Software-Agents and Liberal Order:

An Inquiry along the Borderline between Economics and Computer Science

by
Dirk Nicolas Wagner

ISBN: 1-58112-117-2

DISSERTATION.COM

USA • 2001

Software-Agents and Liberal Order:
An Inquiry along the Borderline between Economics and Computer Science

Dissertation.com
USA • 2001

ISBN: 1-58112-117-2

www.dissertation.com/library/1121172a.htm

Software-Agents and Liberal Order

An Inquiry along the Borderline between
Economics and Computer Science

Thesis presented to the Faculty of Economic and Social Sciences
at the University of Fribourg (Switzerland) to obtain the doctor's degree
in Economic and Social Sciences by

Dirk Nicolas Wagner

Accepted by the Faculty of Economic and Social Sciences
on 12th October 2000 on the recommendation of
Prof. Dr. Guy Kirsch (First Adviser) and
Prof. Dr. Jürg Kohlas (Second Adviser)

Preface

In the course of history we have looked at machines from very different perspectives. At the beginning of the modern age leading thinkers were not far from concluding that man is like a machine. In practice, an ambiguous life with machines began that, ever since, has been characterised by a continuously changing appearance and influence of the machine. The spectrum quickly extended from steam engines to microprocessors. The steady outward push of the technological frontier soon inspired science-fiction, which speculatively highlighted the convergence of man and machine. With repeatedly adjusted but ambitious goals, computer science has followed some of the paths imagined by science fiction. Today, the relationship between man and machine is more complex and inscrutable than ever. Nevertheless, it becomes evident that machines are about to play their own social role. Consequently, there is a motivation for the social sciences in general and for economics in particular to enter the scene, looking at machines from a new perspective.

This book adopts an economic perspective. On the one hand it is written with the broader picture of the so called New Economy in mind. It focuses on software-agents who may be considered to be potential key players of this economy in the future. On the other hand this study can be read as a modern restatement of classical liberalism. It unpacks liberalism as a possible social order for software-agents in a world where men delegate more and more actions and decisions to machines. A key feature of the following text is that new and classical elements are tightly interwoven.

Due to its interdisciplinarity, the study covers a relatively broad spectrum of functions. First, it serves to make evident that it can be advantageous to examine increasingly complex machines as social actors and to tackle questions regarding a social order for machines. Second, and more specifically, it offers researchers from computer science an economic framework which may serve as a source of orientation, inspiration and explanation for the design and evolution of complex computer systems. Third, the analysis aims to support those economists who have realised that machines designed by computer scientists open the door to a new dimension of economics: the economic analysis of social systems with software-agents. I am convinced that this room has to be entered.

The inspiration and the encouragement to pick up the subject of this book came from Professor Guy Kirsch and Professor Jürg Kohlas. I am indebted to the

director of the project, Professor Kirsch, for his stimulating and steady coaching. The learning experience drawn from the numerous discussions with their multitude of facets goes well beyond the project itself. I am grateful to Professor Kohlas who efficiently guided my entrance into the world of computer science and who gave me the opportunity to work in the instructive environment of his department.

For their willingness to share views and to comment thoughtfully on my manuscripts I thank Gerald Hosp, Hippolyt Kempf and Heribert Knittlmayer at the University of Fribourg. Similarly, but often over long geographic distances, I have received detailed and helpful comments as well as useful suggestions from Ivan Baron Adamovich, Arno Fehler and Markus Michalke. And I have extensively benefited from constructive critique from Dr. Kurt Annen, who repeatedly made the limits of my own work transparent to me. Of great help to me were Jason Liechty, who carefully proofread the final draft, and Norbert Lehmann, who repeatedly provided technological assistance. Thank you all. Despite all support any mistakes or misinterpretations are my own. Finally I am grateful to the Swiss National Science Foundation for partially funding the project (grant no. 12/555 11.98).

Fribourg, 28 June 2000 Dirk Wagner

Contents

Figures and Tables

Abbreviations

ACE	Agent-based Computational Economics
ACL	Agent Communication Language
AI	Artificial Intelligence
BDI	Belief Desire Intention
DAI	Distributed Artificial Intelligence
DPS	Distributed Problem Solving
FIPA	Foundation for Intelligent Physical Agents
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
MAS	Multi-Agent System
NIE	New Institutional Economics
PCMAS	Partially Controlled Multi-Agent Systems
PDA	Personal Digital Assistant
URL	Uniform Resource Locater
WWW	World Wide Web
Y2K	Year 2000
XML	Extended Markup Language

Part I Foundations

A delicate mission promises to be ahead. New grounds will be explored in a scientific area that can be found between computer science and the social sciences, particularly economics. It will be shown that this area is an interesting and relevant field of research. At the core of this argumentation will be the propositions that software-entities affect the social order of society and that economic theory of social order can provide insights for the design of software and software systems. These insights can be especially valuable because traditional approaches of computer science provide no general answer of how to solve the problems of social order software entities cause among each other and in human society.

Within this study machines are at the centre of interest. As economists do not normally study machines, the topic has to be made accessible to economic analysis. It will turn out that, with the help of software-agent technology, economics and computer-science can be bridged. But it is not only economists who need to gain access. Economic methodology has to be made approachable to computer-science. However, there is no point in competing with textbooks on economic theory. Rather than providing a fully detailed account of all possibly relevant theories, a methodology will be outlined that allows to capture the problems of social order in open systems of and with software-agents. To identify and analyse problems is an important task. But such efforts are often only rewarded if accompanied by the identification and analysis of potential solutions. At the end of this first part it will be proposed to analyse whether the concept of liberal order is applicable to software-agents. This appears to be especially problematic as liberal order requires actors to be free. Software-agents, however, are not normally intended to be free.

1 Computer science and economics - new actors and social order

Dave: Hal, I'm in command of this ship. I order you to release the manual hibernation control.

HAL: I'm sorry, Dave, but in accordance with sub-routine C1532/4, quote, When the crew are dead or incapacitated, the computer must assume control, unquote. I must, therefore, override your authority now since you are not in any