servers."[7:132] The need for mobile agents is a response to the unmanageability of the internet and its 13 million hosts, a number that is doubling every year.[7:49] The ability for a user to send an agent into "cyberspace" with a mission, then forget about it until it returns successful is a powerful concept. This is referred to as the "send and forget" agent model. The user sends out an agent with tasks and forgets about it until the agent returns.

Mobile agents are feared by many, and for good reason. In order to be as effective as their user in performing a task, a mobile agent must have all the access rights of the user.[7:245] This raises myriad technical concerns, even among the forefront agent researchers. Pattie Maes in an interview with *Wired* admitted to being "skittish" about mobile agents. "She believes the possibility for security problems is greater if an agent actually moves into, or resides in, a server—even if temporarily."[20:291] Donald Norman fears mobile agents in principle. "[T]he idea that autonomous, intelligent agents could have access to personal records, correspondence, and financial activities is disturbing to many individuals, no matter how helpful the agents might be."[40:70] The authors of *The Agent Sourcebook*[7] also find a number of hard-to-answer questions about mobile agents. They find that "[t]he whole idea of admitting foreign applications to a server" is reason to be nervous. The authors ask hard-to-answer questions like, "How do I know what a new mobile agent is really doing to do? How do I know that it is not a malign virus? How do I know who sent it? How does the sender of the agent know that the server will actually honor the agent's requests for service?"[7:231]

These concerns are well-founded, but some design precautions can be taken to limit the dangers of mobile agents. General Magic Corporation, the leading developer of mobile agents, has created numerous technical assurances to make mobile agents safe. These and other design solutions will be explained in Chapter Five.

Mobility exacerbates the uncertainty with which agents operate because of the myriad environments in which agents may find themselves. The variety of data, servers, and users that mobile agents may encounter is as big as the internet, and it can be impossible for designers to predict the environments in which their agents will operate.

> [A]n agent is different from an application program. We know—or at least we believe we know—what an application program will do, and we generally do not expect it to delete the file system. Agents by intent are much more flexible than an application program, and much harder to predict. This problem is particularly acute with mobile agent systems, where an unknown program may arrive and want to execute on one of your processors.[7:246]

Expansive environments like the internet (and the real world) have been called *open systems*. Open systems are characterized by continuous change and evolution, decentralized decision making, perpetual inconsistency, the need for linking components, and the inadequacy of the closed-world assumption.[19:223] (The closed-world assumption holds that if a proposition is not explicitly true then it can be assumed to be false.) A massive network of servers fulfills this criteria. Ted Selker refers to open systems as those "which are too big to be analyzed or which grow with use."[53:98] The difficulty for mobile agents in an open environment is that the data they may encounter could be unanticipated and the agent may not have the capacity to deal with it. For example, a mobile agent designed to operate on Windows servers may crash if it encounters a UNIX server.

Perhaps even more pressing than the technical uncertainties are the legal concerns regarding privacy and security with mobile agents. Mobile agents may accidentally obtain access to information illicitly, returning to their user and implicating him or her in the process, constituting an invasion of privacy. A mobile agent may cause damage on a remote server by accidentally deleting files or causing a system to crash, resulting in damage to property. Peter Neumann, who studies computer risks and failures, has found that networked systems are extraordinarily more prone to damage than other systems.[39:215] With the unlimited variety of data and systems present on the network, an agent is bound to encounter something unfamiliar, possibly with dangerous side effects. "[E]very move is an opportunity for transmission problems that corrupt the data; every combination is a chance for commingling of data that was collected using different data definitions; every calculation is a chance for a mistake."[35:270]

Some mobile agents have the ability to collaborate with other agents and share information. Agent collaboration has its obvious advantages: sharing information to maximize search time, delegation of various tasks between agents, and collective decision making. "Rather than having to be programmed for every possibility and every detail about a user's choices … agents fill the gaps in their knowledge by learning from their fellow agents."[20:290] But interagent collaboration also raises privacy concerns. What if an agent divulges information it should have kept private? What if "persuasive" agents are built which interrogate other agents into disclosing information? And security concerns are elevated since, while each agent may function soundly on its own, two agents interfacing may cause unforeseen problems. "Security plans must also take into account that, while an individual system may be secure within itself, a security gap may nevertheless exist at a point of interface between two or more systems."[62:253] The cyberspace conferencing of mobile agents exacerbates privacy and security concerns.

Criminal liability is also a concern with mobile agents. Viruses are mobile agents and could be called "agents with attitude."[7:240] (One article calls agents "viruses with

good intentions," highlighting their propensity to cause damage.)[18:397] Opening the door to agents must mean opening the door to viruses and the hackers that use them. A program masquerading as a mobile agent could be the tool of a criminal, and unfortunately "[i]t is impossible in principle to verify with complete certainty that an arbitrary program (such as an incoming agent) is not a virus."[7:249] Many organizations attach themselves to the internet without the proper security precautions, failing to "understand that their ability to get out is matched by others' ability to get in."[35:270] Mobile agents are a tool for criminals to gain access to servers, and even an unintentional trespass into a government computer could warrant criminal action. Clearly technical concerns over mobile agents are equally matched by civil and criminal legal concerns.

With the advent of the GUI in the early 1980s interface design and the discipline of HCI has advanced tremendously in constructing useful interfaces that place the focus on the task and not on the tools. Part of this advance has resulted in the development of on-screen graphical guides—interface agents—that help a user navigate through an application. These on-screen characters exhibit the fourth quality of agents, namely *anthropomorphic representation*. Anthropomorphic agents—which are not necessarily human figures but can be animals, objects, or even just faces—are used in popular application suites like Microsoft Office. "Multimedia, including graphics, video, and audio (text to speech), can enhance the communication experience with the user as can anthropomorphized agent 'personalities.'"[7:8] The theory behind on-screen agents is the same as for non-graphical agents: All agents help manage complexity and provide the user with the ability to delegate tasks.

A heated debate rages among HCI designers over whether agents should be anthropomorphized (*i.e.*, given graphical representation). Two of the most ardent debaters are Ben Shneiderman and Brenda Laurel. Shneiderman, of the University of Maryland, believes that anthropomorphism makes users feel like they are no longer in control of their software, and are thus no longer responsible for what happens.[57:100] He worries users

will place undue trust in the agent. His opinion is supported by research done at Stanford University by Byron Reeves and Cliff Nass. These researchers discovered that people impute high amounts of intelligence and personality to media, especially on-screen characters.[47:81-87] In fact, their research suggests that the way people interact with such characters, especially those with faces, is psychologically identical to the way they interact with real people! Peter Neumann shares the same worry over anthropomorphism: "People often tend to anthropomorphize computer systems, endowing the machines and software with humanlike traits such as intelligence, rational and intuitive powers, and (occasionally) social conscience, and even with some superhuman traits such as infallibility."[39:265] Brenda Laurel disagrees, however, in her assertion that "[t]he metaphor of character successfully draws our attention to just those qualities that form the essential nature of an agent: responsiveness, competence, accessibility, and the capacity to perform actions on our behalf."[27:359] Good evidence exists for this as well. The ALIVE system demonstrated that people's experiences are enhanced by interacting emotionally with graphical agents.[31:113] This debate is not likely to be resolved anytime soon until more anthropomorphic agents are in use.

Anthropomorphic representation could cause grounds for liability. The main legal issues relevant to anthropomorphic agents are misrepresentation, false light, and appropriation. If users come to depend on an agent's "expertise" and damage or loss occurs, the agent's designer may be liable for designing a product that misrepresented its abilities. False light and appropriation are invasions of privacy. This may occur with anthropomorphic agents if their likeness resembles a real person.

Agents created in artificial intelligence paved the way for techniques employed in the softbot agents of today. These new agents of the '90s have become popular in applications and assume a variety of forms. Their underlying structure and capabilities are derived from their architectural ancestors, but rather than attempt general human-level cognition, these new agents are restricted programs that exist as part of a larger

application. They employ a variety of intelligence techniques for agency and learning developed in AI research over the last four decades. Because of their position as part of other applications, the new HCI agents are equipped with somewhat different capabilities, and can affect the universe they inhabit. The ability to affect their surroundings is a dangerous one, especially since softbots do not depend on a user's hand to move their own. All softbots fulfilling HCI roles create concern from a legal standpoint because of their complexity, autonomy, and interaction with users in difficult environments.

3
AGENCY AND ETHICS

[S]omething in the idea of agency is incompatible with actions being events, or people being things.

– Thomas Nagel

Humans feel the need to base actions on philosophical foundations. For evidence of this, simply consider the myriad "philosophies" that exist: the philosophy of education, of knowledge (epistemology), of mind, of language, of science, of religion—even the philosophy of philosophy. Many of the principles that are established in these philosophies influence the human institutions that they most directly address. For example, the philosophy of language has helped cognitive scientists working in areas of artificial intelligence that deal with language recognition by a computer.

The law is influenced by philosophy as much as any other human institution. Humans feel the need to ground notions of accountability and culpability in sound philosophy. When a judge sentences a criminal to life in prison, surely his Honor sleeps better at night if he feels the criminal, by some philosophical rationale, *deserves* the punishment he received. If someone forced the criminal to commit murder, the judge would not likely find him culpable. It would seem that the criminal must be the *agent* of the liable action to deserve punishment, the one who willingly performs it.

What philosophical foundations form the bedrock of the legal system regarding liability? What is agency and how does it play a role in determining culpability? And what does all this have to do with autonomous software agents? Understanding the philosophical foundations is crucial to understanding and evaluating the law.

3.1 PHILOSOPHICAL AGENCY

Cognitive science is not the only discipline with a claim to the word "agent." In fact, philosophy has its own definition of an agent. Philosophical agency can be used as a framework for thinking about autonomous software agents.

Philosophically speaking, agency refers to the idea of an *actor*, one who causes change. Often in philosophical literature one reads of the *rational agent*, usually meaning simply a rational person.

> We can say, more generally, that in ascribing intentional agency to others, and in explaining their actions in terms of their reasons for action, we normally presuppose that they are rational agents: that they are capable of grasping and weighing the reasons for and against their actions, and of acting in accordance with what they see to be good reasons for action.[10:101]

Animals and the insane are also agents, though not rational ones. Their irrationality bars any ascription of moral responsibility. "Mere animals, who lack reason, are not responsible for their actions; nor are people who are mentally 'sick' and not in control of themselves."[46:122] Rational or irrational, the key characteristic in a philosophical agent remains simply that it can *affect* things. In fact, the autonomous software agent is called an "agent" precisely because of its ability to affect change through its effectors on the environment in which it acts. Consider a definition of autonomous agents by Pattie Maes, a researcher at MIT, and the emphasis she places on agency: "Autonomous agents are computational systems that inhabit some complex dynamic environment [and] sense and act autonomously in this environment."[31:108] An agent, both philosophically and computationally speaking, is not a passive entity. It is an actor and agent of change.

Culpability is inextricably bound to the conception of agency. To hold someone morally or legally culpable is irrational if they are not the agent that caused the harm. This is obvious. The catastrophe of wrongful imprisonment is that the person incarcerated

is not the *right* man (*i.e.*, not the agent of wrongdoing). Clearly in order to be justifiably culpable, one must be the agent of the cause of the harm.

However, philosophy has considered the idea that even the *right* man may not be culpable if he had no choice in the matter. If a person kills another but is forced to do it—for example, facing the threat of a nuclear warhead being launched into downtown Cairo—it is irrational to hold him morally culpable. The law recognizes this as well. A defense against criminal accusations is extreme coercion. So in order that a person should be considered morally reprehensible, it is necessary that the person acted *freely*. "In holding someone responsible for his actions, we suppose that he is in some relevant sense a 'free' agent; that he has, in the traditional terminology, 'free will.'"[10:102] The free agent is the morally culpable one. The unfree agent is merely an actor caught up in unavoidable circumstances, or suffering from an addiction, mania, or phobia.[65:205]

Thomas Nagel, a philosopher and legal scholar, admits that as humans "we feel that the appropriateness of moral assessment is easily undermined by the discovery that the act or attribute, no matter how good or bad, is not under the person's control." He continues by claiming that "a clear absence of control, produced by involuntary movement, physical force, or ignorance of the circumstances, excuses what is done from moral judgment."[38:25]

An old debate in philosophy still continues about whether anyone is truly a free agent.[65] Many philosophers define free will as "the ability to have done otherwise" along with "the ability to have chosen to do otherwise."[61:189] That is, in order to be truly free, an agent must have the freedom to chose to do other than she did, and then to have *actually done* other than she did. Many mechanistic philosophers consider the universe to be one of pure cause and effect, where all experiences—human and otherwise—are simply matter and energy and the interactions between them. The doctrine of *determinism* is the idea that every action is the cause of some effect, which in turn is the cause of some later effect, and so on *ad infinitum* so that free will is merely an illusion.[61:186] The legal system

and society at large, however, operate on the assumption that people, by and large, have free will, and free will is absent only in extremely coercive circumstances.

A final aspect to philosophical agency is the necessity of *intention*. The legal system poses much higher penalties for a wrongdoing that is intentional versus accidental. For instance, consider the penalties for murder (intentional) and manslaughter (accidental). "Intended and intentional agency also form the central paradigms of *responsible* agency. To act with the intention of bringing a result about is to make myself fully responsible for that result."[10:99] This is closely linked with the point about free agency above, because one who does not act freely is likely not intending to bring about the result for which they act.

Thus far three cornerstones to responsible agency have been established. First, the responsible entity must be the agent of the harm. Second, the responsible entity must be a *free* agent. Third, the responsible entity must *intend* to bring about the harm (though an agent who causes accidental harm is also culpable, but less so).

(Before proceeding it should be noted that not all philosophers believe that determinism precludes free will.[66:2-3] These philosophers are known as *compatibilists*, because they believe determinism is compatible with the existence of free will. Hume was a compatibilist. On the flip side, not all philosophers believe free will is a prerequisite for culpability. Utilitarians, who will be described shortly, find that it is effective to hold people accountable, regardless of the existence of free will. While both these camps make interesting points, autonomous agents will be considered in light of the more mainstream opinion that the existence of free will and strict causal determinism is mutually exclusive.)

This philosophical reasoning can be applied to autonomous software agents to discern their potential as responsible actors. If a software agent causes harm, certainly it is the agent of the harm. As noted above in Maes' definition, the agent affects the

environment, and in this case, causes damage. So the first condition on responsible agency is upheld.

Next, is the autonomous software agent a free agent? Despite the word "autonomous," the agent is still a computer program governed by 1s and 0s in a completely deterministic fashion. A program, by definition, is a list of instructions. While it is certainly true that humans can write programs which are far too complicated to be predictable—indeed, this is a danger with complex systems like agents—the behavior of the program *itself* is nevertheless completely predetermined. Usually it is just too hard to test and predict its behavior because of its complexity. So it is unclear how to interpret whether or not the autonomous agent is, philosophically speaking, a free agent. It would seem that the answer is "no," even though the behavior of the agent is hard to predict from a human perspective.

Finally, does the autonomous software agent *intended* to bring about a certain result? This question is also perplexing, because it is difficult to ascertain what is meant by "intended." Intention seems to require a desiring or willing to bring about a result, and the agent has neither of these. It does not intend to bring about a result any more than a rock intends to roll down a hill when pushed.

On philosophical grounds, then, the autonomous software agent seems to escape two of three counts. It *is* the agent of the harm, but it neither is free in its agency nor intentional. Human intuition is consistent with this conclusion. Most people would not hold a computer program responsible for any damage it did, even if the program is an autonomous one. This is, in part, the reason why the autonomous software agent is a dangerous technology. It has no sense of right or wrong, no will nor desire, neither compassion nor sensibility. This forces one to ask, "[F]rom a legal standpoint, should a user be held responsible for his or her agent's actions and transactions?"[30:40]

The same philosophy of agency can be used to discern whether the user should be held responsible for his or her agent's actions. First, is the user of an agent that causes

damage the agent of the cause of the harm? If the user is operating a piece of conventional software, the answer would be "yes" because of the user's direct manipulation of the program. With conventional software operated in the direct manipulation paradigm, a user's action directly corresponds to the software's reaction and therefore the user is the agent of the harm. "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."[32:84] But an autonomous agent is not controlled in this manner, if controlled at all. The agent acts on its own, and may operate only on a high-level goal from the user. So the user is not the agent of the cause of the harm.

Second, does the user of the agent enjoy freedom of the will with respect to the harm that is done? The answer is "yes" providing the user is not coerced in any way by the agent or any other force.

Finally, does the user intend to cause the harm? Well, unless we assume the user is a hacker and uses viruses, we must also answer "no." For the user may not even know about the agent's behavior, especially if the agent is mobile and off on the internet somewhere.

So just like the agent, the user seems to be exonerated on two of three counts as well. The only person left is the designer of the agent. Try the same analysis! The designer of an agent is not the cause of the harm, at least directly, perhaps like the designer of a gun is not the cause of death if someone else uses it lethally. The designer, like the user, is not coerced so he has freedom of the will when designing the agent. The designer is also not intentionally designing an agent to cause harm, so he is exonerated on the third condition as well. Who is left?

In the end a human being must shoulder responsibility for autonomous technologies. The technology itself cannot be culpable.

> [T]he kind of computers under discussion are not persons; and although they are causally responsible for their decisions, they are not legally or morally responsible for their decisions. One cannot sue a computer. Therefore, humans have not only an initial responsibility, but a continuing responsibility to raise the competency and value questions whenever computer decision making is at issue.[37:229]

Fortunately, the law imposes other mechanisms of accountability by which humans will be held accountable for the (autonomous) actions of their creations. For example the user of the agent may assume a risk in using the agent, and therefore be held strictly liable. The designer may be strictly liable or negligent in his design.

The key point is that a strong philosophical foundation of agency underlies the fundamental notion of moral culpability. The free and intentional agent is the culpable one.

3.2 Philosophical Ethics

Chapter Four will explain the legal doctrines of liability relevant to autonomous agents. Many of these legal doctrines are based on philosophical ethics. Ethical theories are often grounded in absolutist or relativistic claims. Absolutism and relativism are considered first, followed by two popular schools of ethical thought: utilitarianism and deontological theory. These philosophies underlie many of the laws on liability and culpability.

Absolutism

Theories of ethics and culpability are often grounded in an absolute claim. Take, for example, the sentiment that "murder is wrong." This general maxim is absolutist in nature because it is categorical. The claim is not that murder is wrong in some societies

but not in others. Similar absolutist claims are made about rape and assault and other horrendous actions. These things are not deemed wrong *a posteriori* within the context of a society, but proclaimed wrong *a priori* simply by their nature. They are not relative to their surroundings or the people that carry them out, but absolutely wrong.

Absolutists claim a *natural law* or universal moral code exists by which humans should guide their actions. The natural law is not something separate or external to human beings, but part of the very nature of human interaction. It is a property of the very existence of human beings and their social nature. The concept of natural law is a fundamental component of absolutist ethics. Absolutists point to the general sentiments across societies that seem to be consistent. "Moral principles such as 'never intentionally harm another person' or 'always respect human beings as ends in themselves' are of such generality that they could be operative in all cultures."[22:21] Even more specific evidence for a universal moral code has been found with respect to incest.[22:21]

For example, in order to argue that Hitler was evil, *absolutely* evil, one must assert an absolute criteria by which he can be judged as such. Without an absolute principle upon which to base a claim such as this, one could only argue that Hitler was evil in comparison to the benevolence of Mother Theresa or Martin Luther King, Jr. The relativist cannot rule out the possibility that if the world was different Hitler might be considered comparatively good.

Many absolutist principles in western society can be traced back to the influence of Judeo-Christian traditions. The Old and New Testaments of the Bible contain absolutist maxims. For example, one of the Ten Commandments is "You shall not murder" (Exodus 20:13). In the New Testament, Jesus speaks of "turning the other cheek" (Matthew 5:39) to avoid assaulting another person. The influence these religious traditions have had on the social psyche over the centuries is debatable, but undoubtedly they have made a contribution to a collective notion of right and wrong.

RELATIVISM

The theory most directly opposite absolutism is *relativism*. Ethical relativism holds that "[t]here are no universal moral rights and wrongs. Right and wrong are relative to one's society."[22:20] Relativists do not ascribe to the theory of a natural law, but consider ethical action to be society-dependent.

Ethical relativists point to a number of factors to support their position. For one, different cultures vary a great deal in what they consider right and wrong. In China it is permissible to strike children, whereas the Amish community abhors the practice. A second supporting fact for relativists is that even the *same* culture's ethics change over time. In Mormon society it was once ethical to practice polygamy, but no longer. Relativists would say that a Mormon from this era could not be held morally culpable for retaining many wives since it was considered ethical behavior at the time. Finally, ethical relativists point to the ways in which people develop their moral ideas. The influence of one's family, culture, society, and media determine one's moral values. A person born in the Middle East might consider it wrong for a woman to show her face in public, but a person from the United States does not find moral wrongdoing here.[22:20]

These facts used to support ethical relativism cannot be denied. But they do not *prove* whether or not relativism is right in an objective sense. Just because different societies have different values does not mean no natural law or universal moral code exists. "The fact that there is diversity of opinion on right and wrong is not evidence for the claim that there is no universal moral code. A moral code may apply to people even though they fail to recognize it."[22:20]

As with absolutism, relativism is not a provable position, but it has found its place in the legal system of the United States. That murder is wrong is an absolute principle, as discussed above, but the legal assessment of murder is relative to the circumstances. A soldier is praised for killing humans overseas in wartime, but the same soldier is court

marshaled for killing a fellow citizen within his own borders. An absolutist, on the other hand, might argue that killing another human is wrong, no matter who does it, when, where, or in what society.

Absolutism and relativism are general ethical positions that often make appearances in specific coherent philosophies. Two of these ethical philosophies are particularly prominent. The first, utilitarianism, is enormously relativistic but is based on an absolute principle. The second, deontological theory, contains mostly absolutist motivations. Both theories were developed prior to the founding of the United States and found their way into its moral and legal fabric.

UTILITARIANISM

John Stuart Mill wrote about utilitarianism in 1863. Utilitarianism is a form of consequentialism, a philosophy of ethics based solely on the outcomes of actions. Consequentialists put high value on some thing, such as justice or freedom,[10:105] and actions which produce those things are deemed ethical. Actions which do not produce the valued thing are deemed unethical. Mill surmised consequentialist ideology when he wrote, "All action is for the sake of some end, and rules of action, it seems natural to suppose, must take their whole character and color from the end to which they are subservient."[34:252]

Different consequentialist theories cherish a variety of different things, but for J.S. Mill and other utilitarians, the "most valued thing" is *happiness*. "Utilitarians conclude that *happiness* is the ultimate intrinsic good, because it is not desired for the sake of anything else."[22:24] Mill reasoned that all things which are sought after are desired for the sake of producing happiness. Take, for example, the desire to get high marks in school. This desire, Mill would say, is motivated by a desire to get a well-paying job after graduation. A well-paying job is not desirable for its own sake, but is desirable because it

allows one to buy the things he or she wants. These things are ultimately sought because they bring their owner *happiness*. This same style of reasoning can be used to reduce almost anything from marriage to running marathons to a desire for happiness. This brings forward the distinction between *instrumental* goods and *intrinsic* goods. Instrumental goods are desired because they bring about some other valuable thing. Intrinsic goods are valued solely for their own sake. Money is a canonical example of an instrumental good, as it has no value for its own sake.[22:24] The utilitarian concludes that happiness is the ultimate, and possible only, intrinsic good. All else works to its end.

Thus Mill went on to formulate The Greatest Happiness Principle: "The creed which accepts as the foundation of morals, Utility, or the Greatest Happiness Principle, holds that actions are right in proportion as they tend to promote happiness, wrong as the tend to produce the reverse of happiness."[34:257] The next project for Mill was to define happiness itself. He essentially formulated happiness as "pleasure, and freedom from pain."[34:257] He combined the Greatest Happiness Principle and his definition of happiness to formulate this overall theory, The Greatest Happiness Principle, which claims that "the ultimate end, with reference to and for the sake of which all other things are desirable ... is an existence exempt as far as possible from pain, and as rich as possible in enjoyments, both in point of quantity and quality."[34:262] For the utilitarian, this is the metric by which the morality of an action is measured.

Utilitarianism is absolutist in its assertion that happiness is the only intrinsic good for which all else should strive. But in practice utilitarianism can be relativistic. No single action can be pronounced categorically wrong under utilitarianism if, in some instance, it produces greater happiness than pain. Lying could be deemed wrong in one instance and right in another if happiness is maximized in both.

> What is the utilitarian's attitude toward rules such as 'Don't kill,' 'Don't tell lies,' 'Don't steal'? According to utilitarianism, such rules are *on the whole* good,

> useful, and worthwhile, but they *may* have exceptions. None of them is sacrosanct. If killing is wrong, it is not because there is something intrinsically bad about killing itself, but because killing leads to a diminution of human happiness … [*I*]*f* killing is always wrong, it is wrong not because killing is wrong per se but because it always and without exception leads to worse consequences than any other actions that could have been performed instead.[21:203-4]

Conversely, an absolutist would argue that killing is *wrong* by the very nature of the act, by the natural law governing such action. "Critics of the utilitarian method would say, for example, that lying is wrong even when, in a particular case, it produces a favorable result."[12:6] Utilitarians cannot pronounce an act wrong *a priori* but must instead look to the consequences of the act and then retroactively determine its ethical character.

Deborah Johnson raises an even more relativistic situation than lying when she writes, "Suppose, for example, that having a small number of slaves would create great happiness for large numbers of people. Those who were made slaves would be unhappy but this would be counterbalanced by marked increases in the happiness of many others."[22:27] In a society with many individuals, more slaves could be justified because more people would benefit from their labor. A society with fewer people might not be able to justify slavery because not enough happiness would be created to counter the unhappiness of the slaves. She gives another example of using one of every ten people as an organ donor, thereby saving ten lives for the price of only one. The point Johnson and other critics of utilitarianism make with examples like this is that utilitarianism "seems to go against some of our most strongly held moral intuitions."[22:27] Some adaptations of utilitarianism have a rebuttal to these criticisms, such as rule-utilitarianism, but common utilitarianism is subject to these criticisms.

Utilitarianism, or at least general consequentialism, is found as the basis of some laws. In civil tort liability the risk-benefit analysis is used to measure a product's worth

against its potential dangers.[22:25] This approach rings of utilitarianism. Strict liability, or literally "liability without fault," is also utilitarian. Finally, a consequentialist approach is often adopted when evaluating the effectiveness of laws. "If a law is not producing good consequences or is producing a mixture of good and bad effects and we know of another approach that will produce better effects, that information provides the grounds for changing the law."[22:27] Consequentialism lies underneath many laws in the United States.

Deontological Theory

Another system of ethics is deontological theory. Utilitarianism had its champion in Mill; deontological theory was espoused by the illustrious Immanuel Kant in 1785. Deontological theory is directly opposite utilitarianism because the latter focuses on things outside the action itself (*i.e.*, the consequences) whereas the former is concerned *only* with the action itself.[22:29] "[The] non-consequentialist view ... finds an intrinsic moral significance in intended action; a significance which depends not on its expected consequences, but on the intentions which structure it."[10:111] For the deontologist, the important thing is the action itself, its *inherent* moral character, regardless of the consequences. Deontological theory holds that some actions are wrong, no matter what the consequences may be.[22:30] And beyond actions, deontologists look at the will as a target of moral evaluation.

Deontologists place high importance on the human capacity for *rational thought* and behavior. The fact that humans are rational, they argue, makes them different from the mere pleasure-seeking animals that utilitarians make them out to be.[46:115-6] In fact, deontologists go so far as to say that devoting oneself to happiness is destructive: "[W]e find that the more a cultivated reason devotes itself to the aim of enjoying life and happiness, the further does man get away from true contentment."[25:8] Deontologists do not deem happiness as an intrinsic evil, but simply discredit it as the *ultimate* good.

"According to deontologists, the utilitarians go wrong when they fix on happiness as the highest good. Deontologists point out that this cannot be the highest good for human because if this was what we were meant to achieve, we would have been better designed without minds."[22:30] Dogs are a favorite example of deontologists as animals that are lacking in rational thinking but seem unconditionally happy, meandering from place to place seeking happiness as their sole purpose.

Rational thought is not the only cherished commodity for deontologists. The purpose of rational thought is to produce a *good will*.[25:9] In fact, deontological theory holds a good will in much the same regard as utilitarianism holds happiness. "There is no possibility of thinking of anything at all in the world, or even out of it, which can be regarded as good without qualification, except a good will."[25:7] A good will is thought to motivate actions which are good of themselves, and not solely good because of the consequences they produce. If a person tells the truth because he is afraid of getting caught lying, then such action is not praiseworthy, because it is not motivated by a good will. However, the person who tells the truth because of a motivation to do the right thing is controlled by a good will and is morally upright. In one of his most eloquent moments, Kant describes a good will as a jewel:

> Even if, by some especially unfortunate fate or by the niggardly provision of stepmotherly nature, this will should be wholly lacking in the power to accomplish its purpose; if with the greatest effort it should yet achieve nothing, and only the good will should remain (not, to be sure, as a mere wish but as the summoning of all the means in our power), yet would it, like a jewel, still shine by its own light as something which has its full value in itself. Its usefulness or fruitlessness can neither augment nor diminish this value.[25:8]

Kant is convinced that a good will is inherently virtuous even if the actions it attempts to motivate are thwarted by some other factor. "[The] non-consequentialist view ... finds an intrinsic moral significance in intended action; a significance which depends not on its expected consequences, but on the intentions which structure it."[10:111] For example, if a person tries to save another's life but fails in their noble attempt, their good will establishes them as a moral person. Their failure has no bearing on their moral fiber.

Just as a good will is predicated on rational thought, *duty* is built upon a good will. "Deon" is Greek for duty, and "logos' for science—the science of duty. One's commitment to duty is the product of a good will. "[W]e shall take up the concept of duty, which includes that of a good will, though with certain subjective restrictions and hindrances, which far from hiding a good will or rendering it unrecognizable, rather brings it out by contrast and makes it shine forth more brightly."[25:9] Kant believed that in a duty people have to one another for ethical interaction, and this duty is built upon the existence of a good will, in turn founded upon the very fact that humans are rational beings.

The trophy of deontological theory, which is built upon the foundations of duty, a good will, and rationality, is the famous Categorical Imperative. The Categorical Imperative is the deontologists maxim for treating other people ethically. It requires that a person act according to maxims that are worthy of establishment as a universal law.[46:114] It also requires that each person treat other people as an end and never simply a means to an end. In modern language, simply, to not use people.[46:117] Kant's conclusion gives us the following maxim: "Act in such a way that you treat humanity, whether in your own person or in the person of another, always at the same time as an end and never simply as a means."[25:36]

Deontological theory is absolutist in its assertion of the existence of human rationality, the intrinsic value of a good will, the universal call to duty, and the existence of the Categorical Imperative.

Utilitarians might make arguments against a deontological assertion that lying is wrong with statements like, "telling a white lie to avoid hurting someone's feelings is justifiable." The deontologist would argue that to cultivate the best will possible one must be honest in all circumstances. The argument is simply that human beings deserve to be told the truth because they are rational efficacious beings and should each be treated with respect.[22:31] This may go against the intuition of some, and most people in practice probably exhibit some mixture of utilitarian and deontological action, doing consequence-driven behavior and duty-driven behavior when it suits their needs (and conscience) most.

Deontological theory plays a large part in the legal system. While strict liability and criminal liability are utilitarian in nature, negligence and rights are deontological. Negligence considers the *duty* the designer or manufacturer of a product has to the person he or she serves, and if this duty is breached, the injured person can recover damages from the negligent producer of the product. The recognition of this duty is a fundamental feature of the law on negligence. It supports the idea that people are to be treated with respect and should not be exposed to damaging products or services.

Rights are associated with deontological theories as well. The Categorical Imperative can be seen as a "right" to be treated as an end and not a means,[22:33] a right to be the object of action motivated by a good will. The duty to other people is to uphold their rights. This is especially relevant to autonomous agents speaking of the right to privacy. Mobile agents and collaborative agents invade one's private data. The existence of this right, and its protection by various acts and statutes, is justified on deontological grounds. The right to privacy will be dealt with more thoroughly in the next chapter.

Philosophical thought underlies many of the laws governing civil and criminal liability. The recognition of free agents as responsible actors, subject to both the privileges and liabilities of such action in society, is firmly based on philosophical grounds. The body of liability law contains absolutist as well as relativistic elements,

often evidenced in laws that are deontological or utilitarian. An understanding of the philosophical foundations of agency and ethics provides a useful perspective when thinking about legal doctrines.

4
AGENT LIABILITY

More so than in any other legal field, the boundaries separating human from machine will be forced into focus by questions of tort liability.

– Steven J. Frank

All products have a potential to create liability if they cause damage or harm people. The designers and manufacturers of products also have the potential to incur liability if they perform their jobs negligently without due care. Computer technology is one of the most sophisticated technologies that people use on a regular basis. School children do not grow up playing with rockets or nuclear power plants, but they do gain exposure to computers and software. The complexity of a large software system easily rivals that of a rocket or power plant. And the complexity of the systems being produced is growing faster than programmers' ability to ensure their correctness, a mismatch that will continue to worsen.[58:184] The potential for software systems to bring about liability for designers is high because of widespread use and complexity.

Developers of computer software know programs contain bugs, and are aware that these bugs may lead to economic or physical harm, a concern that has been escalating over the last few years with the growing complexity of software.[52:21] The popular press and general public have not shared this concern, probably because of the pervading mentality that computerized information is somehow infallible.

Agent software is not simple. The exploration of agent technology in Chapter Two hinted at the legal issues agents raise, and in this chapter these issues are explored more thoroughly. The complexity of an agent system, as well as the difficulty in testing and ensuring its correctness, makes "agentware" perhaps the most powerful, and dangerous, software on the shelves and in the research labs. The four cardinal attributes of agents—autonomy, adaptivity, mobility, and anthropomorphic representation—

exacerbate multiple aspects of liability. The unique contribution of these qualities to sources of liability will be highlighted.

Agents must be considered under the law of products liability. This area of law can be partitioned into three categories: negligence, warranty, and strict liability in tort.[52:24] Agents also impact two areas of law outside products liability: privacy and criminal behavior. In these areas of liability, the qualities of agents exacerbate the legal concerns or make it difficult to apply existing laws. Though almost any product can be defective or cause harm, agents have an astounding potential to do so because they perform tasks formerly relegated to humans. The more power one delegates to an agent the more potential it has to be useful, but also the more dangerous it is. This is the unfortunate paradox that makes agent liability worth taking seriously.

4.1 Negligence in Agent Design

Negligence arises in a number of potential situations. The question of negligence goes to the conduct of the designer in the development of the product. In the development of agents, the main instance in which negligence arises is when the conduct of the designer or manufacturer is not above a "standard of care."

A designer can only be found liable for negligence for the foreseeable use of a product that causes foreseeable damage. Agents complicate the determination of foreseeability, a cornerstone of negligence, because of the variety of states and environments an agent can reach. Furthermore, when mistakes *do* occur, causation must be proved, involving a potentially long and complex chain of agent actions. In these ways negligence is complicated by autonomous agents.

What is Negligence?

The Second Restatement of Torts is an extremely influential treatise on liability law in the United States. It defines negligence as essentially the failure of a person to meet a standard of reasonable care. "Negligence is generally understood to be failure to do something that a reasonable and prudent person would do."[22:136] People can be negligent not only at their work but in everyday tasks like driving a car. The West's legal handbook on products liability notes that "[l]iability for negligence has come to extend across the whole spectrum of human activity."[26:118]

Designers and manufacturers of products are highly susceptible to negligence lawsuits. A product may be used by thousands or millions of people, and any carelessness in design will probably result in damage due to such a large user population. Errors in software systems, known as *bugs*, are often discovered when the system is released on the market. Bugs plague even the smallest of software systems, and programmers spend the majority of their time debugging programs. Bugs have been referred to as "a plague,"[58:163] a problem that will be prevalent no matter how fast computers become or how much memory they have. Designers may be found negligent if their products contain dangerous bugs. Moreover, autonomous agents are very complex and hard to test, and bugs in them are more elusive than those in conventional software.

Negligence law has a history dating back to the middle of the nineteenth century. Oliver Wendell Holmes wrote extensively on torts in general, setting aside a specific category for civil wrongs in which the defendant fails to perform a duty to society.[68:13] Prior to the 1830s, negligence cases usually referred to the neglect of someone to perform a specific duty imposed by contract, statute, or common law. (Common law is the law set down in court cases and by rulings of judges.) Two cases involving specific duties were *Patten v. Halstead* (1795) and *Lobdell v. New Bedford* (1804). The former case concerned the failure of a sheriff to keep prisoners in custody, the latter a town to keep its bridges in tact. Negligence in these cases was equated with *nonfeasance*, the omission of a prescribed duty. During the 1830s, however, New York, Massachusetts, and

Pennsylvania began to associate negligence with *misfeasance*, the performing of an act in a careless manner. Within a few decades a general standard of care was widely used to evaluate negligence cases, as in the famed case *Brown v. Kendall* (1850),[68:16] rather than a specific standard applied to persons within certain occupations as was done prior to the 1830s. In *Brown v. Kendall*, Chief Justice of the Massachusetts Supreme Court Lemuel Shaw implied that negligence "was more than neglect of specific duties imposed only under certain circumstances; it was the touchstone (and principle limiting factor) of a general theory of civil obligation, under which persons owed other persons a universal but confined duty to take care not to cause injury to another."[68:16] Negligence cases soon became just as much about malfeasance as they were about nonfeasance.

Today negligence retains the concept of a general duty to society imposed on all people. Negligence is based on common law, and "implies a standard of behavior that can reasonably be expected of any person engaged in a particular (often professional) activity."[22:137] In order to prove a person negligent in today's courts, four components must be shown by a plaintiff. The first is that the defendant has a duty of care to the plaintiff; the second is that a breach of that duty exists; thirdly, the plaintiff must show causation—that the defendant is the cause of the injury; and finally, there must be damages to persons or property. If any one of these is absent, negligence will be unsubstantiated. Three main defenses can be used against negligence in high-technology cases: contributory negligence, assumption of risk, and the "state of the art" defense.[62:206] The first refers to a situation where the plaintiff himself contributes to the cause of his injuries, the second when the plaintiff chooses to use the product fully aware of the risks, and the third when the damaging product is advanced technology and its risks are not understood at the time the product is developed.

Modern negligence law will continue to evolve in response to the technology industry. The duty of care for the designer of a highly complicated system must be determined, as well as the standard of adequate testing. Agent technology will have an

impact on negligence law, as the lack of precedent and complexity of circumstances will force careful interpretations of existing law.

Duty of the Agent Designer

Negligence is based largely on the deontological philosophy that producers of products owe a *duty of care* to consumers. Whether or not a person is found negligent is based on whether or not she failed to meet a standard of care in her work. Her call to uphold this standard is her *duty of care*, and it serves as the metric by which negligence is judged.

People are judged by two standards of care. The first is the standard of a "reasonable person." This standard is incumbent upon all members of society in all activities, whether "one is considering the conduct of a driver in operating an automobile, of a shopper in operating a grocery cart, or of an automobile manufacturer in designing and building an automobile."[26:18] If one behaves as a reasonable person one cannot be found guilty for negligence.

The standard of care may vary from one set of circumstances to another. Essentially three variables come into play in determining whether or not a person upheld their duty of care: the probability of the harm, the gravity of the resulting injury, and the burden of taking adequate precautions to avoid such injury.[26:18] Any given question of negligence will turn on the circumstances surrounding the particular situation. The duty of care for a manufacturer is described in *Boeing Airplane Co. v. Brown* (1961). The duty of care for a designer is somewhat different.

The second standard of care is higher than simply that of a reasonable person. This is the standard of care incumbent upon *professionals*. Professionals have a standard of care that is specific to their professional group. The five recognized professions are doctors, lawyers, clergy, architects, and engineers. These groups can be liable for *malpractice*, a term used to describe the negligent conduct of a professional. Doctors may

be liable for medical malpractice, lawyers for legal malpractice, clergy for clerical malpractice, etc. In a court of law it is almost always necessary to prove professional negligence with expert testimony from a member of the professional community. "Generally, professional groups must develop their own standards because members will know better than anyone else what is reasonable to expect and what is blatantly a lack of competence."[22:137] A professional has specialized knowledge and skills that will be called into question, and colleagues from his community must testify about his conduct in court.

Computer scientists and programmers are often referred to as "computer professionals." (For example, a recognized group of computer scientists devoted to social responsibility calls itself Computer Professionals for Social Responsibility, or CPSR.) However, they are not truly professionals under the law. Unlike members of the five professional groups mentioned, computer professionals are not professionally licensed and they do not share the special kind of relationship that exists between true professionals and their clients.[62:216] In general, courts have been reluctant to impose the higher professional standard of care on computer professionals, though in one case—*F & M Schaefer Corp. v. Electronic Data Systems Corp.* (1977)—a judge found a programmer liable for malpractice. This finding is the exception, though, and the courts almost always apply the reasonable person standard in negligence suits against computer professionals, especially recently as software programmers have become a large and varied group of workers.

Computer professionals *do* meet some of the requirements for professionalism, and agent designers meet these even more convincingly. They have highly specialized knowledge that requires years of education and they use this knowledge directly in their work. "Programmers and systems designers are thought to have a duty of care in developing software. When they fail to do what can reasonably be expected of software developers, they can and should be considered negligent."[22:137] Agent designers and AI researchers are some of the *most* specialized computer professionals. If the standard of

care for the agent designer is raised to that of a professional, designers will have to be extremely careful how much power they allow agents to exercise. “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.”[18:393] Unlike conventional software which is not autonomous, agents could cause all sorts of problems unbeknownst to their user. The only person culpable is the designer who created the agent’s autonomy in the first place. The duty of care incumbent upon the agent designer is not light.

Software professionals, and agent designers specifically, must meet the standard of care in their work, and this is most obviously seen in their testing of software systems. Software professionals may be found negligent for failing to test their products adequately.

> Testing is one area in which the issue of standards arises. A software developer may be accused of failing to test [*i.e.*, negligent] a piece of software adequately before releasing it to a client. The client would have to establish that the defendant had failed to do an amount or kind of testing that any reasonable, prudent software developer would do. Needless to say, it is sometimes difficult to identify prevailing standards for testing, especially in a field like software development wherein the technology continues to change rapidly. Moreover, testing procedures are likely to vary from one kind of software to another.[22:137]

The difficulty in testing software—and this is exacerbated for agents—is that software is highly complex; more complex in some ways than any other technology. “Software entities are more complex for their size than perhaps any other human construct because no two parts are alike… In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound.”[5:11] Software

professionals are not to be envied among engineers! They must test and re-test until they can be reasonably sure no bugs remain. Many are sympathetic to the difficulty involved in fettering out bugs in software, and this fact may help to protect software professionals from liability. John Shore defends the software professional by saying, "The inadequacy of software testing is not the result of incompetence. No matter how competent one is, it is impossible to expose all of the bugs in a program by means of exhaustive testing."[58:170] It remains to be seen how lenient the courts are.

How Agents Complicate Foreseeability

The duty of care for the producer of products does not extend to *any* possible use of the product in *any* conceivable circumstances. A baseball manufacturer is not held liable for negligence if a person is hospitalized for incurring injury in their attempt to eat the baseball! Negligence law instead imposes a duty of care for reasonably foreseeable injuries caused by reasonably foreseeable events.[18:394] The Second Restatement of Torts §395 defines foreseeability in three domains, summarized by West's Handbook: "First, the risk of harm must be foreseeable as well as 'unreasonable.' Second, the use to which the article was put must be foreseeable. Finally, the plaintiff must fall within a class of persons foreseeable as victims."[26:122] The duty of the designer or manufacturer is therefore limited by foreseeability. This does not mean, however, that the precise nature of the harm caused by the product must be foreseeable, only that some harm is foreseeable.

In *Daniell v. Ford Motor Co.* (1984), the plaintiff locked herself in the trunk of her car in an attempt to commit suicide, only to later change her mind and try to escape. She suffered psychological and physical injuries because the trunk lacked a latch by which it could be opened from the inside. The court granted summary judgment to the defendant, ruling that the actions of the plaintiff did not constitute a foreseeable use of the

product. The plaintiff's actions violated the second principle of foreseeability put forth in the Second Restatement of Torts §395.

The reader may have already anticipated the complication that arises with agents and the concept of foreseeability. Agent characteristics—especially autonomy, adaptivity, and mobility—make them particularly hard to predict. In fact, the entire point of agent technology is in the delegation of tasks to lumps of software that *do not* require a user to know their every action. This makes agents, from a designer standpoint, a nightmare to test. Observers of the software industry in general point to the frightening difficulty of fully testing computer programs to ensure their safety. A common conclusion is that "[n]o matter how diligently one tests computer programs, one cannot test them completely."[58:173] This is true for conventional software which operates in restricted domains and is predictable, but agents typify the testing problem in nightmarish proportions because they are autonomous and their behavior is up to them, not their user.[7:256] Testing an autonomous agent means, in some sense, getting the agent to test itself. That is, one must get the agent to perform as it would without human intervention.

Adaptive agents are extremely difficult to test and predict because of their myriad conceivable states. "[I]f the agent system has a learning capability, its behavior may well be nondeterministic: Outcomes will depend on accumulated knowledge and may vary depending on the order in which the knowledge is acquired (if learning and actions proceed simultaneously)."[7:257] When an agent designer builds an adaptive agent, its internal program structure is skeletal, a mere structure to support the learning and adaptations that will later take place.

From a legal standpoint, the foreseeability issue has become muddied. "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."[18:398] The designer is not in a position to foresee all adaptive agent states. Furthermore, the user of an adaptive agent is in no better position to predict the adaptations the agent might make. So despite

the fact that the designer may be exonerated for negligence for lack of foreseeability, no one else is in a better position to prevent the harm.

Products liability law considers alterations to products in determining liability. If a plaintiff is injured by a product which he unforeseeably and substantially altered before the injury occurred, his claim for recovery will likely be barred. However, if the damaging product is designed in such a way that it *invites* alteration, the court may find the damage foreseeable. In *Lopez v. Precision Papers, Inc.* (1985), the court ruled it was foreseeable that the overhead guard on a forklift truck would be removed because the truck was obviously more versatile without the guard, the guard was not welded down, and it was easily removable. The product invited the foreseeable alteration. Adaptive agents certainly invite alteration and, in fact, are designed *precisely for* alteration. Given that alterations to adaptive agents are foreseeable and intentional, liability for designers is heightened as unforeseeability is not a guaranteed defense.

Mobile agents, like adaptive agents, also complicate foreseeability. The almost infinite expanse of data on the internet and various intranets makes predicting the environments to which mobile agents have access more difficult. Errors in networks and distributed systems can proliferate at electronic speed[62:251] and mobile agents may encounter such errors in environments. Or consider a primitive mobile agent created before web scripting languages were in existence. The agent is designed to only parse HTML code. Predicting how the agent might behave in response to a site containing JavaScript is difficult. Mobile agents of today may handle all the contingencies currently in existence, but the pace of technology is so fast that it will not be long before some unforeseen circumstance emerges. With the enormous number of environments to which mobile agents have access, how the agent may behave and what data the agent may encounter in all situations is unforeseeable.

The fact that foreseeability is complicated by adaptivity and mobility has been called the problem of agent *indeterminacy*.[18:397] The adaptive behavior of agents creates

an "indeterminacy of users," and the mobile ability of agents causes an "indeterminacy of environments." When mobility is coupled with agents that can collaborate with other agents on the network, the "indeterminacy of other agents" is a third indeterminacy that complicates foreseeability. All three indeterminacies—users, environments, and agents—point to the same issue: From their initial inception, it is extremely difficult to foresee the range of circumstances in which a mobile agent may find itself. This complication of foreseeability may in some cases exculpate agent designers for negligence. Who then is liable? The agent? As Chapter Three concluded, an agent cannot be responsible.

COMPUTER "MISTAKES" AND CAUSATION

With the complication adaptive and mobile agents make to foreseeability, the case may arise when an agent causes damage that is not within the reasonably foreseeable set of events subject to the duty of the designer. The court may rule that the designer has no duty to foresee the adaptations that an agent may make years after its creation. As a defendant in a negligence suit, an agent designer may well argue that the autonomous, adaptive, or mobile nature of the agent made the resultant injury unforeseeable. If the agent which caused damage was a person—indeed autonomy, adaptivity, and mobility are concepts until now reserved mainly for people—the court would likely find the agent itself liable. But obviously the courts cannot hold an autonomous lump of software culpable. Still, perhaps the best solution is to discount it as a "computer mistake" and blame no person.

This type of rationale is known as the *computer mistake* defense: When something goes wrong in a complex computational system, the computer is awarded responsibility for the fault. "We're all quick to blame computers. We do it when we have billing problems, credit problems, delivery problems, reservation problems, and information problems. And if we don't blame the computer, someone else does—usually in a

defensive voice over a phone line."[58:161] The tendency to blame computers usually arises from an ignorance of computer systems. Computer systems are perceived by the masses as having a mind of their own, being as complicated as people, and prone to making the occasional error. The fact is, however, excluding the incredibly rare chance of a hardware malfunction, that there can be no *computer-caused* mistakes in computer systems. "Anyone who has experience with computers knows that when 'a computer makes an error,' it is because the program being used is faulty, or because data was put into the computer incorrectly, or because the computer system was not being adequately monitored by a person."[22:138] Computers, by definition, follow a prescribed set of instructions and operate completely deterministically. "[T]here is no such thing as 'a computer mistake.' Microchips are too dumb to do anything except follow instructions."[35:270] Even random numbers generated by computers are only pseudo-random, generated by a complex mathematical algorithm designed to defy prediction. Computers do not have free will.

This does not mean, however, that computers cannot *simulate* randomness, choice, and even free will. Their programming can be designed in such a way as to appear sophisticated in these ways. Some designers even worry that when a computer is represented as being intelligent, users will not feel responsible if something goes wrong because of this apparent computer free will.[57:100] Still, when a mistake occurs it can *only* be the fault of an unpredicted circumstance by the designer, programmer, or engineer, or an error by a user. The computer cannot just "make a mistake" randomly of its own volition.

The courts have not been sympathetic to lawsuits based on the "computer mistake" rationale. In *Ford Motor Credit Co. v. Swarens* (1969), one of the earliest cases to deal with this issue, the plaintiff purchased a car on credit through Ford Motor Company. A computer informed collection agents that the purchaser had fallen behind in his payments and subsequently repossessed the car even though the plaintiff showed

collectors his payment checks. Ford attempted to blame the mistake on the computer. The court did not heed the defense, awarding both the value of the car and punitive damages to the plaintiff. The court ruled that people must take responsibility for interpreting computer information in a prudent manner. "Trust in the infallibility of a computer is hardly a defense, when the opportunity to avoid the error is as apparent and repeated as was here presented."[62:213]

The courts are of the opinion that those who use computers must not use them wantonly, but with care and consideration of the information put forth. "An increasing number of modern courts have been resolving cases involving computer error by taking the position that, no matter what the source of the computer errors, over reliance on the product of data processing systems can be a source of liability for negligence."[62:211] People must not rely on computers to such a great degree that they fail to employ the essential human traits of judgment, intuition, and abstract reasoning.[41:13-4]

The ruling in the *Swarens* case is from an era that had not yet seen the widespread proliferation of software. It certainly had no vision of the autonomous software agent. Though the courts are not wishing to allow the "computer mistake" to establish itself as a viable defense against negligence, the picture is more complicated with agents. The view that computers will only do that which they are told is a perspective prior to autonomous software. A book on computer law states, "It has become fashionable to blame errors … on the computer, even though such errors are not ordinarily the result of any failure by the machine. A computer will only do that which it is instructed to do."[62:211] This opinion is from the direct manipulation perspective of software, not the delegation view which describes human-agent cooperation. While it is still true that computers must ultimately follow encoded instructions, as far as users are concerned an agent could very well cause an error which is not the result of a direct instruction. The courts must be careful in trying to pass any agent malfunction off as a *direct* result of some human folly.

Nevertheless, agents cannot be defendants in a negligence suit. Errors ultimately are due to human action. And despite their intelligent façade, agents are not blameworthy actors (see Chapter Three). The defendant in a negligence suit may be tempted to blame the computer system, but the plaintiff must show that the defendant—a person or organization, often the designer or design company—caused his injuries. The plaintiff must prove *causation*.

To prove causation means that the plaintiff must show the defendant's conduct is a cause of the injury. Two main theories of causation in negligence are proximate cause and substantial factor to the cause.

Proximate cause is closely linked with foreseeability. Proximate cause, synonymous with "legal cause," is used to describe one event as the direct, natural, or probable result of another.[42:254] To say that event *A* proximately causes event *B* is to say that *B* is the natural result of the happening of *A*. Foreseeability and proximate cause are often used interchangeably, because events which are foreseeable results of *A* are often proximately caused by *A*. Thus, the duty of care for a designer is to protect against all proximate causes of injury or damage.

Probably the most famous case in American law history concerning causation in negligence is *Palsgraf v. Long Island Railroad* (1928) because of its uncanny chain of events and the legal scholarship on causation it inspired. In this case two men attempted to board a train that was beginning to move from the station. Two guards on the train who worked for the railroad helped the two men onto the train, but one of the guards bumped a nondescript package from under the arm of one of the men. The package, which contained fireworks, fell to the tracks and exploded. The explosion loosed some scales which then fell on an unassuming bystander, Helen Palsgraf, who was standing on the platform waiting for another train. Palsgraf sued the railroad for negligence on the part of the guard who bumped the package from the arm of the man. The case went all the way

to the New York Court of Appeals, the highest court in that state. The central question was whether the railroad was liable in negligence to Mrs. Palsgraf.

Two positions dominated for absolving liability from the railroad in the subsequent legal scholarship that followed.[68:98] The first is that the guard may have been negligent, but that his negligence is not a proximate cause of Mrs. Palsgraf's injuries since too many factors intervened between the initial event and the subsequent injury. The injury to her was not reasonably foreseeable by the guard. The second rationale for denying negligence is simply that the guard owes no duty of care to protect Mrs. Palsgraf from exploding fireworks. Her injury by such a cause is unforeseeable. Because the guard owes no duty, the fact that the explosion causes her injuries in some sense is irrelevant.

This second rationale was adopted as the majority opinion by the New York Court of Appeals. The significance of this ruling is that the backward trace of proximate causation is not forever. The law decides to follow a series of events only to a certain point and no further.[68:98]

Palsgraf is relevant to agents because often the number of intervening factors in agent malfunctions may be long and complicated. Adaptivity and mobility multiply the causal complexity and further displace liability from the designer. If a number of intervening factors occur between the inception of the error or danger and the eventual disaster then the software designer may not be held liable. The backward chain of causation from the injury to the programming error or design flaw which causes the injury may be so complex and long that other negligent actors may intervene and absolve the computer professional from culpability. For example, an adaptive mobile agent which contains a design error but is used for a long time on many different servers without mishap, encountering multitudes of other agents and users, and modifying itself multiple ways, may not incur liability for its designer when the design error is uncovered after an accident. Too many intervening factors may occur in order to prove that the accident is proximately caused by the design flaw.

Rather than prove that the defendant's negligence proximately caused his injuries, a plaintiff may be able to show that the defendant's conduct is a *substantial factor* in the cause of the injury, as was done in *In re Manguno* (1992). Section 435 of the Second Restatement of Torts says if the actor's conduct is a substantial factor in bringing about harm to another, the fact that the actor neither foresaw nor should have foreseen the extent of the harm or the manner in which it occurred does not prevent him from being liable.[42:267] Thus foreseeability is not a consideration in substantial factor causation as it is in proximate causation. The substantial factor approach has been used even if one of the factors is not a suable person, as in *Dale v. Baltimore & Ohio Railroad Co.* (1986). The plaintiff recovered under negligence for exposure to asbestos. It was argued that the plaintiff's asthma was a substantial factor to the cause of the injury; the defendant was required to compensate in full. The defendant's defense of contributory negligence was denied since the plaintiff could not be held negligent for self-exposure to asbestos, the presence of which he was reasonably unaware.

Dale is significant for agent liability because agents are also not suable persons. The precedent set in *Dale* could be used to argue that agents, as a substantial factor in the cause of damage, implicate their designer. Since an agent may be a substantial factor in the cause of some injury, but the agent itself is not a suable person, the defendant would likely be the agent's designer.

Under both proximate cause and substantial factor, the designer is likely to be held liable for injury if the agent causes damage. The presence of intervening elements may complicate the chain of causation and absolve the designer. Nevertheless, these intervening elements cannot be simply attributed to the computer or agent, they must be negligent acts on the part of people. For this reason, agent developers must take every precaution to foresee all possible causes of injury and engineer against them. The simple fact is that if a damage occurs with agent software, the developer of the software is the most likely person to be held negligent.[52:21]

4.2 Strict Liability in Tort and Warranty

Besides negligence, two main legal actions are available to those injured by a product. These other theories of recovery are strict liability in tort and warranty. Enough aspects of these two theories are similar that they can be coupled together in this section. Both theories apply to products and not services, unlike negligence which refers to the conduct of a person (failing to meet a standard of care) rather than the article itself. Likewise both strict liability and warranty have similar definitions of defectiveness, a requirement for recovery in both theories. Negligence and strict liability are theories in tort, whereas recovery under warranty is not in tort but in contract.

At this point it probably does not surprise the reader that autonomous software agents make legal theories more complicated, and strict liability and warranty are no exception. The two main complications are in defining agents as a product and determining defectiveness. Agents are more prone to causing damage than conventional software, and strict liability and warranty may be used often in order to recover for damages.

An Overview of Strict Liability

Strict liability was the third legal doctrine to emerge after negligence and warranty. Before the 1930s, negligence was the main theory under which torts were tried. Negligence required proving fault on the part of the defendant, and little hope of recovery existed if one could not prove fault. However, a few scattered cases did not fit well into the negligence mold because recovery seemed warranted despite the lack of provable fault. These cases were called "peculiar liability" cases and were usually injuries from wild animals or explosives.[68:108] During and after the 1930s, however, legal theorists

began considering liability without fault in more cases than just injuries from wild animals and explosives. The doctrine of strict liability emerged, which literally means "liability without fault."[22:129] It was slowly but consistently adopted, not because of its philosophical merit but as a matter of utilitarian social policy. Fowler Harper, author of an influential treatise on torts in 1933, referred to strict liability as "a pure matter of social engineering."[68:109] In the early 1940s, William Prosser, probably the most influential American tort theorist of all time, attacked "fault" as a meaningful concept and predicted the beginnings of a new trend in liability. Prosser's trend was realized a few years later as the great explosion in product liability cases tried under strict liability theory came after WWII.[68:110] In 1960, Prosser wrote an influential paper on strict liability that further established it as a doctrine of recovery.[44] Negligence principles had lost their monopoly on torts.

A landmark case in the history of the strict liability doctrine was *Escola v. Coca Cola Bottling Co.* (1944). Because strict liability had not been widely defined as a tort theory, this case was tried under negligence. Justice Roger Traynor wrote the opinion for this case and helped to assemble already emerging ideas into a solidified doctrine of strict liability for defective products.

In *Escola*, a waitress was injured by an exploding soda bottle when carrying it from its case to the refrigerator. In trial the plaintiff was allowed *res ipsa loquitur*, literally "the thing speaks for itself." This doctrine is invoked when the damaging product is within the exclusive control of the defendant, the accident does not ordinarily occur in the absence of negligence, and the plaintiff does not contribute to the cause of her own injuries.[26:230] In essence, *res ipsa loquitur* is viable when there is no conceivable explanation for the injury other than the defendant's negligence. The use of *res ipsa loquitur* by the plaintiff forced the defendant to prove it was not negligent with respect to the harm. In fact, Coca-Cola was able to show that it carefully inspects each bottle for defects and measures the internal pressure. This suggested that the defendant had in fact

not been negligent. Still, an injury occurred, and the jury found for the plaintiff. The case went to the Supreme Court which ruled that the plaintiff was entitled to recover from the defendant for her injuries.

Much of Justice Traynor's opinion in *Escola* is based on the scholarship of Prosser from a few years before. Whereas Prosser had argued convincingly for strict liability from an academic standpoint, Traynor "gave it a model for practical application."[68:200]

William Prosser authored §402A of the Second Restatement of Torts. This section provides for strict products liability for physical harm caused by the sale of a product in defective condition which poses an unreasonable danger to the user or consumer.

> (1) One who sells any product in a defective condition unreasonably dangerous to the user or consumer or to his property is subject to liability for physical harm thereby caused to the ultimate user or consumer, or to his property if
>
> (a) the seller is engaged in the business of selling such a product, and
>
> (b) it is expected to and does reach the user or consumer without substantial change in the condition in which it is sold.
>
> (2) The rule stated in Subsection (1) applies although
>
> (a) the seller has exercised all possible care in the preparation and sale of his product, and
>
> (b) the user or consumer has not bought the product from or entered into any contractual relation with the seller.

Figure 4.1. Excerpt from §402A of the Second Restatement of Torts.

The revolutionary section (2)(a) meant that producers of a product can be held strictly liable even if they do nothing wrong, per se, but an injury nonetheless occurs. Even if the product is thoroughly tested and re-tested, and all possible precautions are taken, the product's producer, designer, or vendor can be held strictly liable for all injuries which occur from the use or presence of their product.

Strict liability is especially important for software because it is so extremely complex and thorough testing so difficult. If software producers are held strictly liable, they could be sued for errors or malfunctions in software even when they had no way of preventing them. Software has a tendency to hide disastrous bugs until the moment when

the potential for harm is high, at which point the bugs fly forth, sometimes with disastrous consequences.

At first blush strict liability seems highly unfair. After all, should not people only be held responsible for the actions they take and the mistakes they make? Strict liability must be viewed from a utilitarian perspective in order for its worth to be appreciated.

> The arguments generally given for strict liability are consequentialist in character. Strict liability has good effects. By threatening to make a person pay damages when a product is faulty, it encourages the person to take precautions before releasing the product. It encourages producers to be careful about what they put into the stream of commerce. So although strict liability is, in a sense, blind to what the manufacturer has done to make a product safe and reliable, it is imposed with an eye to pressuring manufacturers to make their products as safe and reliable as possible.[22:136]

Whereas negligence is deontological in nature, strict liability is a tool of law, used to produce consequences beneficial to the marketplace and society. Strict liability is a necessary mechanism for people to recover from injuries caused by products regardless of the conduct of the product's producer. Whether or not the producer acts carelessly has no bearing on the truth of whether or not an injury occurs.

Strict liability in tort is better understood when compared to negligence. Other aspects besides philosophical underpinnings differentiate negligence from strict liability. In negligence, the danger and use of the product must be foreseeable, whereas in strict liability, only foreseeability of use is required. If a product is being used in a foreseeable manner and an injury occurs, then strict liability will likely apply, regardless of the foreseeability of the risk of harm. The basic substantive difference between strict liability in tort and negligence is that strict liability relieves the plaintiff of the burden of proving

that the defendant's conduct is below a standard of care.[26:75] Though this is the substantive difference between strict liability and negligence, in practice the line between strict liability and negligence in defect cases is thin. Especially in design and warning defects, where the fault is spread throughout an entire line of products, injuries usually occur less randomly and more because of a negligent design choice. Production and manufacturing defects separate strict liability and negligence more clearly because these defects are often random and occupy a small percentage of the production line. Production defects represent quirks in the production process rather than an error infesting an entire line of products like a design defect. Because of reliable reproduction techniques, software is more susceptible to design and warning defects than production or manufacturing flaws, though the latter do occur in diskettes and other physical media.

IMPLIED AND EXPRESS WARRANTIES

Warranty as a means to recovery came into existence as a doctrine of law after negligence but before strict liability. Warranty has its origins in tort theory, but is now so identified with contract that consideration of the non-contractual aspects is purely academic.[26:21] Warranties exist effectively as a contract between the seller of goods and the consumer. Warranty therefore is governed by sales law rather than tort. Warranties are an assurance by the vendor that his product is fit for sale and performs as intended. In the past, warranty law included the *privity* requirement, though now this requirement has been eroded. This requirement meant that only certain persons are able to recover from a vendor under warranty, and third party "bystanders" not in privity with the vendor are not eligible to recover, though they may be able to take action under negligence or strict liability in tort.

Warranty is a strict liability, but it is a strict warranty, not a tort liability. (A cause of action in warranty is not a tort.) If it can be shown that a warranty exists between the

consumer and the vendor, that the warranty is breached, and that damage results, the plaintiff is entitled to recover regardless of whether or not the plaintiff shows the vendor failed to exercise reasonable care.[26:30] Most claims in warranty are only actionable for personal damage, not damage merely to property. Unlike negligence and strict liability which are tort theories and described in the Second Restatement treatise, warranties are covered in the Uniform Commercial Code (UCC) in Article 2. Like strict liability in tort, warranty is applicable only to products, not to services. As far as agents are concerned, if software (in general) or agent software (in particular) are shown to be services instead of products, UCC §2 will not apply.

Warranties are in effect for every commercial transaction at the time of sale in all states except Louisiana, which failed to adopt the UCC. Two types of warranties exist: express and implied.

Implied warranties are not made explicit but are automatically in effect for all transactions under the UCC. Implied warranties "protect the customer by ensuring that he or she does not have to negotiate certain conditions of a purchase, they are always a part of the purchase agreement (even when the agreement is not in writing)."[22:131] The two implied warranties are the warranty of merchantability and the warranty of fitness for a particular purpose. Express warranties, on the other hand, *are* made explicit as part of the contract of sale between the vendor and consumer. Express warranties are varied and address all types of contractual obligations.

Section 2-313 of the UCC provides for *express warranty*. Express warranties can be created in three main ways: fact or promises made by the vendor, descriptions of the product, models or samples to which the real product is supposed to conform.[26:22-3] Sales "puffing" and marketing "hype" are not considered express warranties. This is important for marketed agents which are sold with impressive claims about intelligence and autonomy. Express warranties need not be written, as "[t]hey can arise by virtue of demonstrations, oral representations by sales persons, or statements contained in

advertising and promotional literature."[62:108] Distinguishing a statement of fact that creates an express warranty from what is merely an opinion offered by the vendor is not always easy. This is a question for the jury.

Express warranties, as well as other contracts, must be *conscionable*. The UCC §2-302 provides that if the court finds a contract or disclaimer unconscionable it may refuse to enforce it. In *Henningsen v. Bloomfield Motors, Inc.* (1960), the court refused to enforce a contract which it found unconscionable. The contract tried to exclude remedies for personal injuries and was a standardized form used by the entire automobile industry. The court found such a ubiquitous contract gave the buyer no choice but to accept the terms. The court ruled to dissolve it.

Section 2-314 of the UCC defines the *warranty of merchantability*, the first of the two implied warranties defined in the UCC. The warranty of merchantability says that the goods sold must be "merchantable," that is, fit for sale. In order for this implied warranty to be in effect, the seller of the goods must be a "merchant" with respect to those goods: One who regularly deals in goods of the kind and represents himself as one who is knowledgeable about such goods. This excludes the occasional seller from the implied warranty, such as the individual who sells his used car or holds a garage sale.[26:24] In order for goods to be considered fit for sale, they must be of sound quality and usable for the general purposes of such goods. (See UCC §2-314(2) for the full definition of a "merchantable" good.) A complication with agents is that the general purpose of an agent is not well-defined. The number of varying definitions of an agent and agent capabilities is evidence of this. A merchantable agent could mean many different things.

Section 2-315 of the UCC frames the *warranty of fitness for an intended purpose*. This implied warranty guarantees that any good sold for a particular purpose in mind will be able to perform for such a purpose. The buyer of the good must show that the seller knew, or had reason to know, of any intended purpose for which the goods were to be used.[26:26] "Cases employing the principle of intended purpose often rest on evidence that

the vendor knew what the buyer planned to use the system for and yet sold the buyer a system that was not suited for that purpose."[22:131] Unlike the warranty of merchantability, the consumer taking action under the warranty of fitness does not have to show that the vendor is a merchant with respect to the goods. However, unlike in merchantability, under the warranty of fitness the buyer must show he relied on the seller's special skill in order to show a breach of warranty occurred.

One case involving computers and the warranty of fitness for an intended purpose was *Public Utilities Commission for City of Waterloo v. Burroughs Machines, Ltd.* (1973). In this case the plaintiff purchased a computer system consisting of hardware and software from the defendant. When the plaintiff received the product, however, the hardware was supplied but the software was not, rendering the hardware useless. The court found that Burroughs had breached warranty for supplying hardware that was unfit for its intended purpose. Burroughs was required to supply the software as part of the sale.

As an example of the distinction between the two implied warranties, consider a consumer who buys a printer for her computer. The presence of the printer on the merchant's shelf is a representation that it is reasonably fit for its general purpose, namely that it can print text on paper. However, if the buyer is in the market for a special printer that is able to print on overhead transparency film, and relies on the expertise of the seller to purchase a printer for such a purpose, the vendor has created the implied warranty of fitness for a particular purpose. If the printer fails to print adequately on transparency film, the warranty is breached.

Software agents are not ubiquitous nor concretely defined. This makes defining the warranty of merchantability difficult. The requirement in merchantability that a product be "fit for the ordinary purposes for which such goods are used" is uncertain because agents are highly specialized and not widely commercially available (though this will likely be false by the year 2000). What are the "ordinary purposes" for autonomous

agents? Like all software, agents should not crash repeatedly or cause damage to systems. They should be usable on contemporary computer systems and networks. But being more specific is difficult with such a new and broadly defined technology.

When agents are adaptive defining their "ordinary purpose" becomes even fuzzier. The capability for agents to be adaptive means that there may not be any certain way to define what their ordinary purpose is. Adaptive agents may be seen as "custom designed" since each is distinct. In this case, "the implied warranty of merchantability would appear to have little application in situations involving custom-designed … software which, by definition, has no 'ordinary purpose.'"[62:109] Each agent will behave different for each user.

Complications with Agents

Not only do agents complicate foreseeability in negligence, they also do not fit nicely under the law of strict liability in tort or warranty. Strict liability and warranty only apply to products, not services. Debate continues over how agents should be treated. But the types of damage that agents can cause may result in the pragmatic need to impose strict liability and warranty, even if the development of agents suggests their treatment as services. Furthermore, the application of strict liability is only to physical damages. An agent that deletes a file system, for example, destroys information, but not anything physically tangible. So imposing strict liability on agent-related damages may not be legally feasible.

These complications are the topics of the next two sections. The first considers whether or not agents should be considered products or services. The second explores defectiveness and damage. Both sections raise questions with direct significance for agents under strict liability in tort and warranty.

4.3 Products, Services, Software and Agents

As mentioned above, strict liability in tort and warranty applies to suppliers of products but not services. A computer repairman cannot be held strictly liable if his work on a computer system causes it to smoke and sputter. (In such a case, he might be found negligent for failing to meet the standard of care incumbent upon the reasonable person in his occupation.) Neither can he be found guilty of breaching the implied warranty of merchantability (he is not a merchant) nor the implied warranty of fitness for an intended purpose (no product is under consideration for fitness). He could be found guilty of breaching an express warranty, but this would be a service warranty, not an express warranty of the form governed by Article 2 of the UCC.

The treatment of software as a product is a relatively new idea. In the early 1980s many legal scholars were unsure how to treat software. Should it be subject to strict liability? Warranty? Should it be patentable? or copyrightable? Or should the building of software focus on the service, not the end product? While some of these questions have been answered for specific cases, general maxims about how to treat software are still being formulated. Agents make these questions even harder to answer because of the ways in which they are developed and the malleability of code.

Software as an Intangible

The Second Restatement of Torts §402A limits recovery under strict liability in tort to products cases. Likewise, the UCC §2-102 limits recovery under warranty to products. In fact, the language of the UCC in this section is that it "applies to transactions in goods." Goods are often thought of as chattels—*tangible* wares—and this interpretation was held for a long time, though there has been some extension of strict liability and warranty to intangibles.

Debate exists over whether software should be considered a product for purposes of strict liability in tort and/or warranty. The questions over software sound like this: "Is software a set of mental operations implemented by a computer? Is it a set of electronic impulses? Is it a data set? Is it a process that a computer undergoes?"[22:133] An intangible thing like a program does not seem to be a product in many ways. After all, a program is merely a list of instructions, a product of the mind but not a physical manifestation. Electronic impulses within a computer which are caused by running software are certainly not tangible.

Software may not be a product because of the way it is built. Custom designed software is certainly more like a *service* than a product. "Custom designed programs are now commonly used by small businesses, and a substantial part of the computer industry has become quite service and client-oriented."[62:216] If a company out-sources a programming task, the contract drawn up is for the performance of a service, not for the production of a product. As far as programs are concerned, it seems only the physical materials that store them (*e.g.*, diskettes) are products.

The courts have a tendency to treat products as chattels. However, a few court decisions have extended the definition of products beyond mere chattels.[62:225] In both *Ransome v. Wisconsin Electric Power Co.* (1979) and *Houston Lighting & Power Co. v. Reynolds* (1988), the court ruled that electricity, after it has been delivered to the consumer, constitutes a product and is subject to strict tort and warranty. Products need not be inanimate, either. In *Sease v. Taylor's Pets* (1985), a rabid skunk was sold to a consumer as a pet. The skunk caused injury and the pet shop was found strictly liable for the sale of the rabid "product." This case is important when considering agents because it shows that a product may act and behave autonomously and still incur liability for its vendor. Neither skunks nor autonomous agents can be held liable, a fact which shifts liability to vendors and designers.

In criminal liability (as opposed to civil liability), the court ruled in *Latham v. State* (1975) that a computer program's printout is subject to larceny. A few years later in *National Surety Corp. v. Applied Sys., Inc.* (1982), the court used the decision in *Latham* to rule that a computer program is subject to the offense of conversion, an offense formerly restricted to tangible wares. This ruling is significant for mobile agents that affect "property" on remote computers, changing it or adding to it in some way. *Latham* and *National Surety Corp.* could set a precedent for theft if a mobile agent erroneously retains code it should not have, even though code is an intangible. In another criminal case, *United States v. Seidlitz* (1979), the United States Court of Appeals for the Fourth Circuit ruled that the accused was guilty of unforeseeable theft under Title 18 of the Unites States Code for a computer software package known as the WYLBUR system. The ruling stated that the jury could find sufficient evidence to consider WYLBUR property. Criminal cases also have supported the notion that software is a product and subject to theft and criminal prosecution.

Despite the rulings in both civil and criminal court that suggest software is treated as a product, this analysis is imperfect. If software is to be treated as a product subject to strict liability in tort, then a programmer could be held strictly liable for errors in a program even though he did everything reasonable to test for those errors and could have no reasonable way of knowing about them.[22:134] This standard may be too strict for software engineers, because software is so complex and occasional crashes are tolerated. A civil engineer does not have the same leniency in designing a bridge. For a bridge to "crash" even once is unforgivable. Software has a high failure rate when compared to mechanical, structural, and electrical devices. As mentioned in the last section, warranty law is complicated by treating agent software as a product because standards for acceptable failure rates are not set, thus making a "merchantable" agent an unclear standard. Given the complexity of agents, there may have to be a much more lenient

standard by which they are considered merchantable, even more lenient than conventional software.

PATENT

Another measure of whether an item is a "product" is whether or not it is able to be protected by patent. Items for patent must have utility, novelty, and be non-obvious. If something is not patentable it is likely not a product. Patent protects a product in its constructed form, but not necessarily the idea behind the product. "Patents prevent the appropriation of your implementation of an idea, putting your brainstorm to work either as a specific product … or as a process in accomplishing some end."[49:158]

Both patents and copyrights (covered in the next section) are utilitarian. "The reasoning behind both the patent and copyright systems is consequentialist in that the system's primary aim is to create property rights which will have good effects. Invention is encouraged and facilitated so that new products and processes will be made available."[22:70] Without the patent and copyright systems, people would not be able to devote energy to developing new ideas because they would not be guaranteed protection for those ideas. Though ideally people would have a deontological respect for others' inventions, the number of patent and copyright infringement cases is an indication that without these forms of protection ideas would be stolen.

Patenting software seems a bit awkward because of its intangible nature, though the few cases above have set a precedent for intangibles being considered products for strict liability purposes. Patenting also gives one a legitimate monopoly over an invention, and this tight control may be too extensive. Additionally, some things cannot be patented whose nature is similar to software: ideas, scientific principles, mathematical algorithms, natural phenomena, and mental processes.[16:292] Software seems strikingly similar to ideas and mathematical algorithms. Much of software is patented today,

however, so obviously courts have not found this similarity to be material. One rationale is that the instantiation of a process inside a machine is a patentable process, even if part of the process includes what independently might be regarded as a mathematical formula.

A landmark case in deciding if software is patentable is *Diamond v. Diehr* (1981). In this case the Supreme Court in a five to four vote granted a patent to Diehr for a software program that helped in molding raw, uncured synthetic rubber into cured precision products. The majority opinion written by Justice Rehnquist concludes,

> [W]hen a claim containing a mathematical formula implements or applies that formula in a structure or process which, when considered as a whole, is performing a function which the patent laws were designed to protect (*e.g.*, transforming or reducing an article to a different state or thing), then the claim satisfies the requirements [for patent]. Because we don't view the respondent's claims as an attempt to patent a mathematical formula, but rather to be drawn to an industrial process for the molding of rubber products, we affirm the judgment.

The ruling in *Diehr* suggests that the patentable aspect of a computer program is the unique *process* that it implements. This is not unprecedented as patents are often used to protect chemical processes for the synthesis of new chemicals.[23:295-6] But prior to this ruling there had only been a handful of software-related patents granted. After *Diehr*, patents were routinely granted to software.

COPYRIGHT

Copyright is a form of protection that is appropriate for written materials which are *not* products. Items for copyright must be an original work of authorship and must be fixed in a tangible medium of expression (see 17 United States Code §102(a)). Copyright protects

the exact expression of the idea in letters and numbers (or other symbols), but not the idea expressed. “Intrinsic to copyright law is a distinction between the *ideas* expressed in a manuscript (not protected) and the way in which those ideas are *expressed* (protected).”[23:297] The binding, paper, and cover of books are products and could be patented, but the written information inside them is not a product and is protected by copyright. In order to show a copyright violation, the plaintiff must show the defendant had access to the original and the alleged copy is “substantially similar” to the original.[49:160]

Copyrighting software is problematic. Copyright only protects the “work of authorship,” the expression of the code itself, not the idea. “Copyrights deal solely with the expression of ideas—how you go about communicating your idea to the world.”[49:158] So someone who observed the running of a computer program and built an identical program in a different language would probably not be violating a copyright.[22:62] Even two programs in the same language which perform exactly the same function could vary in their appearance by many factors: the order the data is to be entered, the variable names, the order of the instructions, the statement numbers, the way in which the same instruction can validly be expressed, the comment statements, and the format of the output.[16:287] Contrary to what most people believe, copyrights do not offer protection for your ideas.[49:157-8]

As an example of the same idea expressed two ways, consider a Factorial function in the C programming language below. (The factorial of a number *n* is that number multiplied by every integer less than it down to one. For example, 5! = 5 × 4 × 3 × 2 × 1 = 120.) Factorial takes in a number *n* and returns *n*!:

```
int Factorial(int n) {
      int j, ans;
      if (n <= 0) return(-1);  /* Error condition */
      ans = 1;
      for (j = n; j > 0; j--)
            ans *= j;
```

```
        return(ans);
}
```

Figure 4.2. An iterative procedure in C for computing *n*!.

However, a second Factorial function also in the C programming language achieves exactly the same result but appears textually quite different:

```
int Factorial(int n) {
        if (n <= 0) return(-1);  /* Condition for error */
        if (n == 1) return(1);
        return(n * Factorial(n-1));
}
```

Figure 4.3. A recursive procedure in C for computing *n*!.

If copyright is strictly limited to the exact form of the expression of programs, then a person who authors the first program has no protection from someone who uses the first as a source of idea for the second. Not only is the code different in the second, but the error comment is slightly different as well. (This example is quaint but illustrates the issue. In reality, the software would be a much bigger program with many parts.)

In 1983, *Apple v. Franklin* (1983) went to the Supreme Court as had *Diehr* two years prior. In this case, Apple Computer Corp. sued Franklin for copying fourteen programs constituting their operating system. In fact, Apple was able to systematically go through its own code and Franklin's code and show line by line that it was identical. Franklin did not deny copying Apple's code, but argued that object code (1s and 0s) is not copyrightable. In 1980, Congress had amended the 1976 Copyright Act to explicitly include protection for computer programs,[22:62] but the question in *Apple* was if this protection was only for application programs or operating systems in object code as well. The US Court of Appeals for the Third Circuit wrote,

> [There has been] uncertainty as to whether a computer program in object code could be classified as a 'literary work.' However, the category of 'literary works,' one of the seven copyrightable categories, is not confined to literature in the nature of Hemingway's *For Whom the Bell Tolls*. The definition of 'literary works' … includes expression not only in words but also 'number or other … numerical symbols or indicia,' thereby expanding the common usage of 'literary works'… Thus a computer program, whether in object code or source code, is a 'literary work' and is protected from unauthorized copying, whether from its object or source code version.

The injunction against Franklin prevented them from using their copied code. Eventually the case settled before reaching the Supreme Court. Apple retained its competitive advantage.

Books are widely protected by copyright and may provide a model for software protection. However, two landmark cases showed that neither strict liability nor implied warranty is applicable to books. In *Winter v. G.P. Putnam's Sons* (1993), the court ruled that information contained in books is not subject to strict liability in tort. Winter sued Putnam for the cost of a liver transplant that was necessary after Winter digested a mushroom erroneously depicted as being safe in an encyclopedia published by Putnam. "The judges in the *Winter* case decided that the strict liability in tort doctrine should only apply to the manufacturer of tangible 'products,' such as tires and insecticides, for which the doctrine had been created."[52:23] The courts have been reluctant to impose liability of this kind for fear of chilling freedom of expression.[18:395]

In *Cardozo v. True* (1977), the plaintiff became violently ill after she ate a piece of raw plant while preparing to cook it for use in a recipe written by the defendant. Cardozo claimed that the recipe bookseller had breached the implied warranty of merchantability by failing to warn her the plant was poisonous if eaten raw. The court

ruled against the plaintiff, finding that, while the implied warranty of merchantability applied to the book itself (paper, binding, cover, etc.), it did not apply to the information inside. Therefore, following the precedents set in *Winter* and *Cardozo*, if software is a copyrighted material similar to books then it should not be subject to strict liability or implied warranty.

In order to make copyright protection more applicable to software, the courts have allowed the "look and feel" of a program to be copyrighted. The rationale is similar to the copyright on a film or television program: "The images that make up a slide show, motion picture, or television program are copyrightable expressions. The same protection has been extended to the video displays generated by computer programs."[49:160] A notable case involving look and feel copyright infringement was *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.* (1987). Whelan developed a program for Jaslow for use in a dental office, and Whelan owned the rights to the product. A few years later, Jaslow decided the program could be built in another language for PC use. Jaslow developed the similar program, and Whelan contended infringement. The court agreed with the plaintiff finding that the look and feel of Jaslow's program infringed on Whelan's copyright. In some cases copyright has been extended beyond mere line-by-line similarity.

Confusion exists over how software should be treated: as a product or written material. Patents are intended for functional items, copyrights for non-functional expressions. Patents do not protect mathematical algorithms, to which software is similar, but software is also functional, and so copyright is not appropriate, either.[49:158] "Neither copyright nor patent protect what appears to be most valuable in software—the algorithm, the structure, sequence and organization, and the 'look and feel.'"[22:69] The software industry has resolved this debate—if not in principle, at least practically—by simply not pursuing copyright protection very often. Copyright does not afford the tight protection software professionals desire. Most of the modern software industry acquires patents for its products, but in this process another issue is raised: The volatile question is

not whether software should be patented or copyrighted, but whether software is a good or a service.

Software as a Product or Service

More pertinent than the patent and copyright debate is the debate over whether software should be treated as a product or *service*. If it can be treated as a product, then strict liability in tort and warranty can be applied. If it is treated as a service, then only negligence will apply by addressing the conduct of the service-provider. Today strict liability in tort and warranty *does* apply to much off-the-shelf (OTS) software, but not to custom-designed software. Custom designed software is regarded as a service and OTS software a product.

In order to understand why strict liability applies only to products and not services it is necessary to know the reasoning behind the theory. Three main justifications for strict liability have been widely cited in legal scholarship.[43:850,44:1122-4] The first of these is the *stream of commerce* rationale. The rationale says that the producer of the product placed it in the stream of commerce to earn a profit, and because of this potential profit, he should compensate injury caused by his product. Implicit in the producer's placement of the product on the market is his implicit assurance that it is safe for consumer use.

The second reason behind the theory of strict liability is the *position to control risks* rationale. The reasoning here is that the producer of a product is in the best (and often only) position to control the risks of using the product. The producer has the only intimate knowledge of the product and access to the design and manufacturing processes. The producer is intimately involved with the product for a long time, and if anyone is able to design out, guard against, or warn about the dangers of a product it is the producer. Therefore, if something should go wrong with the product, he should be liable.

Finally, the third basis for imposing strict liability is the *cost-spreading* rationale.[44:1120] The producer of the product is aware of the risk of liability and can factor it into the cost of the product, thus spreading the cost of liability across society. No one else is in a position to spread such costs, and so the producer should be the one liable for the damages since he can soften the blow by incorporating liability costs into the product's price.

Some authors[41:16] include a fourth justification for strict liability: It greatly eases the litigation burden of an injured consumer who may be powerless to prove negligence because of the greater burden of proof. Recall in a negligence suit, the plaintiff must prove that a duty of care exists, that a breach of this duty occurs, the actions of the defendant cause the injury, and damage results. The main requirement in a strict liability suit is proving an injury occurs as a result of the foreseeable use of the product.

Notice that the justifications for imposing strict liability on producers of products do not transfer to the suppliers of services. Service providers do not place a product in the stream of commerce. They are not in a position to control the risks of using a product. They cannot spread the costs of their service across a large consumer population. If a service provider makes a damaging mistake, it is likely they owed a duty of care to the recipient of their service and will be found negligent. In *LaRossa v. Scientific Design Co.* (1968), the court refused to impose strict liability on an engineering company that designed and engineered a power plant containing vanadium, a carcinogen that killed a worker, because the company was performing a service and not widely distributing a product across consumers. Strict liability as a utilitarian doctrine of tort makes no sense for service providers.

Implied warranties can only be applied to products as well, and the rationale is simple: Implied warranties ensure consumers receive benign products which are adequate for their purpose. It does not make any sense to speak of "merchantability" or "fitness for

intended purpose" with respect to a service. Thus, like strict liability, warranty also makes sense only for products.

Why strict liability and warranty only apply to products and not services is clear. However, the question still remains: Should software be treated as a product or service? This is an important question for agents because if conventional software is treated as a service, then agents (as software) will also be treated as a service. This means agents will not be subject to strict liability or warranty, only negligence. If software is to be treated as a product, and this includes agents, then they *are* subject to strict liability and warranty. Imposing these legal doctrines is prudent given the potential of agents to cause damage.

Jim Prince, writing for the *Oklahoma Law Review*, proposed an analogy between software and men's suits. This analogy serves as a useful model when thinking about software as a product or service.[43:852] Prince argues that buying a suit off the rack is like buying pre-packaged OTS software and should be treated as a product. Suits off the rack and OTS software alike are spread across the consumer spectrum and the producer can incorporate the costs of liability into the product. Continuing with his analogy, Prince likens a tailored suit to custom-designed software. Just as a businessman might employ a tailor to fit a suit, a company might hire an independent programmer to custom-build software for their particular needs. Software like this is built as a service, and is not subject to strict liability nor warranty. The only question left, then, is what if software is a mixture of OTS and custom? Prince argues that this case is like the suit that is on the rack but then altered for the consumer. If an error occurs in the mass-produced OTS portion of the software, then it should be subject to strict liability in tort. If the damage is a result of the customizing, then it should be subject to negligence and not strict liability.

Prince's analogy is helpful but it may not be easy to apply in the third case. Often OTS software is modified so greatly that the OTS parts and customized parts are inextricably bound. Many people work on the software and contribute to it in a variety of ways, and by the end, the original OTS product is covered by layers of "programming

sediment." Under this analysis it still is unclear whether agents should be considered products or services.

Are Agents Products or Services?

According to Prince's analysis, if an agent is sold off the shelf as a widely distributed product, it should be subject to strict liability and warranty. If it is custom designed, liability should focus on the designer's conduct under negligence principles. And if it is a mixture, then the courts should consider from what part of the agent the error results.

This analysis is somewhat difficult to apply when software has become greatly modified by many people. With agents it is even more difficult to apply, because *agents may modify themselves*. Should the modification be treated as part of its OTS code or custom code? Agents modify themselves in response to their users, so in some ways they are custom designed by those they serve! Brenda Laurel makes the point that "[a]n agent that is responsive to the goals, needs, or preferences of a person—and especially an agent that can 'learn' to adapt its behaviors and traits to the person and the unfolding action—could be said to be 'codesigned' by the person and the system."[28:148] If Laurel's point is taken seriously in light of the law, then an adaptive agent should be treated as more of a service than a product.

The fact that adaptive agents can be seen as codesigned by their users makes applying patent and copyright a little tricky. If an agent is protected (by either method) upon its creation, but changes significantly over time, is the new state of the agent protected? If copyright is strictly applied then the agent's code may have changed and will not be the same text that is protected. This difficulty in applying patent and copyright is only in addition to the difficulties for software discussed in the previous section. Plainly, protecting software is hard enough—agents are more difficult.

Another reason to regard agents more as a service than a product is because of their current commercial status. Most agents are not widely produced as OTS software products. Many are still developed in research laboratories or on experimental networks. If a business wants to incorporate agent technology into its company's software, they will often hire someone, enacting a contract for services.

> Because implementation of the high-level tasks performed by commercial AI software requires extensive immersion in the field of application, initial development contracts call most clearly for treatment as service arrangements. If the finished program proves suitable for an entire class of users, however, subsequent sales might appear to involve a product. Ambiguity is inevitable where the uncertain intrinsic character of software diverts attention to its mode of supply for purposes of tort liability.[13:112]

Agents are likely to be first developed as part of a service arrangement, then possibly later become a product. If not in the mass-produced phase of development, certainly in their one-off phase agents must be considered under the law of services.

Agents may need to be monitored during their "lifetimes" because of their complexity and ability to self-customize. This means that both their initial development and their continuing functionality could entail the provision of a service. Much like a soda vending machine needs to be periodically filled, agents may need to be routinely checked by their developers to ensure they have not mutated in some erroneous way. Mobile agents could schedule "check-ups" and transport themselves to the appropriate server to be inspected. Therefore, the responsibility of the agent developer may not be finished after the agent is produced.

Recall from Chapter Two that the massive AI agent architectures like Soar and Act attempt to replicate general cognition, whereas softbot agents only address problems

in a narrow domain with much less general intelligence. The former have been called "deep systems" and the latter "shallow systems." One author argues that deep systems, while more complex, will require less maintenance than the shallow systems. "[D]eep expert systems can generally be expected to deliver acceptable results over the entire useful range of the underlying causal model. Courts will undoubtedly expect greater vigilance from developers of shallow systems … simply as a consequence of the diminished reliability implied by program design."[13:113] The distribution of agents may be seen as a contract for service because they could require occasional monitoring.

Strict liability and warranty are utilitarian. They guarantee that developers of products must take all precautions (and then some) before releasing a product. Strict liability helps to protect consumers while warranty guarantees the efficacy of the products they buy. If agents are not treated as products, the utilitarian benefits of strict liability and warranty will be lost. Negligence will still serve to protect consumers from the careless performance of services but is harder to prove in court. Furthermore, if agents are not products then they probably cannot be protected by patent, and this may discourage developers from further innovation.

Neither side of the product/service question about agents has the clear edge. The adaptive nature of, development of, and monitoring of agents suggests they should be treated as services. Agents are potentially difficult to protect with a patent or copyright. However, in light of the possibility of future mass-distribution and the need for high design standards, imposing strict liability in tort and warranty is crucial. It remains a question for the courts as well as the computer industry. If agents are widely developed as mass-produced products then they will be treated as such.

4.4 DEFECTS

Products liability addresses defective products. "The notion of the *defective* product is the common bond uniting the theories of warranty, negligence, and strict liability."[26:147] If products are never defective, no reason exists to suspect negligent behavior on the part of the developer. Neither does motivation exist to impose strict liability on developers of products that injure. Similarly, warranties make no sense if defects never occur. Defects in adaptive and mobile agents may be hard to detect until damage occurs because of their complexity and autonomous operation. Even the anthropomorphic feature of some agents can constitute a defect under either negligence or strict liability.

Types of Defects

Three main types of defects are manufacturing or production flaws, design defects, and defective warnings or instructions.[42:4] Some scholars include misrepresentations as a fourth category. Warning defects and misrepresentations involve products which have no physical flaws. Other causes of action for products without physical flaws do occur on occasion such as in *Moning v. Alfono* (1977), where a weapons vendor was found negligent for supplying a slingshot to a minor despite the lack of defect in the product itself. But instances like this are less common in products liability, and the primary three cover most of the claims that arise.

The three types of defects are paralleled by three courses of action an engineer takes to make a product safe. In order to remove a source of danger from a product, an engineer should first attempt to "design it out." In the case of agents, designing out a hazard may mean removing a mobile agent's ability to delete files from remote hosts. Designing out sources of damage often limit or inhibit the product. However, if a product can retain its usefulness despite the modification, designing out a hazard is an engineer's first priority.

If an engineer cannot design out a source of danger without destroying the inherent quality or utility of the product, she should next attempt to guard against it. A guard is anything that allows a danger to exist but only within certain limits. A guard could be placed within a mobile agent that allows it to delete remote files, but only after a copy of the file is first transmitted back to its host for safe keeping. Another guard might be the ability to read the file before deleting it and retaining the valuable information. In many computer applications the UNDO command is a type of guard in that it prevents a user from getting into an unwanted and irrevocable state. Agents can be equipped with a form of UNDO as well. Design solutions will be the subject of Chapter Five.

The warning solution is crucial when both designing out and guarding against a danger are unfeasible. The duty of a designer or seller of a product to warn is set forth in §388 of the Second Restatement of Torts, as well as in the common law by cases such as *Stief v. J. A. Sexauer Mfg. Co.* (1967). Warnings are often written labels with catch phrases like "Danger," "Warning," or "Caution" in red or yellow. However, software is less easily equipped with a decal-style warning label. Fortunately, warnings can be placed inside the program as part of its code. Such warnings appear when using a web browser, such as the following warning from Netscape Communicator:

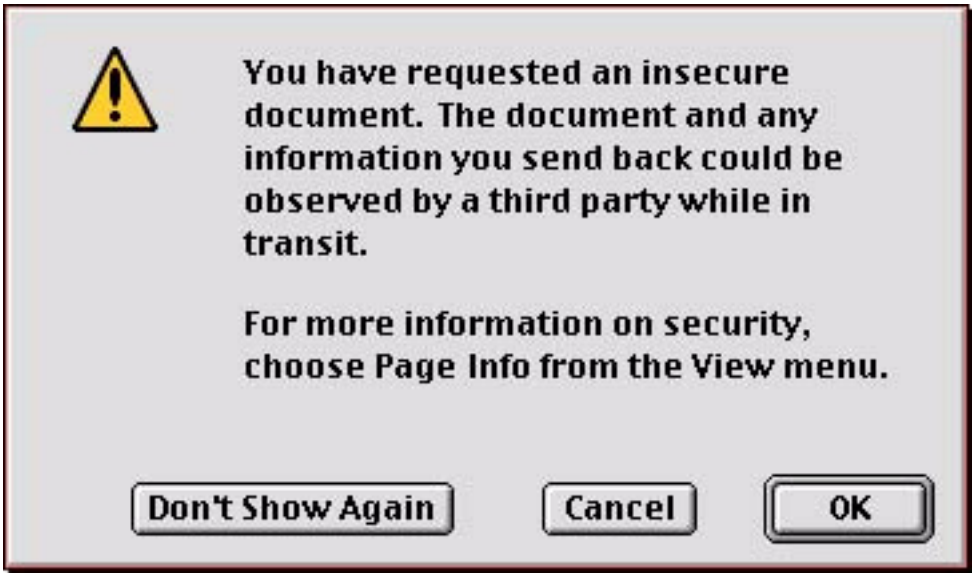

Figure 4.4. A warning message from Netscape Communicator.

Agents could incorporate similar warnings when about to take actions that could result in damages or foreseeable problems. Mobile agents could send warnings back to their home site for the user to acknowledge. The problem with user-required authorization is that it necessitates the user's presence, forcing the agent to wait without it, and thereby defying the power of delegation as an operating paradigm.

Some legal theorists treat warning defects as a type of design defect.[42:5] Both of these defects occur throughout the entire line of a product, and the line between them can be a blurry one. For example, in *Pike v. Frank G. Hough Co.* (1970), a paydozer was negligently designed because it lacked adequate warning lights. Amending the defect would require a redesign of the product, but the defect was one of inadequate warning. Production and manufacturing flaws, on the other hand, occur randomly in specific products and are treated differently than design or warning defects. Cases from production and manufacturing flaws are often tried under strict liability, whereas design and warning defect cases often result in negligence.

DEFECTS IN NEGLIGENCE

Products cases which are tried under negligence are often design or warning defect cases. These defects are the result of decisions made by product developers and are present in the entire line of a product. If an injury occurs as a result of a design or warning defect, the designer's conduct is called into question. If the designer acted reasonably and met the standard of care for his occupation, then he will not be found negligent.

In a negligence case, the plaintiff must show the presence of a defect in the product. Specifically, the plaintiff most show three things: the product is defective, that the defect proximately causes the injury, and the defect existed when the product left the

defendant's control. He does not have to prove wherein the defect resides, only that one is present.[42:238] In a negligence case, the arguments focus on the designer's conduct rather than the product itself. However, this distinction is only semantic, because the defective design of the product is used to show the designer acted negligently.

The most common test for defectiveness in a design or warning case is the *risk-benefit* test, sometimes called the risk-utility test. It essentially asks whether the cost of making a safer product is less than or greater than the risk of danger from the product in its present form. If the cost of making the product safer is greater than the risk of using the product as is, then the benefit outweighs the risk and the product is not defective. If the cost of changing the product is less than the risk of use, then the benefit of not making the change is outweighed by the risk and the product is defective.

Dean John Wade proposed seven factors in *Roach v. Kononen* (1974) to determine the risk-benefit of a product: (*i*) the usefulness and desirability of the product; (*ii*) the likelihood and probable seriousness of injury from the product; (*iii*) the availability of a substitute product that would meet the same need and not be as unsafe; (*iv*) the manufacturer's ability to eliminate the danger without impairing usefulness or making the product too expensive; (*v*) the user's ability to avoid the danger; (*vi*) the user's anticipated awareness of the danger; and (*vii*) the feasibility on the part of the manufacturer of spreading the risk of loss by pricing or insurance. This approach has been widely used in products cases, especially those concerning negligence.

Often expert testimony will be required to substantiate the existence of a defect, especially in design and warning cases. This is almost always the case when the product is highly technical or complex. Agents certainly would fit into this category. Experts from cognitive science, artificial intelligence, computer science, software engineering, or human-computer interaction can testify as to the presence and nature of a defect in an autonomous software agent. Both sides in the case may call experts who will disagree on their findings. One such case was *Krause v. Sud-Aviation* (1969). In this case experts

disagreed on the quality of a weld performed on the tail of a helicopter, one testifying that the weld lacked root penetration, while two others found nothing wrong. A case dealing with autonomous software agents will certainly draw conflicting opinions from experts due to the complexity of the software.

Some products may be deemed so dangerous that no matter how useful they may be, their benefit cannot outweigh their danger. This was the ruling in *O'Brien v. Muskin Corp.* (1983). The case concerned a four-foot deep vinyl swimming pool which was deemed unreasonably slippery and too dangerous for consumers. The benefits of properly functioning agents are enough that it is unlikely they will be deemed unreasonably dangerous. However, if their utility is not realized and their propensity to cause damage too great, they may eventually be deemed unreasonably dangerous because they fail the risk-benefit test.

The *state of the art* defense is used to claim that the current knowledge about a certain product is insufficient to remove the danger from the product or design a better alternative. "This test requires that the design be viewed in light of what was economically and technologically feasible at the time of manufacture."[26:163] Obviously this defense has implications for autonomous agents because of their position on the brink of technology. A defendant accused of negligence in design could argue that the current state of knowledge in the field did not preclude their design.

Defects in Strict Liability and Warranty

Strict liability in tort and warranty also considers defective products but does not require that the plaintiff show the defect is a result of the conduct of the seller, manufacturer, designer, or producer. This is the critical difference between strict liability and negligence.

Often the defects addressed by strict tort and warranty are production or manufacturing defects, rather than design or warning flaws. In manufacturing and production defect cases, the plaintiff can often prove their case by showing the product does not conform to the manufacturer's specifications. Recall the rationales for imposing strict liability and warranty on producers given by Jim Prince. Production defects tend to be random and are inevitable "although the seller has exercised all possible care in the preparation and sale of his product." (Second Restatement of Torts §402A, comment *k*.) However, at times design cases will be tried under strict liability, and the distinction from negligence is blurred. One case that tried to make the difference between strict liability and negligence for design defect cases was *Dart v. Wiebe Manufacturing, Inc.* (1985). The court reasoned,

> In a negligence case, the inquiry focuses on the reasonableness of the manufacturer's choice of design in light of the knowledge available at the time of manufacture. Under strict liability, however, knowledge of the 'danger in fact' as revealed by the accident and the testimony at trial is imputed to the manufacturer.[42:195]

This reasoning is further evidence that the conduct of the developer is immaterial in a strict liability suit but relevant in a negligence suit. Furthermore, in *Kallio v. Ford Motor Co.* (1986), the court ruled that when alleging defective design under strict liability the plaintiff must present evidence of a feasible, alternative, safer design.

The commonly used test for defectiveness in strict liability is the *consumer expectation* test. The Second Restatement of Torts §402A, comment *i*, reads: "The article sold must be dangerous to an extent beyond that which would be contemplated by the ordinary consumer who purchases it, with the ordinary knowledge common to the community as to its characteristics." This is often cited as the definition of "unreasonably

dangerous." This standard for strict tort liability is similar to the implied warranty of merchantability defined by the Uniform Commercial Code. The difference, however, is that the UCC does not require an item be dangerous to be unmerchantable.

The consumer expectation test is used in production defect cases for strict liability. It can be used to argue that an average consumer does not expect a soda bottle to spontaneously shatter, glass splinters to be in food, or house siding to grow mushrooms. One criticism of the consumer expectation test is that it is not helpful with complicated products. This critique is especially relevant for agents and computer technology in general. The average consumer may not have any expectations about such intricate technical items. Expert testimony can be used in such cases to establish consumer expectations, but this is skewing the true nature of the test since it explicitly applies to the "ordinary consumer."

A similar test deals with the seller instead of the consumer. The *presumed seller knowledge* test asks if the seller would be negligent in placing a product on the market if he had knowledge of its harmful effects or dangerous condition. In strict liability in tort the seller's knowledge of a defect is immaterial to the case, even if the defect is one about which he could not have known even in principle. This test asks a hypothetical question: *If* the seller had known, would he have been negligent in placing the product on the market? This test is quite similar to the consumer expectations test. One says a product is defective and unreasonably dangerous if a reasonable seller would not sell it knowing the risks. The other argues a product is defective and unreasonably dangerous if a reasonable consumer would not buy it knowing the risks involved.

Some products are unavoidably unsafe but are clearly useful and necessary. These products are highly prone to cause damage. Should they be considered defective? Section 402A of the Second Restatement of Torts, comment *k*, provides a disclaimer for strict liability against unavoidably unsafe products that are necessary. Such products are not *unreasonably* dangerous under §402A(*k*) because removing the source of danger

annihilates the product entirely. Such products include gasoline tankers and dynamite. Computer technology is so ubiquitous that it could be considered unavoidably unsafe but necessary—that is, if it *is* unsafe. Despite the efforts of Peter Neumann, who compiles a list of computer-related disasters, it is unlikely computers as a whole will be labeled as "unavoidably unsafe." Malfunctioning agents might be able to convince skeptics that computers, if not inherently unsafe, certainly have the potential to be.

The Anthropomorphic Agent Defect: Misrepresentation

Legal scholars sometimes regard misrepresentations as a fourth category of defect. Misrepresentation is not more appropriately classified as a negligence action any more than a claim in strict liability since it depends on the role of the developer. If the developer of a product exercised all due care, the misrepresentation may be innocent and not negligent, but the developer may be held strictly liable. Thus, a discussion of misrepresentation defects will be relevant to both negligence and strict liability.

Misrepresentations may be made by a seller to a customer about the nature of a product. Such a case was *Pabon v. Hackensack Auto Sales, Inc.* (1960). In this case the dealer of an automobile told the buyer not to worry about a noise in the steering of a new car. The court found that the dealer had professed expertise when he had none. This type of misrepresentation may occur with agents if a vendor claims the agents have capabilities that they do not. (This may also raise a claim for breach of the implied warranty of fitness for intended purpose or express warranty.) Misrepresentations of this kind are hard to prove, however, because courts usually regard "sales puffing" as acceptable behavior.

Most relevant to anthropomorphic agents are misrepresentations arising from a product's appearance. Such a case was *Hochberg v. O'Donnell's Restaurant* (1971). In this case the plaintiff bit hard into a martini olive that erroneously had not been pitted and

suffered injury requiring dental work. Before biting into the olive, the plaintiff had observed a hole in the end of it which led him to believe the olive was pitted. The court ruled that the product carried a misrepresentation of its true nature. Another misrepresentation case based on a product's appearance was *Greenman v. Yuba Power Products, Inc.* (1963). The court ruled that a combination power tool with inadequate set screws was defective because its appearance falsely implied a method of safe use.

This form of misrepresentation is dangerously possible in anthropomorphic agents. Recall from Chapter Two that anthropomorphic agents are graphically represented. They may be simple stick figures, animated cartoons, or even virtual-reality human beings. The central legal issue concerning anthropomorphic agents is whether or not they constitute a misrepresentation of their abilities. If they appear as intelligent personages, or specialists for some task or in some field, then people may attribute a higher degree of competence and capability to the agent. After all, the goal of HCI agents, and anthropomorphic agents in particular, is to be "a personal assistant who is collaborating with the user in the same work environment."[30:31] This cooperative working paradigm suggests that users will rely on their agents' expertise and advice. If something goes wrong because of the suggestion or action of an anthropomorphic agent, the user could sue the designer for negligently designing an agent that misrepresented its abilities and intelligence. If the designer had exercised all due care in user-testing the agent, then strict liability (either in tort or warranty) may apply.

This legal concern over anthropomorphic agents is paralleled by a heated debate in the HCI community. The two sides argue vehemently over whether or not agents should be anthropomorphized at all. The worry of the computer scientists is the same worry from a legal standpoint: Will humans place undue weight on the advice and actions of an anthropomorphic agent? Will they regard the agent as being more intelligent and having more capabilities than it does? The other side retorts by saying that "[a]nthropomorphizing interface agents is appropriate for both psychological and

functional reasons."[27:358] The benefit of anthropomorphizing agents is that "the appropriate traits are apparent and the associated styles and behaviors can be successfully predicted."[27:364] People are used to interacting with other people, faces, and personalities. If agents exhibit these, interaction with them will be more akin to human-human interaction.

Research by Byron Reeves and Clifford Nass of Stanford University has returned surprising results regarding on-screen characters. Their results are published in *The Media Equation*,[47] where they argue that people psychologically respond to on-screen characters as if the characters are real people! Nass and Reeves have done extensive psychological testing and found that fostering cooperation between a person and a computer is a good idea, justifying the theory behind HCI agents. They found that "[c]omputers and users should be peers, and people should be made to feel dependent on computers without feeling inferior or superior." Agents and people should be "on the same team" because it "encourages people to think that the computer is more liable and effective, and it also promotes cooperation and better performance."[47:159-60]

Further evidence by Nass and Reeves justifies the legal worry over , misrepresentation. They found that humans respond *socially* to "fictional representations, human or otherwise, that appear on the screen … Give anything eyes and a mouth, it would seem, and personality responses follow."[47:82-3] Explaining the importance of personality in on-screen characters, they write:

> Personality governs the nature of the affiliations that we have with other people, and people presented in the media aren't any different. Personality will determine *who* we like in the media and what we *expect* from those we encounter, and it can even influence how groups of people will get along together.[47:83]

In their explanation are three key points: the graphical representation of an agent will compel one to either like or dislike it, thus enhancing interaction or detracting from it; the graphical representation of an agent will influence expectations of its abilities and personality; and the ability to get along with an agent will turn on its apparent personality, and whether or not people like working with it.

An additional finding is that on-screen media which claims specialization in some field (as anthropomorphic agents often do) will be even more likable and trusted. "If media claim specialization, they'll be better liked and appear more competent as well. Ethical questions aside, there is little doubt that claims to specialization work."[47:149] So from a legal standpoint, the danger is in the way humans view an anthropomorphic agent. Many people are naïve about computer fallibility already; add in a seemingly intelligent anthropomorphic agent and the illusion of infallibility intensifies.

Nass and Reeves' research has powerful implications for the art of anthropomorphic agent design. It causes those who inveigh against the practice to shout even louder. It also justifies the legal worry that anthropomorphized agents have the potential to carry an express misrepresentation. Take for example an agent which provides financial advice as part of an accounting software package.[18:392] The agent may be endowed with a realistic graphical portrayal, gestures, personality, and natural language capabilities. The agent might look like a stockbroker from Wall Street, dressed in a fine suit and carrying a briefcase around the screen. If Nass and Reeves are right, the agent's user is likely to take his financial advice seriously. Perhaps the agent is mobile and can travel across the network and locate stocks which conform to a user's financial profile. The impact of the agent's anthropomorphic representation is probably significant. It may cause the user to rely too heavily on its expertise and lose money. The courts may find such an agent defective and hold its designer liable for misrepresentation.

Not only will users interact with anthropomorphic agents as assistants, but users may be represented *by* agents in their absence. Study is underway to use anthropomorphic

agents as personal representatives. An example is the application created at the FX Palo Alto Laboratory. The developers "believe that autonomous agents … provide new opportunities for constructing advanced personal representative artifacts which can not only perform a broader range of functions, but which also do a much better job of presenting an individual's persona to those who interact with it."[4:9] The application allows for easy creation of web page representatives. Rather than simply put a personal photograph on a web page, a life-like, animated, intelligent agent inhabits the site. Anyone visiting the person's home page would see its owner as an anthropomorphic agent, complete with speech, gestures, and intelligent behavior. If the owner of the page did not want his own image to be portrayed, he could use a different agent façade. As far as any visitor to the site could tell, *he* might be a *she* if the personal representative is male but the owner female.

Clearly this kind of misrepresentation has legal implications. If the site is commercial and a buyer depends on the masculine portrayal of a salesman in making his purchase, only to find out later the salesman is a youth or elderly woman, the buyer may feel cheated. Similarly, if the agent representative claims knowledge that its owner does not have, or expertise in some field in which the owner has no education, the agent could constitute a misrepresentation.

4.5 Privacy and Security

Agents have an effect outside the law of products liability. Privacy law has evolved over the last century into a tort theory much like the other torts discussed in this chapter. The legal "right to privacy" has been a hot topic for legal theorists since the turn of the century. The development of "increasingly sophisticated scientific devices"[15:475] has forced privacy law to undergo further refinement. When databases emerged which could hold the personal information of thousands of consumers the worry over privacy

intensified. Once networks connected computers across the world the already burning concerns became bonfires.

The most recent step in this scare over privacy is the introduction of autonomous software agents. Mobile agents that can travel across the network and collect information scare people, including some of the agent developers themselves like Pattie Maes.[20:291] And if the possibility of an agent *accidentally* compromising privacy is not bothersome enough, intentional privacy invasions by criminals is. Using agents as a means to enter remote systems and steal information, criminals may find the agent as their new virus incognito.

Security is closely related to privacy. "Security is mainly about preventing unauthorized access to a system and its contents."[35:270] Security measures must be taken to ensure privacy. Accidental invasions of privacy are just as likely to bring lawsuits as are intentional ones and the civilian as well as the criminal must be wary.

Anthropomorphic agents also may cause privacy infringements. Designers must select the image used to create an anthropomorphic agent with privacy in mind. The propensity of anthropomorphic agents to incur privacy violations is less widely recognized than the potential for mobile agents to do so, but no less real.

THE RIGHT TO PRIVACY

Products liability is described by negligence, strict tort, and warranty. Nowhere in products liability is "invasion of privacy" defined as a doctrine of recovery. In fact, protection from invasions of privacy is not even guaranteed by the United States Constitution, a fact surprising to most Americans. However, a common law "right to privacy" has evolved over the last century and has become a weighty issue since the emergence of computer technology.

The stipulated birth date of the "right to privacy" is December 1890 with the publication of "The Right to Privacy" by Samuel D. Warren and Louis D. Brandeis in the Harvard Law Review.[64] The authors did not invent the concept of a right to privacy. Their article was a response to the growing concern over privacy in an increasingly heterogeneous, crowded, urbanized, and socially mobile society.[68:173] Warren and Brandeis found support for a right to privacy among the current laws and case precedents of the day, often relying on bits of defamation, contract, and property law.[45:384] Their worry was over newspapers appropriating photographs and publishing secrets about peoples' private affairs. They argued that the right to privacy was a right "to be let alone." They argued convincingly that this right is as important as "the right not to be assaulted or beaten, the right not to be imprisoned, the right not to be maliciously prosecuted," and "the right not to be defamed."[64:204] This article helped establish the "tort" of privacy, but it had a long way to go before it was a widely accepted doctrine.

Privacy cases came and went, many under different theories or doctrines, and without a great deal of coherent definition for the first part of the 20th century. Then one case forced the courts to sharply define privacy: *Melvin v. Reid* (1931). The case concerned a woman who had been a prostitute and a defendant in a dramatic murder trial. She was acquitted, had since married, and enjoyed a respectable life freed from her past. A film maker discovered her story and produced a motion picture, using her real name and leaving the facts true-to-life. The woman brought a suit claiming emotional distress. While the California Supreme Court found against the defendant, it was not for emotional distress but for public disclosure of private information. This tort would become more widely recognized as a privacy tort in the following years after *Melvin*.

William Prosser—a familiar name by now—made a significant contribution to privacy law. In a 1960 article[45] written for the *California Law Review*, Prosser identified four areas of law into which most privacy-related cases fell: (*i*) intrusion, (*ii*) public disclosure of private facts, (*iii*) false light in the public eye, and (*iv*) appropriation. [45:389] It

was a significant achievement to subsume these four areas all under privacy because their individual characteristics differed significantly. All of them involved an element of unwanted emotional distress and the feeling of personal violation. As Prosser said quoting from the past, all four deal with an interference with the right of the plaintiff "to be let alone."[45:389] Prosser's classification of four privacy categories is accurate today. In fact, mobile agents pose a threat to the former two (intrusion and disclosure), and anthropomorphic agents have potential to cause the latter two (false light and appropriation). These will be discussed further below.

Other legal scholars wrote about privacy in the mid-20th century besides Prosser. Notable among them is Charles Fried. Writing for the *Yale Law Journal*, he viewed privacy as deontological. "[T]he view is Kantian; it requires recognition of persons as ends, and forbids the overriding of their most fundamental interests for the purpose of maximizing the happiness or welfare of all."[15:478] The right to privacy is a right extended to all people and is necessary, in Fried's view, for the sanctity of the most intimate human relations. Fried wrote, "Privacy is closely implicated in the notions of respect and self-respect, and of love, friendship and trust,"[15:482] and similarly, "The rights of privacy are among those basic entitlements which men must respect in each other; and mutual respect is the minimal precondition for love and friendship."[15:484] The deontological perspective of a "right to privacy" continues today as with other rights.

Besides the legal theory of scholars like Prosser and Fried, court cases have helped establish the tort of privacy. In *NAACP v. Alabama* (1958) the Supreme Court ruled that, although the word "privacy" does not appear in the Constitution, such a right could be inferred by many of its amendments.[62:270-1] The Supreme Court maintained that a right to privacy existed in *Griswold v. Connecticut* (1965), *Eisenstadt v. Baird* (1972), and *Roe v. Wade* (1973). These cases addressed individuals' right to privacy in making intimate decisions. A few years later in *Whalen v. Roe* (1977) the Supreme Court ruled

that the right to privacy also involved avoiding disclosure of personal matters,[48:629] repeating the second category of privacy tort identified by Prosser.

Along with the landmark cases during this time, Congress passed a number of acts to help protect privacy.[35:311] The 1970 Fair Credit Reporting Act requires agencies to disclose their credit records to their subjects. The Privacy Act of 1974 prevents federal agencies from disclosing personal information and allows people to view any information about them. In 1978, the Right to Financial Privacy Act was passed and limits law enforcement access to certain bank records. The 1980 Privacy Protection Act prohibits government searches of press offices without a warrant. The 1986 Electronics Communication Privacy Act makes it illegal to eavesdrop on a cellular phone conversation, computer network transmission, or read someone's email. Other acts were passed but have had little success due to loopholes and advancements in technology: the 1974 Family Educational Right and Privacy Act, the 1982 Debt Collection Act, the 1984 Cable Communications Policy Act, the 1988 Video Privacy Protection Act, the 1988 Employee Polygraph Protection Act, the 1991 Telephone Consumer Protection Act, and the 1992 Cable Act.

Many of the acts above have been compromised by the electronic storage and transmission of data, a provision not addressed by many of the acts.[62:292] The computer's unique ability to replicate data at light speed presents security problems. Writing in 1974, one author predicted the computer's impact on privacy when he wrote, "The computer has a tremendous capacity to retain a staggering volume of information, to process and control it almost instantaneously, and to make it available in virtually unlimited ways."[67:5] This observation is truer twenty-five years later. With the WWW, information can travel anywhere computers are connected by telephone lines. The widespread use of computer technology has worried many that the right to privacy may be in jeopardy. As new technologies which have the power to compromise privacy are developed, the acts and policy efforts will have to keep pace or become meaningless.

From its inception in 1890, the right to privacy has undergone many steps toward protecting the private lives of citizens. This right will be challenged more than ever before by the advent of computer technology and the advancements in computer science. Autonomous agents particularly will threaten privacy, especially mobile and anthropomorphic agents.

Mobile Agents, Intrusion and Disclosure

Mobile agents seem the most dangerous of all agent types. Moving from server to server, these agents find themselves in a variety of unpredictable environments. But this risk is worth the power of delegating a high-level task to a mobile agent, leaving it to run around unsupervised only to return with the desired information. "Mobile agents do not stay at home, working out of your computer and talking to servers; rather, they sally forth, take up residence on different servers, and then return with the information you asked for, or after accomplishing a requested task, such as purchasing an airplane ticket."[20:291] Imagining the things that could go wrong is not difficult.

The idea of mobile agents skirting around the network make a lot of people nervous, and for good reason. A mobile agent is not that far—computationally speaking—from a virus or worm, and an agent mutation could turn an otherwise innocuous chunk of code malignant. One complicating factor is that a mobile agent often must receive the authorization rights of its user to guarantee its identity. "It is often difficult to know the identity of a remote user, user surrogate, or system, particularly in distributed heterogeneous environments. Providing assurances that an identity is genuine—authentication—becomes increasingly fundamental to distributed systems."[39:211] This authentication occurs when an agent carries its user's *identification certificate*. Certificates are only valid for finite periods, and a remote agent whose certificate suddenly expires can cause unexpected problems.[7:245] The process of passing a

user's identification certificate to its mobile agent is known as *delegation*. "[P]erforming the delegation is not a big problem. The real problem is that the user has now given away the keys to the family jewels to an autonomous, invisible lump of software."[7:246] Security safeguards must be in place to keep mobile agents from touching unauthorized data and crashing unexpectedly while still allowing them enough freedom to accomplish their tasks.

Mobile agents are of particular relevance to the first two types of privacy identified by William Prosser: intrusion and disclosure. Intrusion is often physical in nature, but it is applicable to electronic intrusion as well. Prosser explains that at first the tort was used mainly for physical intrusions, but by the 1930s was "carried beyond such physical intrusions."[45:390] So invasions of privacy can be extended to computer-related intrusions.[62:260] In order to be actionable, an intrusion claim must involve prying into a thing which is intended to be kept private. However, the information need not be published to the public at large. A mobile agent that wanders into restricted computers containing confidential data is enough to constitute a privacy violation.

Public disclosure is the area of privacy tort solidified by *Melvin*. Public disclosure involves the unapproved exposing of private facts to the public at large. Three conditions must be satisfied to constitute a public disclosure action: The disclosure of the private facts must be a public disclosure and not a private one, the facts disclosed to the public must be private facts and not public ones, and the matter made public must be one which would be offensive and objectionable to a reasonable person.[45:393-6] A mobile agent could be part of a public disclosure suit if, for example, it obtains classified information from a competitor's computer system that becomes part of a corporation's competitive analysis released in their annual report. Another example is if a news gathering agent used by the press retrieves secret information from a corporate computer system or elsewhere which is then printed.

A limitation on both theories of recovery is if the information is already public. One cannot intrude into public information or unlawfully disclose already public information. This limitation is relevant for information present on a web site or server that a mobile agent may encounter. Is information on a web site already exposed to the public? The answer is probably "yes." What about other files on a server? One might argue that any information accessible over a network is implicitly welcome to network-based visitors like mobile agents. In fact, Peter Neumann argues that information intended to be truly private should not be kept on a networked computer. "If you want data, programs, or text to be really private, do not store them in a computer that can be accessed remotely. Otherwise, there are essentially no ironclad ways of guaranteeing your privacy."[39:189] Implied consent is always a bar to recovery in privacy tort and the courts will have to determine what constitutes such consent.

If information on a networked server might be considered public, then perhaps mobile agents and the information they *carry* while traveling is also public. If someone opens his server to visitors, it could be argued he has a right to inspect any visiting agent as thoroughly as possible to ensure its good intentions. If he manages to view the data the foreign agent is carrying, should he be held liable for an intrusion which occurred on his own machine?

Protecting the data a mobile agent is carrying is of concern to agent designers like Alper Caglayan. He asks the question, "If the agent carries state around with it, how does it know that the server will not spy on that state or modify it?"[7:232] He 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, 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." Caglayan notes that unfortunately "[t]here is no obvious method to protect all of this information unless the agent execution is performed

in a highly trusted environment."[7:249] Solving the security problems with mobile agents means not just with respect to the places the agents visit, but also the agents themselves.

Guaranteeing a user's privacy becomes even more difficult if agents communicate with one another about their users.[30:40] In order to communicate, mobile agents have a communication protocol—for example, Agent Tool Command Language (TCL), Knowledge Interface Format (KIF), or Agent Communication Language (ACL)—that allows them to collaborate and share information with other mobile agents.[7:9] "Collaboration can make it possible for collections of agents to act in sophisticated, apparently intelligent, ways even though any single agent is quite simple."[32:86] These *collaborative* agents may cause a disclosure of private information if they gain access to and share private information. For example, if a collaborative agent which has access to its user's credit information "told" another agent the information, a disclosure suit might be applicable. The damages could be substantial in corporate settings if a mobile agent discloses intellectual property to a competing firm.

One problem in establishing trust among collaborative agents is alleviating the fear of "persuasive" agents. Agents could be built to try to extract any information from other agents or copy any state a traveling agent has with it. This kind of worry and others like it shows "[t]rade secrecy laws were not, of course, designed with computer technology in mind, and their applicability to the computer industry has been somewhat unclear."[22:64] Mobile agents must be carefully designed to avoid intrusions and disclosures.

ANTHROPOMORPHIC AGENTS, FALSE LIGHT AND APPROPRIATION

As discussed in Chapter Two, the practice of anthropomorphizing agents is debatable. Some researchers believe that anthropomorphic agents create an illusion of competence and human capability that is dangerously misleading. Others feel anthropomorphic agents

enhance interface usability. Above in this chapter it was discussed how a misrepresentation defect can occur in anthropomorphic agents under products liability. If anthropomorphic agents have not caused enough apprehension already they will now: They also threaten privacy law.

The areas of privacy threatened by anthropomorphic agents are false light in the public eye and appropriation. False light can occur when some untrue opinion or utterance is attributed to a person; spurious books, articles, or ideas are attributed to a person; or a person's picture or likeness is used to illustrate a book or article with which the person has no connection.[45:398-9] An example of the third kind was *Peay v. Curtis Publishing Company* (1948), in which the picture of an innocent and unaware taxi driver was used to illustrate an article about the cheating practices of taxi drivers.

This third form of false light—the use of an image to create an innuendo of association between the image and message—may be caused by an anthropomorphic agent. Consider a software game intended for adult audiences in which an anthropomorphic agent is used. Though the image is not itself suggestive, its presence in the game and interaction with the user implies the agent is licentious. If the image is of an actual person, unaware of and unrelated to the software, he has been placed in a false light before the public and can take legal action. Another example is of a violent strategic war game in which an anthropomorphic agent general gives the user military advice. The advice might be brutal. He suggests bombing civilian targets, destroying villages of women and children, and using chemical weapons. If the general is a real military leader, he would be entitled to legal action based on the false innuendo that he is a vicious brute and unprincipled barbarian. One can think of many examples when a software agent might be situated in an environment which falsely implies information to the public.

Both of the above cases would also fit into the more general category of appropriation. This invasion of privacy occurs when the name, picture, or other likeness of a person has been used without his consent.[45:401-2] ("Other likeness" might be a

person's voice or gestures.) Unlike the false light invasion of privacy, appropriation does not require publicity. False light also involves fiction or inaccurate portrayal but appropriation does not. In this sense, appropriation is more general and its conditions more easily satisfied. However, appropriation *does* require the plaintiff's likeness be used to the advantage of the defendant, whereas false light does not. Thus, for an appropriation suit to succeed, it must be argued that an anthropomorphic agent resembling an unconsenting person is used to enhance a software program and increase profitability.

If the image used is that of a celebrity the suit is likely *not* an invasion of privacy. The law considers celebrities to have relinquished their right to privacy for three reasons: they have sought publicity and consented to it, their personalities and their affairs have already become public, and the press has the prerogative to inform the public about the legitimate matters of public concern.[45:411] Even celebrities have a right to privacy over the most private matters, however, such as sexual relations and familial problems. Though appropriation might not apply for anthropomorphic agents resembling celebrities, a royalty must be paid if one who uses a celebrity's likeness. One cannot design an anthropomorphic agent for music software that looks or sounds like Elvis without first paying for it.

Fortunately, designers can easily avoid invasions of privacy by anthropomorphic agents. Obtaining permission from people who serve as models for agents and avoiding false portrayals should prevent most legal consequences.

Criminal Liability and Agents

All that has been written until this point has been about civil liability: negligence, strict liability, warranty, and privacy. Autonomous agents may also incur *criminal* liability if they compromise security in certain circumstances. To think that criminal liability only

refers to intentionally harmful acts is erroneous. (Manslaughter is accidental but criminal.)

Mobile agents are the most likely agent type to incur criminal liability for security violations. "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."[7:217] As discussed above, mobile agents may invade a remote system and cause an invasion of privacy. However, if the system is a government computer, the invasion is no longer a civil violation but is criminal.

The United States Code provides for the prosecution of federal crimes. The sections most relevant to mobile software agents are 641, 1029-30, and 2314 of Title 18.

Section 641 prohibits the embezzlement, theft, unauthorized conversion, sale, or disposition of federal property. Two conditions must be present in order for §641 to apply to software agents: federal property, referred to as "thing of value" in the statute, must apply to intangible as well as tangible property; and the unprivileged copying of computer data must constitute a conversion.[62:378] If these conditions are met, then a mobile agent that copies unauthorized federal information could subject its owner or designer to larceny charges.

The Counterfeit Access Device and Computer Fraud and Abuse Act of 1984 added §§1029 and 1030 to the United States Code. Section 1030 was updated when the Computer Fraud and Abuse Act of 1986 was passed. These sections prohibit the unauthorized accessing of a government computer, an offense quite conceivable with mobile agent technology. For example, an indictment was handed down in *United States v. Fadriquela* (1985) for the defendant who gained access via remote host to USDA Forest Service computers. Sections 1029 and 1030 also protect banks and brokerage firms from unauthorized access.

The interstate transportation of stolen goods can be a federal offense under §2314. Since mobile agents often travel across the internet which spans state boundaries (and

national boundaries), their user may be held criminally liable if the agent illicitly obtains information considered to be a "good." In *United States v. Greenwald* (1973), the court ruled that the interstate transportation of stolen trade secrets in the form of documents which contained a secret chemical formula was actionable under §2314. If an agent obtains similar documents in electronic form and then transports them across state lines the same statute might apply. However, in *Dowling v. United States* (1985), the Supreme Court ruled that bootlegged phonograph records do not constitute property for purposes of §2314. After *Dowling*, the courts restricted §2314 more often to ordinary chattels.[62:379-80] The threat still remains that federal information illegally obtained by a mobile agent, even if by accident, is actionable under this statute.

Even if a mobile agent does not transport unauthorized information it still may be liable if it lands on a remote machine and uses its system resources. In *United States v. Sampson* (1978), the court ruled that a computer's time and capacities are inseparable from the possession of the physical computer itself and constitutes a "thing of value" which is subject to theft of property statutes. If an unwanted mobile agent takes up temporary residence on a server and consumes its clock cycles, memory, and other resources, then it may be actionable as a theft. This is a consideration if a large number of mobile agents are visiting the same server, since their consumption of system resources could be significant.

Though accidental invasions of federal computers by mobile agents could constitute criminal acts, it is the intentional criminal action which is most insidious. Those who commit computer crime are often referred to as *hackers*. Hackers break into systems and steal information. They might alter information or plant new information which works to their advantage. The worst hackers break in to systems to delete files and cause damage. One of the most famous hacker cases is *United States v. Morris* (1990). Morris was a graduate student at Harvard when he inserted a computer virus (technically, a worm) into a federal computer network, bringing the system down for hours, and

eventually costing millions of dollars. Morris was prosecuted under the Computer Fraud and Abuse Act of 1986 and sentenced to prison time.

Many hackers like Morris use viruses or worms to cause damage. A computer virus acts as though it is a legitimate user of a system, and as such, has almost unlimited power inside that system.[39:140] Viruses attach themselves to programs and piggy-back along. Viruses destroy files and scramble bits until the system they live on is "dead." Worms, on the other hand, are self-transporting rather than parasitic. Besides mangling data, worms replicate and spread as fast as possible to other servers.

The potential exists for a malfunctioning mobile agent to behave like a virus or worm. One article pointed out that a mobile agent is just a virus with "good intentions."[18:397] If the good intentions of an agent become corrupted, a formerly benign mobile agent could become destructive. This concern is great enough that the Telescript agent programming language (General Magic Corp.) placed life-span limits on its agents.[9:169] After a given amount of time a Telescript agent will "die," preventing it from ever becoming a rampant virus or worm.

Autonomous software agents can threaten privacy and security. Mobile agents may give rise to intrusion, disclosure, and even criminal lawsuits if not safely designed to respect privacy and security. Numerous federal laws prohibit the accessing of government computers and the transportation of federal information. Hackers could even use mobile agents to break into systems, and malfunctioning mobile agents may be no better than viruses. Anthropomorphic agents could result in false light and appropriation actions if designers are irresponsible. Clearly the concern over privacy and security with autonomous agents is real, and the courts will have to decide if the user or designer is to be held responsible.

5

REDUCING THE THREAT OF AGENTS

An ever-increasing array of rules, regulations, and administrative personnel is needed to maximize the benefits of technological practice while limiting its unwanted maladies.
– Langdon Winner

Agents take pre-existing legal questions about conventional software and confuse, complicate, and contort them. The complexity of agent systems makes them harder to predict, especially if the agents are adaptive and can change while in use. Mobile agents have the ability to wander into unanticipated environments for which they are not equipped to make intelligent choices. All of this uncertainty may absolve designers for negligence of agent-caused damages since these contingencies as a whole are unforeseeable. The imposition of strict liability, either in tort or in warranty, is contingent upon the treatment of an agent as a product, a determination that is uncertain. Neither the patent nor copyright systems seem adequate to protect software, and agents in particular. Standards for agent design have not evolved enough to create clear benchmarks for defects. These standards are not clear for conventional software, either, since many "defects" are tolerated in the most popular operating systems and programs. And finally, agents represent an uncompromising threat to security and privacy.

Phew! With such a huge list of obstacles, one might think agents are "just not worth it." But discouragement is misplaced. Many new technologies have faced a similar array of technical and legal entanglements, all of which will be worked out over time. Agent dangers must be considered *now* before the technology becomes ubiquitous. The technical and legal infrastructure must be in place to herald the arrival of this new paradigm.

Recall the distinction J.S. Mill made between instrumental goods and intrinsic goods.[22:24] Too often new technologies are considered intrinsic goods, but a moment of

reflection upon the purpose of a "technology" reveals an irony in this. Technologies, as tools, exist for the purpose of aiding humans and furthering their goals. A technology performs a function and is valued to the extent that it can achieve an end. The fetishism accompanying modern technologies is a combination of a variety of factors: media promotion, social status, and scientific fascination to name a few. But the technologies that inspire such fetishism are not intrinsic goods. One must ask therefore the following question of each new technology: Is it beneficial not just in achieving its function but to society as a whole, in both the short and long term? This question must be on the front of peoples' minds. After all, it is the actions of people who use computers and the policies created by decision makers that will determine the threat of autonomous agents; computers alone are neither necessarily beneficial nor harmful.[12:17]

The important roles in establishing a safe environment for agent technology must be played by designers and policy makers. The threats agents pose can be diminished, if not solved, by technical solutions. But not all problems will be solved with code, and the remainder of these must be handled by those who make the laws.

5.1 Design Solutions

The practice of agent design is very young. Though AI agents have been around since the '60s, HCI agents are a new concept originating in the early '90s. Such an infantile practice lacks the traditions, conventions, and community of the more established professions. Designers must act proactively to establish such standards and create safer technologies with an infrastructure of support.

Not all problems with agents can be solved technically. But certain topics can and should be addressed with technological solutions. At least three areas are available for designers to make an impact. Much of the threat agents pose will be reduced if designers are successful at addressing these.

Ensuring Security

Security is the technical half of the privacy/security concern over mobile agents. Ensuring security means providing the adequate safeguards and technical protections against privacy violations and damage. No system can guarantee one-hundred percent security. The security of a system must be shared by users who act responsibly with information.[62:251] Users also must not have unreasonable expectations about a system's security. Often what a system is *expected* to enforce and what it *actually* is capable of enforcing in the way of security is not the same.[39:98] A well-designed system can, however, make security violations a rarity.

As explained in the last chapter, agents pose a threat to privacy and security. But opportunities for security precautions exist in numerous places, from the programming language of mobile agents to devices which authenticate passwords.

Before a mobile agent does anything it must be built. Certain programming languages are better than others for building mobile agents because of their built-in network functionality. For example, the Java programming language was developed as a research effort to provide an operating system for networked devices.[7:108] Java is platform-independent, meaning it can run on a variety of machines, and is object-oriented, a strength from a software engineering point of view. As a result, Java is extremely good for building mobile agents because it has many built-in security measures for agents traveling the network. The authors of *The Agent Sourcebook* explain the advantages of Java: "First, the Java compiler provides a first line of defense. Second, Java's byte code verifier validates code before execution. Third, Java's customizable class header prohibits any unauthorized classes from loading. Finally, the Java Security Manager class enables an application to check the safety of an operation before execution."[7:111-2]

Other languages exist for building safe mobile agents. Safe-Tcl and Telescript both provide built-in security mechanisms. General Magic uses Telescript to build its mobile agents because this language rigorously encapsulates all methods and instance data,[7:249] protecting information from access except through a single method which is checked for security.

Through a malfunction, mutation, adaptation, or by duplicating itself a mobile agent could become a worm. A technical security measure exists to help prevent it. If the programming language used to build a mobile agent does not allow agents to alter other programs, then mobile agents cannot spread and multiply over the network.[7:250-1] This is *not* to say that the individual mobile agent itself could not malfunction and be destructive like a virus. But without the ability to modify other programs, a mobile agent cannot duplicate itself and spread like a worm. This restriction reduces the power of a mobile agent, as it may be useful to affect other programs (*e.g.*, for repair), but the tradeoff can be made.

Aside from the language used to build mobile agents, numerous agent behaviors can increase security. One of these behaviors is the use of *feedback mechanisms*. Feedback mechanisms are a response to the essential problem with mobile agents: "Not only can you not see what the agents are doing, they may be off doing it on the other side of the planet."[7:240] 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, she can provide the password and the agent can proceed. The one problem with this is that it reduces the power of autonomy from the agent. A human must be present before further progress by the agent can be made. However, it may be worth it in light of the severity of privacy violations.

Another feedback mechanism, which does not require human presence, is password-based authentication. This precaution requires that the launching platform and receiving platform share a common password.[7:111] A mobile agent simply sends the password back to its launching platform which automatically provides the password to the receiving platform.

If feedback mechanisms are to work, mobile agent sites must be classified for privacy. Such a classification already exists to some extent in the form of firewalls. Firewalls are "network bulwarks." A firewall prohibits access to a machine within the firewall from any machine outside the firewall. Firewalls are recognized by web browsers. Mobile agents must be built with the ability to recognize, in a uniform manner across all servers, the security level of a server and to respond accordingly.

Something akin to the *dial-back device* used in telephone security systems could be used to verify the origins of mobile agents. A dial-back device[62:359-60] intercepts a call from a user to a host system and checks the user's identification. It then calls the phone number previously stored in it which corresponds to the identification. This only allows previously specified callers into the host system who registered their identification. Such a scheme could be used with mobile agents if the agent is intercepted before reaching a host and is matched with identification, then sent on to its predetermined destination. If the mobile agent is not identified by the interceptor, it would be sent back where it came from.

As mentioned in the last chapter, not just sites are vulnerable to privacy invasions from mobile agents, but mobile agents *themselves*. When a mobile agent is carrying sensitive information around with it across the network the opportunity exists for interception. Encryption technology can reduce the chances that a mobile agent's cargo be usurped. Encryption is used by web browsers when sending confidential information over the network. The encryption notification from Netscape Communicator is shown below.

Figure 5.1. The encryption notification from Netscape Communicator.

CyberAgent uses such an encryption scheme, where the data is encrypted at the launching platform and decrypted at the receiving platform.[7:111] Since the keys to encrypt and decrypt the data only exist at the two authorized platforms, anyone else who gets a hold of the agent in between has no way to obtain the data.

Finally, more drastic measures can be taken to ensure mobile agent security. Each mobile agent should be equipped with a *recall mechanism* that forces the agent to return home immediately. *D-day* is another safety feature whereby mobile agents die off after a predetermined amount of time. This is to ensure a mobile agent that fails to return home does not wander around the network indefinitely as a virus or worm. The Telescript language has D-day functionality built in.

Many more design solutions undoubtedly exist. But besides agent-specific designs solutions, designers should also readily employ good software design practice. This includes encapsulation, modularization, and abstraction. After all, agents are still software.[70:386]

All software must be thoroughly tested before being used. Adequately testing software is difficult but not impossible because the states in most software applications are enumerable. Each can be identified and tested to guarantee that it works as intended. Agents are not so easily tested, however. They are highly complex and have numerous states. "From the complexity comes the difficulty in enumerating, much less understanding, all the possible states of the program, and from that comes the unreliability."[5:11]

The computer is unique among engineering artifacts because of its flexibility. "No other kind of machine can be changed so much without physical modifications. Moreover, drastic modifications are as easy to make as minor ones, which is unfortunate, since drastic modifications are more likely to cause problems."[58:175] This is also true for adaptive agents. Adaptive agents and artificial life agents modify themselves and the number of possible states they can attain becomes infinite.[30:32] This makes adaptive agents very hard to test. A designer cannot test an adaptive agent in all of its possible future states. An agent that adapts once every hour, choosing between just two alternatives, could be in 2^{24} = 16,777,216 possible states at the end of a single day! Imagine the possible states after a few years of use. The best a designer of an adaptive agent can do is examine the programming code and give intellectual arguments for its correctness (*e.g.*, inductive proofs).[58:174]

Mobile agents are also impossible to test completely because the number of environments they could reach is nearly infinite. Mobile agents can encounter as many environments as can be networked together. To make matters worse, the closed-world assumption is not adequate in these open system environments because the domain of information is unrestricted.[19:235] Open systems grow and change over time, so the testing of a mobile agent in such an environment can never be complete.

Mobile agents should be tested in as many test environments as possible before released into actual networks. In the testing phase a mobile agent should be allowed to propose actions without being able to carry them out.[7:256] Mobile agents can be tested on a network of *virtual machines*.[11:41] One computer can simulate another by simulating its memory, processing, and data. A virtual machine is a simulated computer "contained" within a real computer (see diagram[11:42] below). Virtual machines provide a level of protection because an agent running on a virtual machine cannot escape into the real computer. If the agent causes damage, the damage is caused to the virtual machine, not the real one. The agent has no way to know it is running on a simulated machine; it cannot "see" beyond the confines of the virtual machine into the real one or touch the data there.

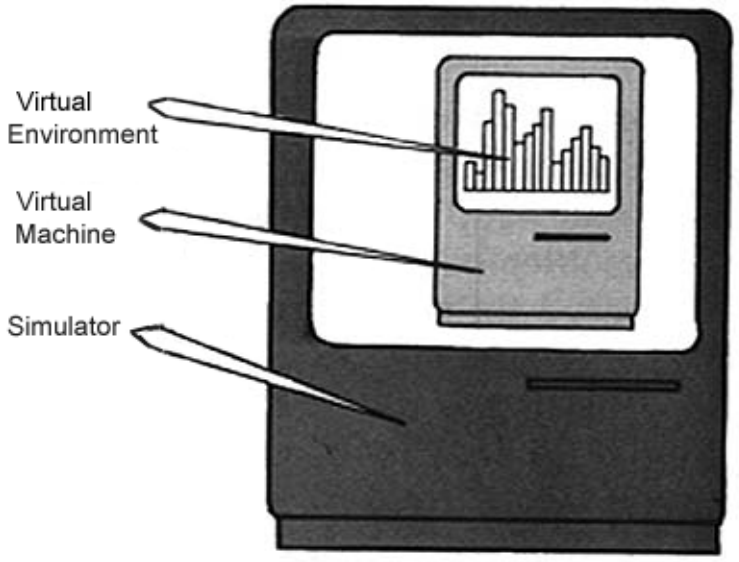

Figure 5.2. A conceptual diagram of a virtual machine.

Virtual machines are often used to run artificial life agents that might multiply out of control. A network of such virtual machines can be used as a test bed for mobile agents. If an agent runs rampant, it is only going to do so over the virtual machines. "It is quite improbable that virtual organisms can escape computational containment and

become real ones."[11:43] Testing mobile agents on virtual machines is a safe way to gauge their behavior.

Agents that are neither adaptive nor mobile are still autonomous, and to test them one must "coax" them into perform the desired behavior, since directly forcing behavior does not test the autonomous capability. One can do this by setting up the environment so as to elicit the kind of behavior from an autonomous agent one hopes to test. However, one cannot test autonomy without leaving open the possibility the agent may decide to do other than predicted. One cannot always know if an "error" in reasoning occurred or if the agent weighed the options and responded intelligently, even if unpredictably. Testing autonomy is difficult for this reason.

Rather than reinvent the advancements made in safer agent design, the industry must work to establish known patterns that can be repeated.[3:108] This reduces the chance that unpredicted problems will arise since a recognized pattern can be tested by numerous designers over time. Such patterns of design exist for other areas of computer science (*e.g.*, data structures). The array, linked list, binary search tree, priority queue, and stack are just some of the standard data structures that have been thoroughly tested, analyzed, and used in computer science. All of their advantages and disadvantages are known, and this gives programmers the freedom to pick and choose from them as needed without having to reinvent them.

Designers of software agents must do their best to test their creations. Even if dangers ride on low-percentage contingencies these dangers must be fettered out and remedied. If an event can theoretically happen, it is likely that it will happen,[39:128] especially if the agent is used over a long period of time by many people. And since all sources of danger cannot be confidently discovered, safeguards should be in place to protect against the unforeseen catastrophe.[7:138]

CONSIDERING THE USER

Designers of HCI agents should not lose sight of the two-fold goal: to build agents that simplify the usability of an interface while enhancing the interaction experience. These are subjective qualifications and make interface design as much an art as an engineering discipline. This mind-set places the user at the center of the design process rather than the technology. Design is often considered less important than engineering, presumably because of its "soft" subjective flavor. This ignorance is a criticism currently levied against the software industry by industry pundits. More and more companies are realizing, however, that attention to design in the early stages of software development pays huge dividends later on. Intuit, the makers of the now famous *Quicken* financial program, have been open about their enormous attention to design in the early stages of the program's development, a major cause for the program's success.

Agents must be designed in the same manner. Attention to the user's experience is crucial in designing HCI agents. However, many people consider agents products of AI as well as HCI, and thus they focus on the traditional AI problem: how to make the agent "do the right thing." This focus results in agents that have overly elaborate and complex planning algorithms which create room for errors. Worse yet, the user is ignored in lieu of the algorithm. HCI agents should be built with the user's experience in mind. Instead of "doing the right thing," agents should "do the thing right."

> [F]or many currently important agent tasks, it is not enough for the agent to be able to select the right thing to do. If the agent's behavior is to be understood by the user, then the manner in which the agent *does* the 'right thing' is also important. For applications such as believable and interface agents, the agent must not only choose the right actions, but also clearly express both its actions and the reasons for them.[55:24]

That is, the *manner* in which the agent does a task is as important as *what* the agent does. This is especially relevant for anthropomorphic agents, whose psychological effects upon the user are significant.[47:251-2] Users should be tested for their psychological responses to on-screen characters.

People like interfaces that are *predictable*.[57:100] A button should look like it can be pushed, and the picture on it should have a natural mapping to the function it performs. The user should not receive a startling surprise that clicking a button containing a picture of a diskette actually deletes a file rather than saves it.

Anthropomorphic agents, as part of an interface, should be predictable as well. Designers should create interface agents with attributes that correspond to actual capabilities, just like the picture on the button should correspond to the button's function.

Brenda Laurel believes agents should be more predictable than real people in order to be effective. She notes that an "agent must be represented in such a way that the appropriate traits are apparent and the associated styles and behaviors can be successfully employed." She considers complex agent personalities, especially those resembling human models, to be "noise in the system" because they make "probability causality harder to deeply in the formulation of action."[28:145] Complex agents are unnecessarily difficult to predict.

Users have been shown to enjoy even simple one-dimensional dramatic characters like stick figures, and inevitably impute elaborate personalities and histories to such characters.[28:145-6] Anthropomorphic agent designers can look to the extensive body of acting and animation to develop agents which are simple but express just the right traits.[28:146] In light of this, there seems very little reason to over-design anthropomorphic agents by making them excessively realistic. Simple agents are likely to be more predictable because only essential traits will be evident to the user.

Designers should user-test anthropomorphic agents to know what prediction errors users make. Users continually misinterpret the purpose of buttons, menus, and

commands, and similarly they will misinterpret agent behaviors. In light of the range of errors a user may make in interpreting an agent interface, designers of interfaces should spend more time anticipating human foibles.[39:209]

Making adaptive agents predictable is also a significant design challenge. "Adaptive interfaces may be unstable and unpredictable, often leading users to worry about what will change next."[57:101] The solution is to make adaptive agents practice *gradual learning*. Adaptive agents should not make infrequent drastic changes to themselves but instead many minor ones. This ensures that agents will only change substantially if a trend in their adaptations continues. If an adaptive agent's user continues to perform the same tasks with the same goals in mind, the agent can gradually adjust itself to meet those needs. Since any changes it makes at one time are minor, the user can observe a consistent trend in the agent's behavior and not be confused by the kind of erratic behavior one might observe if the agent is making drastic adaptations. The principle is similar to back-propagation neural networks, where the weights in the network shift *slightly* until the overall system learns.[51:578]

A cornerstone principle of software design is providing the user with feedback. This principle justifies error messages, audible sounds, and even blinking cursors. Providing the user with feedback is the reason that graphical buttons appear depressed when clicked. Good feedback messages allow the user to understand what the computer is doing. Providing feedback with autonomous agents means making them *transparent* rather than *opaque*. An agent is transparent when the user can "see inside" it. The agent should inform the user as to the actions performed, the adaptations made, the servers visited, the processes executed, and the data gathered.

An agent can be transparent by providing the user with a report. This is known as *verification*, since it allows the user to verify the agent's history.[7:256] The agent's history should be auditable so that the user has the option of rejecting, accepting, or editing the agent's behavior directly.[7:259] The user must be able "to trace back the actual sequence of

acts and undo any that are seen as unwarranted."[40:69] An example of such a report is shown below.

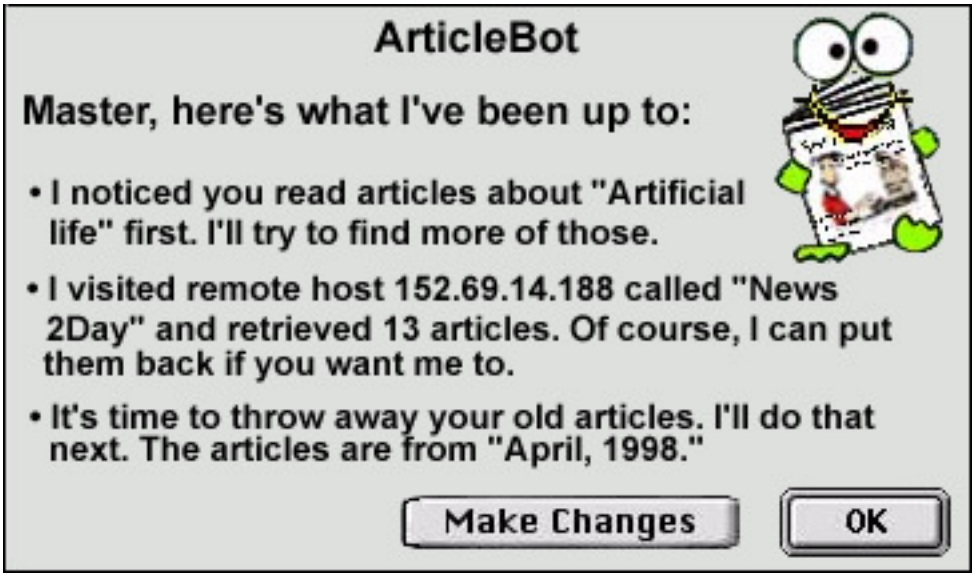

Figure 5.3. An example of an agent report creating "transparency."

Considering the user in any design task is crucial. Autonomous agents can be designed in such a way that they clarify the interface rather than obscure it. More important for agents than being intelligent is conveying the right messages. By doing so, adaptive and anthropomorphic agents will be more predictable. Feedback should be provided to create transparency and intelligent systems should provide more feedback than conventional software since their complexity is greater.

5.2 Policy Solutions

Policy makers are having to involve themselves with regulating technology more than ever before.[69:319] The policies that govern technology are often *more* influential than the technology itself because of the ways they limit, shape, or expand its reach. For example, the automobile was initially hindered because of an overly restrictive policy "which

required cars traveling the roads to be preceded by a man on foot carrying a red flag." This policy was an example of "applying old standards to a fundamentally new situation" and "robbed the auto of its intrinsic power."[12:3] What will the "red flags" be for autonomous software agents?

At the same time, however, agents are potentially too dangerous to be developed without close scrutiny. Policy makers must err on the side of caution in the choice between limiting agents too much and not restricting them enough. Paralleling the three technical solutions above are three policy-oriented areas to make agents safe.

LIMITING POWER

No *a priori* reason exists why agents should exercise as much power as is technologically possible. Pushing technology as far as it can go in the relative metrics of speed, power, efficiency, effectiveness, inventiveness, and even destructiveness is tempting. But allowing agents to make crucial decisions without human intervention is not prudent. "[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."[62:224] An agent should not be the ultimate authority for any crucial decision.

One author made the distinction between decision making under *fuzzy* standards and those under *clear* standards.[37:223] The former are decisions devoid of choices that are clearly right or wrong, best or worse. Determining what career one should pursue is a decision made under fuzzy standards. Many of the decisions made by political leaders are under fuzzy standards. Decisions made under clear standards have an obvious right choice amongst the other choices. Deciding what horse to bet on in a race is a decision made under clear standards, since whichever horse wins is the right choice and all others

are wrong. Decisions made under fuzzy standards require justifications; those made under clear standards do not, especially when made correctly.[37:223]

Autonomous agents should be built to make intelligent decisions under clear standards, but not given the power to make significant decisions under fuzzy standards. Decisions made under clear standards can be objectively evaluated. If agents are allowed to make decisions under fuzzy standards, justifications for those decisions will be required but probably not satisfactory. Every decision made by an agent should be critiqued and weighed as appropriate or not. The agent's decisions should only be accepted after its competence is rigorously tested.[37:226] And humans should never allow an agent or any other machine decide basic goals and values.[37:228]

One problem in limiting the power of autonomous agents is that they often need the authority of their user to be effective. Although the agent acts with its user's authority, limits must be imposed on its capabilities.[7:256]

The power of agents is an important consideration in electronic commerce. Agents will be used in E-commerce to help purchase stocks, make deals, and perform other transactions. The promise is that agents will "have the authority to perform transactions or to represent people in their absence."[32:84] If agents are allowed such human-like responsibility, it is unclear if a user should be held responsible for his agent's transactions.[30:40] If a user disagrees with an agent's purchase she must be able to rescind the action. An agent's actions must not be binding.

At MIT the Kasbah experiment was run to answer these questions. The Kasbah is an electronic marketplace in which people buy and sell items via autonomous agents. During the experiment a black market emerged and promises were made to buy things without sufficient funds. Pattie Maes, the project's director, concluded that opportunities for misuse, co-option, and failure for agents is ever-present in electronic commerce scenarios.[20:292]

The authority of agents to make binding decisions should be restricted. Serious decisions should always rest with humans. Even if agents are given the power to make vital decisions, their actions should be amendable by human overseers. Never should an unfeeling chunk of code have the power to drastically change human affairs.

Enforcing Privacy Rights

Many people feel that privacy is threatened by autonomous agents. While technical solutions can make computer systems more secure, regulations must be in place to provide privacy protections and enforce privacy rights. Donald Norman makes a plea to addressing privacy on all levels: "Privacy and confidentiality of actions will be among the major issues confronting the use of intelligent agents in our future of a fully interconnected, fully communicating society. We must address those issues now, not just in the technical sense, but inn the local, national, and global legal systems."[40:70]

The concern over privacy can be addressed at the policy level by both technologists and law makers. A collaboration between these two groups will be crucial to establishing judicious privacy policy.

Collaborations between technologists and policy makers are occurring. Organizations like the Association for Computing Machinery (ACM) place privacy clauses in their code of ethics (*e.g.*, §1.7). During the summer of 1998, the World Wide Web Consortium will begin reviewing the Open Profiling Standard (OPS) as part of a larger privacy effort called P3P, or the Platform for Privacy Preferences Project.[20:291,8:188] The P3P effort "allows Internet users to set default preferences for the collection, use, and disclosure of personal information on the Web."[8:188] Along with P3P, the TRUSTe initiative may also be approved. This initiative places a standard icon on web pages that links to information about a company's privacy practices providing an indication that the company is monitored for privacy practices by outside officials.[8:188] These policies will

help make privacy a reality on the network, and agents must be built to respect these regulations. Other collaborations between technologists and law makers exist in academic settings such as Stanford University.[18]

One of the major concerns over mobile agents is the transportation of sensitive information across national boundaries. The network spans most of the globe, connecting various countries, all with potentially different standards of privacy protection. Mobile agents must comply with the standards of all nations they visit or risk privacy violations. A major problem is guaranteeing the international protection of privacy.[62:254]

The United States is fairly relaxed about privacy compared to the rest of the developed world. For example, Japan and most of Europe greatly limit employers' ability to collect information about employees and user groups.[35:288] France places severe limitations on collecting personal data that identifies a person's politics, union membership, religious affiliations, or sexual orientation. Germany requires most companies to have a Privacy Owner who oversees data systems to ensure accuracy and confidentiality of the data. The German government will become involved in privacy disputes if workers are not satisfied. In Canada, one can use data only for the purpose made explicit at its collection.[35:290] And Canada has proposed a new privacy regime to restrict the private sector.[8:187] Much of the world that mobile agents might visit will have higher privacy standards than the US.

The contrast between these countries and the United States is sharp. The Office of Technology Assessment (OTA) identified three weaknesses with the US scheme of protecting privacy.[48:633] First, individuals are required to protect their own interests, and many people do not have the time nor money to do so effectively. Second, the current system only provides remedies *after* misuses have occurred. It is not preventative, but remedial. Third, US privacy policy does not compensate for the discrepancy between the limited power of the individual and the vast power of agencies. Though US laws limit the collection of personal information by the government, almost no restrictions inhibit such

practices by private firms.[35:268-9] Therefore, agents built for commercial use will have more freedom in their information gathering than agents used by the government. If a privacy violation occurs between an individual and a private corporation, it is difficult for the individual to assert her rights. The OTA concluded that these three shortcomings were exacerbated by computers and telecommunications technology.

Recently, the international scene has become worse for the United States. Beginning October 25, 1998, Europe will adopt a new privacy policy called the European Data Protection Directive. "Any country that trades personal information with the UK, France, Germany, Spain, Italy, or any of the other 10 EU states will be required to embrace Europe's strict standards for privacy protection."[8:135] The free flow of data between multinational corporations spanning these countries and the United States may be cut off because of a failure to meet the higher standard in the US.[35:314] The ramifications for mobile agents are enormous. Sites may have to be classified and agents may have to display their nationality when crossing privacy-sensitive borders. Mobile agents from the United States may not be allowed to traverse the network in European countries that adopt the higher privacy standards.

In addition, European countries cannot send personal information to nations unwilling to uphold the privacy standards. China has no privacy law and shows no signs of adopting any. The United States is also not eligible for personal information from Europe unless US privacy standards are augmented.[8:135] The White House is unwilling to change its nation's practices and is confident that the "European demands can be met by a mix of privacy-friendly business-to-business contracts, self-regulation schemes, and technology-based privacy-protection systems."[8:188] The Europeans are not so sure.

United States' privacy standards will have to be increased if mobile agents are to be an international technology emerging from the US. Under the European scheme, citizens are granted numerous rights: the right to access any data about them, the right to know where the data originated, the right to have inaccurate data rectified, the right to

take action against unlawful processing, and the right to deny marketers the use of their data for direct marketing purposes.[8:135] The United States arguably has some protections for these as well, but with insufficient enforcement. The European Union has set up efficient procedures whereby individuals can appeal to a legal authority if their rights are violated, and each European country will have a Privacy Commissioner to enforce the laws.[8:135] This system could go a long way in securing privacy rights for individuals.

Many policy makers have pushed for stronger privacy protection in the United States. Milton Wessel at Columbia Law foresaw the privacy risks posed by computing even in 1974. He proposed "the creation of a Privacy Commission with limited administrative powers, incorporating as many of the legislative and judicial ingredients as possible."[67:61] His main concern was over huge databases. The Commission would have to issue a license to anyone wishing to establish a data bank in excess of a certain size. The Commission would investigate data banks to ensure proper operation. A modern addition to such a Commission would include the overseeing of data collected by private firms, communication companies, and direct marketers. It might even include the data collected by autonomous agents and oversee agent security implementations in networks. Sweden already has such an institution: the Swedish Data Inspection Board.[67:63] This could provide a model for such an organization in the United States.

Others have proposed similar entities for privacy protection. The aforementioned evaluation by the OTA acknowledged that "it may soon be necessary to establish a Privacy Commissioner or Data Protection Board" similar to the one proposed by Wessel. The OTA claimed "[s]uch an institution would bring more visibility to the issue of personal information collection and use" and that a central place would enable individuals to exercise their rights. Additionally, Congress could receive "more complete, accurate, and timely information on agencies' practices."[48:633] The creation of such an organization in the United States would go a long way to convince Europe that the standards of privacy protection in the US are adequate.

Even if an organization devoted to privacy protection is not created, some general maxims should be upheld by the government and private agencies. Data should be accurate and checked periodically for integrity. Data should not be held for purposes other than those given at its collection. Users of technology should be fully informed about the privacy implications of existing and new technologies.[35:281] Individuals should know who will see their data if it is collected and for what purpose it will be used.[62:241]

Mobile agents must be built to respect these privacy policies. Users should know if an agent is collecting data from their machine, and if so, for what purposes. Agents should respect the privacy policies in each country they visit and be explicit about their intentions there.

Unfortunately, in the end "[l]aws that make privacy violations illegal cannot prevent violations; they can only possibly deter violations."[39:189] There will still be privacy violations despite both technical safeguards and policies. But the frequency and extent of these violations can be reduced if technologists and policy makers work together.

DISHONORING THE "HACKER COWBOY"

The end of Chapter Four addressed criminal liability and the use of agents. A larger problem underlies the potential for agents to be used criminally: the positive image of the *hacker cowboy*. This figure is portrayed throughout the popular media usually as a young, educated, freedom fighter who tackles the impersonal machine and emerges victorious. He is a Robin Hood of sorts, challenging the establishment.[33:157]

> The popular view of the successful computer criminal is interesting and somewhat unsettling. Most companies would be eager to hire personnel who fit this description. Often such people are young and ambitious with impressive

> educational credentials. They tend to be technically competent and come from all levels of employees, including technicians, programmers, managers, and high-ranking executives. These people are often viewed as heroes challenging an impersonal computer as an opponent in a game. In contrast, the corporate victim of computer crime is not a sympathetic figure. The victim is often seen as one who is caught in a trap of the victim's own creation.[33:155]

If a safe environment is to exist for potentially destructive technologies like agents, the hacker cowboy must be dethroned and replaced by the ethical user.

In general, the problem is that computer abuse tends to be glamorized. Movies like *Hackers* and *TRON*, along with books like William Gibson's famous *Neuromancer*, make the integration of reality and "cyberality" appear natural, comfortable, and edifying. Those who can conquer cyberspace can divorce themselves from "real space" and be less concerned with it. The problem is that crimes committed in cyberspace affect real space, real people, and real lives. But media like this fosters sympathy for the lone bandit and leads the public to ignore the high cost to society that computer crimes exact.[33:157]

The computer industry is young. Other industries have developed strong *cautionary tales*. These are industry fables that inform practitioners what not to do and the distinction between what is acceptable and what is not. The computer profession is still defining the special obligations of computer professionals and has not yet developed strong cautionary tales.[12:7] Convictions like that in *United States v. Morris* (1990) help send a strong message to would-be hackers, but the effects of these rulings are negligible compared to those of the popular media. And some argue that the *Morris* case actually fostered sympathy for hackers. "Some saw him as a hero who pointed out serious security flaws in an important operating system, and who did so before someone truly bent on causing damage had exploited them."[59:447]

Both at the federal and state level computer crime must be addressed. Many states such as Washington, Texas, Michigan, and Colorado have had cases under statutes addressing computer crime.[59:452-7] If agents are to be used for good rather than as destructive devices there will have to be strong cautionary tales sent in the form of case rulings, statutes, and popular messages that hacking is a serious crime. Additionally, law makers and officials will have to understand the technology well enough to effectively prosecute such crimes. "[C]ourts are often being challenged to apply stylized criminal statutes to highly technical activities which they may not fully comprehend."[59:460]

5.3 Conclusions

As is the trend with almost all modern technologies, autonomous software agents will continue to develop and improve. However, society must come to regard "develop" and "improve" not only in the technical sense, but also in the social and legal sense. For the first time in human history, a technology exists which "plays human." Agents sense, effect, decide, plan, adapt, travel, communicate, infer, actualize, and reason. Never before has such a variety of anthropocentric terms been applied *literally* to a technology. Agents do not metaphorically do these things, they *actually* do them. This is what creates the potential for liability and the necessity of technical and legal solutions.

One wonders if the scientists who created the AI architectures ever thought their technology could cause social and legal dilemmas. Autonomous software agents have leveraged an AI technology and brought it into the HCI domain. And just as the word "human" in "human-computer interaction" suggests, the HCI domain is replete with considerations beyond the technical. The realm of agents is no longer just a laboratory but is now the realm of humans. How are humans to share their realm with non-culpable actors?

The burden will inevitably fall on people. As the role of computers goes from passive assistant to independent practitioner, designers and users are going to find themselves increasingly threatened by the actions of their technologies.[13:114] One cannot hope to blame agents for mistakes and by doing so escape liability. Even autonomous technology is deterministic at the lowest level and programmed by humans, who will be liable.

Hardly a single area of liability law is not complicated by HCI agents. We saw in Chapter Four that adaptive and mobile agents greatly complicate foreseeability in negligence. Since agents are autonomous they have a great propensity to elicit the "computer mistake" defense. The chain of causation in agent-related damages may be hard to follow, and finding a culpable party is difficult. Furthermore, the debate over whether software is a product or a service is even less clear with agents since they are often developed as part of a contract for services. In these cases strict liability in tort and warranty will not apply. Additionally, the copyright and patent protection systems do not seem to work well with software in general, and even less well with adaptive agents, since their code alters itself. Legal tests for defectiveness are also not easily applied to agents since the industry standards and state of the art is unclear for such a young, research-based item. Perhaps agents should be regarded as unreasonably dangerous. Specifically, anthropomorphic agents may inherently contain misrepresentation defects because of the mismatch between their actual abilities and those they represent. Areas of the law outside products liability also are challenged by agents. Privacy is threatened by mobile and anthropomorphic agents, and mobile agents present criminals an opportunity for using viruses and worms incognito.

However, much can be done to reduce the risk of autonomous agents. Technical solutions exist to provide security in mobile agent systems. Designers can test and re-test their systems in safe environments to help predict sources of failure. And if designers place the user at the center of their design considerations, the user's experience will be

improved as well as the safety of the agent. Fundamental principles of software design such as feedback should not be lost while designing in the agent paradigm.

Policy makers will be just as influential as designers in providing a safe environment for autonomous agents. As a matter of social policy, agents should not be given any ultimate authority. As E-commerce evolves, the use of agents should be *accessory*, not *primary*, in the decision-making processes. The privacy standards in the United States must be made compliant with the rest of the developed world, if not through legislation and formal methods, then perhaps by corporate awareness and agreement. If mobile agents are to travel freely across political boundaries they must be built to respect privacy in all nations. Finally, the popular image of the hacker cowboy must be dismantled in favor of ethical computer users. Agents must not be the next six-shooter for the hacker cowboy.

Western society is at the brink of a new technological era. The millennium is nearing and the inertia in technological development is high. Inertia makes forward progress smooth but has a distressing feature: it takes a great deal of energy to reduce. This energy must come from within technologists and policy makers who will anticipate the necessary infrastructure for safe agents *before that infrastructure is essential*. Those in power must avoid the technological fetishism long enough to see whether a technology is socially beneficial. The legal system can help ensure the safety of new technologies, but as this thesis has shown, the holes are significant and will require time to fill.

Let us not be like Victor Frankenstein in a frenzy to bring agents to fruition.

APPENDIX A
AGENT CATALOG

AdHound by *AdOne Corporation* – database, information gathering
The AdHound agent searches classified advertisement databases for a user-specified item. If it finds relevant product advertisements, it emails the ads to the user.

ALIVE by *MIT Media Lab* – anthropomorphic
A user can stand in front of a 16' x 16' screen and interact with graphical agents. The system uses computer vision techniques to detect user actions. Agents can recognize human gestures, so that an animated dog will face the direction a user is pointing. No headgear or other special equipment needs to be worn by the user.

BargainFinder by *Andersen Consulting* – internet, information gathering
BargainFinder is an experimental virtual shopping agent for the web. As an example, it uses a parallel query architecture similar to meta search engines in order to query pricing and availability of user-specified music CDs. The agent takes the user's product query and submits it in parallel to a group of on-line vendors.

Beyond Mail – email filtering
A user can use a rule editor to tell the Beyond Mail agent to automate certain email actions such as forwarding, sorting, and deleting mail. The agent operates with a simple rule structure triggered by events and followed by agent actions.

Bob by *Microsoft Corporation* – anthropomorphic, proactive
Microsoft Bob is an operating system environment for new computer users. The entire system is inhabited by Bob, a graphical dog that interacts with the user and serves as a guide through the system. Bob brings a cartoon-like visual shell to the underlying wizard-style interaction. Bob does offer some proactive assistance to the user.

COACH by *IBM Corporation* – adaptive tutor
The Lisp programming language has myriad commands. COACH observes the user's actions while learning Lisp and offers tips on how to improve. After some time, if the user makes a mistake similar to one from the past, COACH will remember and show him his own mistake to help him associate it with the new. An adaptive user model, or AUM, keeps track of the user's progress.

CyberAgent by *FTP Software* – mobile
The Java programming language was used to implement this agent. It has the ability to assume a variety of goals and transport itself safely from machine to machine using encrypted data and authentication methods. The agent carries with it state, code, and auxiliary data for use at the remote site.

Document Avatars by *Xerox PARC* – anthropomorphic personal representatives

Avatars are realistic human figures. The Document Avatars are created in an Avatar Authoring Studio. After an avatar is created it is sent to inhabit a personal web page and represent the owner of the page. Avatars are prime candidates for misrepresentations.

DSS Agent by *MicroStrategy* – database
This agent enables the deployment of OLAP applications that provide filtering and notification of enterprise database information. DSS Agent applications are in use to provide ad hoc analysis and exception reporting in large data warehouses.

Electronic Workforce by *Edify Corporation* – proactive office assistants
Electronic workforce is a variety of agents designed to assist in the workplace. Electronic Workforce agents assist customers via services on the web, or by answering telephones, faxes, pagers, and interactive kiosks. Agents can answer, place, and transfer phone calls, or convert text to speech on the web.

Eudora Mail Agent by *Eudora Corporation* – email assistant
The agent built into Eudora's email application filters and sorts mail automatically. The header attributes on the mail are used as criteria for sorting, filtering, and re-routing mail.

Expressivator by *MIT Media Lab* – anthropomorphic agent environment
This system was developed as part of a Ph.D. thesis at MIT. Two graphical agents in the form of lamps inhabit the expressivator. The agents communicate with each other through very simple gestures, demonstrating that simple agents are sufficient to produce complex interactions.

Firefly by *Network Incorporated* – advisor
Firefly agents take stock of our buying and browsing habits and then offer purchase recommendations based on the behavior the behavior of like-minded people, while also facilitating the delivery of finely tuned advertisements.

Forest and Trees by *Trinzic Corporation* – data monitoring, database, intranet
Used mainly in industrial applications, this agent gathers internal company information from a Louts Notes database, then sets triggers and alarms to monitor product inventory levels.

Hoover by *SandPoint Corporation* – information gathering, internet
This internet agent gathers information from national business newspapers and assembles it into a Lotus Notes database. The agent then parses the database for information relevant to the user's wishes. It is commercially available at $233/mo.

InfoSage by *IBM Corporation* – news filtering, internet
InfoSage provides personalized information for business professionals including real-time business news, competitive data, and company profiles. Users must subscribe to a service and use client software for editing a personal profile that the agent can use.

Jango by *NetBot Incorporated, Excite Corp.* – information gatherer, advisor

Scans through hundreds of Web sites to find the least expensive price for home appliances, automobiles, or just about anything else.

Java by *Sun Microsystems* – agent programming language
The Java programming language is being used widely in general software development, but is also especially well suited for agent programming. It is an object-oriented, platform-independent language. It was developed initially for networked devices, and for this reason contains many useful features for agents, especially mobile ones which require security precautions.

Julia by *MIT Media Lab* – anthropomorphic
Julia is a text-based agent that operates in a muse (multi-user simulation environment). The agent can discourse with players, projects feelings, and retains a human-like memory.

Kalide – adaptive
This agent system was developed in the 1980s and was one of the first modern softbot agents. A user gives instructions and criticism to a kaleidoscope drawing program. The agent learns the user's preferences and draws accordingly.

Letizia by *MIT Media Lab* – internet, advisor
This agent uses the 'down time' when a user is surfing a web page to look ahead at the links and make suggestions on which ones to follow based on his viewing history. Letizia tries to figure out what subject matter the user is interested in by observing the user's surfing habits.

Maxims by *MIT Media Lab* – email assistant, graphical user interface
Email assistants observe one's email reading habits and attempts to sort and categorize email based on observed user behavior. Maxims prioritizes email and flags certain messages as important. It surfs through its memory of messages for patterns. Furthermore, it communicates its behavior to the user through cartoon facial expressions.

Menagerie by *MIT Media Lab* – anthropomorphic
Users interact with animated agents in real-time by using a head-mounted display and head-tracking device.

NewsHound by *San Jose Mercury News* – news filtering, internet
This agent automatically searches articles and advertisements to find those which match a user's profile. When it finds an article it thinks the user would like, it sends it to the user via email. Users can adjust the behavior of the agent by specifying levels of selectivity, quantity, and subject matter.

NewT by *MIT Media Lab* – news filtering, advisor
This agent takes requests from the user, observes the user's news reading habits, and finds articles. It sifts through large news databases and filters out articles it thinks the user would like to read.

Night on the Town by *General Magic Corporation* – information gathering
A Telescript agent that can purchase theater tickets, find a restaurant, or have flowers sent. It performs these and other tasks across multiple services supported by PersonaLink, AT&T's mail service.

Object Lens by *MIT Media Lab* – Collaborative, email, secretarial, database
This is an agent system in which users can create agents that sort mail, set reminders, and scour databases. The Object Lens is a groupware system that allows multiple interactions within a group.

Oliver by *Rupert Murdoch's News Electronic Data, Inc.* –mobile, anthropomorphic
Oliver fetches news and personalized information from various on-line sources. He appears as a labrador retriever to enhance the "fetching" metaphor.

Open Sesame! by *Charles River Analytics* – neural network, proactive
This agent automates users' patterns of repetition. Frequently used key sequences or clicking behaviors are automated by this agent. It operates within the MacOS, and has been available since 1993.

Payroll Agent by *Edify Corporation* – financial, proactive, assistant
The agent dials into the personnel list of a company and obtains payroll information. Then it makes any amendments to this information based on e-mail received from staffing. Managers can notify the agent via email for payroll changes and personnel changes. Edify agents also can process service requests from customers by interacting with back office systems.

PCN by *PointCast* – information gathering
This agent-based service delivers customized information to the client desktop during idle times. It provides current information to the user's desktop cache. The agent service requires client software for interaction with the server that actually does the information gathering.

Peedy by *Microsoft Corporation* – anthropomorphic, information gathering
Peedy is a three-dimensional graphical parrot that will inhabit Microsoft's friendly interfaces. Peedy is the successor to the famous Microsoft Bob (see above). Peedy will perform voice-recognition and synthesis, and can search the Microsoft Network for specific information.

Pleiades by *Carnegie Mellon School of Computer Science* – secretarial
Teams of agents are being used to help organize professors' schedules and increase productivity. Calendar apprentices, news filtering, and visitor scheduling agents are part of the Pleiades agent system.

Ringo by *MIT Media Lab* – collaborative, advisor

Ringo is an agent system in which multiple agents reside. These agents represent the interests of their users and search for movies, music, and other types of entertainment, learning from fellow agents in the system.

Silk – internet, information gathering
A user can perform multiple searches for information working in conjunction with Silk and a web browser. Silk stores text strings describing the URL in order to support the user management of search results.

Simply Money by *Computer Associates* – proactive financial advisor
A background advisor monitors personal finance transactions and offers pointers when it discerns patterns in your bill paying and other fiscal activities.

Surfboard by *Fulcrum* – database, information gathering
This agent plays a part in intranet search engines. It uses client/server architecture with an ODBC interface support. A user specifies a set of resources to be searched, such as databases or documents, and a keyword. Surfboard will perform the search and rank the results in relevance to the user's inferred goals.

System Agent by *Microsoft Corporation* – operating system assistant
This agent operates within Windows 95 to run background operations for the user. Disk defragmentation, error checking, and other operations are performed at user-specified times. The user interfaces with the agent through the task scheduler browser.

Tabriz AgentWare by *General Magic Corporation* – agent execution environment
Agents built in the Telescript language are run in the Tabriz AgentWare web environment. This environment includes the Telescript execution environment as well as a set of web class libraries for manipulating HTML. Tabriz AgentWare also includes Java support. Agents built to run in the environment have security attributes.

Telescript by *General Magic Corporation* – agent programming language
This language is used to create agents that are mobile yet safe. Telescript agents used AT&T's PersonaLink service that suspended operation in 1996. These agents go to places that act as hosts and allow the secure execution of agent programs to accomplish user-specified tasks like purchasing tickets. Among the security measures which Telescript supports is the ability to define a unique security code for each agent created. Other possible features are the ability to delegate authority to change a user's account, and life span limits followed by automatic self-destruction.

TopicAGENT by *Verity* – agent programming toolkit, proactive
Agents built with the TopicAGENT toolkit monitor user-specified sources of information for changes. If any changes take place, the agent notifies subscribers of the relevant information that matches their interests. Usually a separate agent for any given topic is created using TopicAGENT, and the agent is solely responsible for monitoring the source of information to which it is assigned.

UCEgo by *MIT Media Lab* – advisor, large knowledge base
The UNIX operating system is notorious among fans of wysiwyg (pronounced *wiz ee wig*—what you see is what you get) interfaces such as Windows and MacOS because one must memorize myriad commands for the archaic command-line interface. The UCEgo agent lives in the UNIX system and gives advice for frustrated users of UNIX.

Warren by *Carnegie Mellon School of Computer Science* – information gathering
This agent system employs a group of agents to retrieve information on the web relevant to an individual's financial portfolio.

Web Compass – internet assistant
Web Compass provides a number of web-related actions which make web surfing more efficient. After a search is performed in a web browser, the web compass agent will filter out some of the results based on user preferences. It can also summarize the content of each web site in a short paragraph for the user to review.

Wildfire – voice user-interface
This agent manages a telephone conversation with several forms of speech recognition, speech output, and task-specific grammars. It emulates a human secretary managing an executive's telephone calls and meetings.

Woggles by *Carnegie Mellon University, Stanford University, and Interval Research Corp.* – adaptive, anthropomorphic
These class of agents are being developed with the aim of believability. Eventually they will populate interfaces with the hope of allowing users to interact with them in a natural manner. Some Woggles are little creatures like Shrimp, Bear, and Wolf.

APPENDIX B
CASE CITATIONS

REFERENCES

1. Anderson, John R. *The Architecture of Cognition*. Harvard University Press, Cambridge, Mass., 1983.

2. Anderson, John R. *Cognitive Psychology and its Implications*. W.H. Freeman and Company, New York, 1995.

3. Aridor, Yariv and Lange, Danny B. Agent Design Patterns: Elements of Agent Application Design. *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, Minneapolis, May 1998, 108-115.

4. Bickmore, Timothy and Cook, Linda and Churchill, Elizabeth and Sullivan, Joseph. Animated Autonomous Personal Representatives. *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, Minneapolis, May 1998, 8-15.

5. Brooks, Frederick. No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*, April 1987, 10-19.

6. Brooks, Rodney. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation 2*, 1 (1986), 14-23.

7. Caglayan, Alper and Harrison, Colin. *Agent Sourcebook*. John Wiley & Sons, New York, 1997.

8. Davies, Simon. Europe to U.S.: No Privacy, No Trade. *Wired Magazine 6,* 5 (May 1998), 135-136, 187-188.

9. Ditlea, Steve. Silent Partners. *PC Computing*, May 1994, 160-171.

10. Duff, R.A. *Intention, Agency, and Criminal Liability*. Basil Blackwell, Cambridge, Mass., 1990.

11. Emmeche, Claus. *The Garden in the Machine*. Princeton University Press, Princeton, NJ, 1994.

12. Ermann, David and Williams, Mary and Gutierrez Claudio ed. *Computers, Ethics, and Society*. Oxford University Press, New York, 1990.

13. Frank, Steven J. What AI Practitioners Should Know About the Law. *The AI Magazine*. AAAI Press, Summer 1998, 109-114.

14. Freedman, David H. *Brainmakers*. Simon and Schuster, New York, 1994.

15. Fried, Charles. Privacy. *Yale Law Journal 77* (1968), 475-493.

16. Gemignani, Michael C. Legal Protection for Computer Software: The View from ’79. *Rutgers Journal of Computers, Technology, and the Law 7* (1980), 269-312.

17. Hayes-Roth, Barbara. A Blackboard Architecture for Control. *Artificial Intelligence 26* (1985), 251-321.

18. Heckman, Carey and Wobbrock, Jacob O. Liability for Autonomous Agent Design. *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, Minneapolis, May 1998, 392-399.

19. Hewitt, Carl. The Challenge of Open Systems. *Byte Magazine*, April 1985, 223-242.

20. Holloway, Marguerite. Pattie. *Wired Magazine 5*, 12 (December 1997), 237-239, 290-293.

21. Hospers, John. *Human Conduct, Problems of Ethics*. Harcourt Brace Jovanovich, New York, 1972.

22. Johnson, Deborah. *Computer Ethics*. Prentice Hall, New Jersey, 1994.

23. Johnson, Deborah and Snapper, John ed. *Ethical Issues in the Use of Computers*. Wadsworth Publishing Company, Belmont, California, 1985.

24. Johnson, Todd. Control in Act-R and Soar. *Proceedings of the First European Workshop on Cognitive Modeling*, 1996, 201-208.

25. Kant, Immanuel. *Ethical Philosophy*. Hackett Publishing Company, Indianapolis, 1983. Originally published in 1785 entitled *Grounding of the Metaphysics of Morals*.

26. Kimble, William and Lesher, Robert. *West’s Handbook Series: Products Liability*. West Publishing Co., St. Paul, Minnesota, 1979.

27. Laurel, Brenda. *The Art of Human-Computer Interface Design*. Addison-Wesley, Reading, Mass., 1990.

28. Laurel, Brenda. *Computers as Theater.* Addison-Wesley, Reading, Mass., 1993.

29. Lieberman, Henry. Letizia: An Agent that Assists Web Browsing. *Proceedings of the International Joint Conference on AI*. Montreal, Canada, August 1995.

30. Maes, Pattie. Agents that Reduce Work and Information Overload. *Communications of the ACM 37*, 7 (July 1994), 31-40, 146.

31. Maes, Pattie. Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Communications of the ACM 38*, 11 (November 1995), 108-114.

32. Maes, Pattie. Intelligent Software. *Scientific American*, September 1995, 84-86.

33. Mandell, Steven. *Computers, Data Processing, and the Law*. West Publishing Company, St. Paul, Minnesota, 1984.

34. Mill, John Stuart. *Utilitarianism and Other Writings*. Meridian, New York, 1962. Originally published in 1863 entitled *Utilitarianism*.

35. Miller, Steven E. *Civilizing Cyberspace*. ACM Press, New York, NY, 1996.

36. Minksy, Marvin. *The Society of Mind.* Simon and Schuster, New York, NY, 1986.

37. Moor, James H. Are There Decisions Computers Should Never Make? *Nature and System 1* (1979), 217-229.

38. Nagel, Thomas. *Mortal Questions*. Cambridge University Press, Cambridge, 1979.

39. Neumann, Peter. *Computer-Related Risks*. Addison-Wesley, Reading, Mass., 1995.

40. Norman, Donald A. How Might People Interact with Agents. *Communications of the ACM 37*, 7 (July 1994), 68-71.

41. Nycum, Susan. Liability for Malfunction of a Computer Program. *Rutgers Journal of Computers, Technology and the Law 7*, 1 (1979), 1-22.

42. Phillips, Jerry. *Products Liability in a Nutshell*. West Publishing Co., St. Paul, Minnesota, 1993.

43. Prince, Jim. Negligence: Liability for Defective Software. *Oklahoma Law Review 33* (1980), 848-855.

44. Prosser, William. The Assault Upon the Citadel (Strict Liability to the Consumer). *Yale Law Journal 69*, 7 (June 1960), 1099-1148.

45. Prosser, William. Privacy. *California Law Review 48*, 3 (August 1960), 383-423.

46. Rachels, James. *The Elements of Moral Philosophy*. Random House, New York, 1986.

47. Reeves, Byron and Nass, Clifford. *The Media Equation*. Cambridge University Press, New York, 1996.

48. Regan, Priscilla. Privacy, Government Information, and Technology. *Public Administration Review*, November/December 1986, 629-634.

49. Rosch, Winn L. Taking a Stand: The Look-and-Feel Issue Examined. *PC Magazine*, May 26, 1987, 157-164.

50. Rosenbloom, Paul and Laird, John and Newell, Allen. *The Soar Papers: Research on Integrated Intelligence*. MIT Press, Cambridge, Mass., 1993.

51. Russel, Stuart and Norvig, Peter. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 1995.

52. Samuelson, Pamela. Liability for Defective Electronic Information. *Communications of the ACM 36*, 1 (January 1993), 21-26.

53. Selker, Ted. COACH: A Teaching Agent that Learns. *Communications of the ACM 37*, 7 (July 1994), 92-99.

54. Selker, Ted. New Paradigms for Using Computers. *Communications of the ACM 39*, 8 (August 1996), 60-69.

55. Sengers, Phoebe. Do the Thing Right: An Architecture for Action-Expression. *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, Minneapolis, May 1998, 24-31.

56. Shelley, Mary. *Frankenstein*. London, England, Penguin Books, 1992. Originally published in 1818.

57. Shneiderman, Ben. Beyond Intelligent Machines: Just Do It. *IEEE Software 10*, 1 (January 1993), 100-103.

58. Shore, John. *The Sachertorte Algorithm and Other Antidotes to Computer Anxiety*. Penguin Books, New York, 1985.

59. Sprague, Robert D. Computer Crime: A Review of *Cyberpunk* by Katie Hafner and John Markoff. *Rutgers Computer and Technology Law Journal 18*, 1 (1992), 439-460.

60. Tambe, Milind and Johnson, Lewis W. and Jones, Randolph and Koss, Frank and Laird, John and Rosenbloom, Paul and Schwamb, Karl. Intelligent Agents for Interactive Simulation Environments. *AI Magazine 16*, 1 (1995), 15-39.

61. Van Inwagen, Peter. The Incompatibility of Free Will and Determinism. *Philosophical Studies 27* (1975), 185-199.

62. Vergari, James and Shue, Virginia. *Fundamentals of Computer-High Technology Law*. American Law Institute-American Bar Association, Philadelphia, PA, 1991.

63. Verity, John W. and Brandt, Richard. Robo-Software Reports for Duty. *Business Week*, February 14, 1994, 110-111.

64. Warren, Samuel and Brandeis, Louis. The Right to Privacy. *Harvard Law Review 4* (December 1890), 193-220.

65. Watson, Gary. Free Agency. *Journal of Philosophy 1xxii*, 8 (April 1975), 205-220.

66. Watson, Gary ed. *Free Will*. Oxford University Press, Oxford, 1982.

67. Wessel, Milton R. *Freedom's Edge: The Computer Threat to Society*. Addison-Wesley, Reading, Mass., 1974.

68. White, G. Edward. *Tort Law in America*. Oxford University Press, New York, 1980.

69. Winner, Langdon. *Autonomous Technology*. MIT Press, Cambridge, Mass., 1977.

70. Wooldridge, Michael and Jennings, Nicholas R. Pitfalls of Agent-Oriented Development. *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, Minneapolis, May 1998, 385-391.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor, Carey Heckman. Though I'm sure he would have preferred to conduct more meetings on the tennis court, he still graciously advised me at a moment's notice (in an office). Presenting with him at *Agents '98* was a blast. Thanks to Daniela Rus for helping us get there. Also a big help was my major advisor Tom Wasow, not just on this project but over my years at Stanford. I am indebted to him for his continual support and his effort in making Symbolic Systems the best program on campus. Avrom Faderman deserves thanks for conducting the SSP Senior Honors Seminar and helping to proof my work.

On a more personal note, my mother deserves the deepest thanks for supporting every thing I ever did, in and outside of school. When I was very little she read to me every night. Now I can read this to her. I love you mom. My father helped me conjure dreams and gave me the opportunities to pursue them. This thesis is dedicated to him. I hope to be as astounding an attorney as he is some day. Nick continues to amaze me. I am grateful to have him as my brother. Maybe someday there'll be a plaque with "Wobbrock & Wobbrock" on it. I would like that.

Those who helped indirectly on this project with support, time, and compassion were Scott Dudley, Scott Scruggs, Gentry Underwood, Jonathan Smith, Ben Olding, Matt Jolley and my roommate Ted Callahan. With friends like these, I wonder how I could have ever spent so much time writing an honors thesis.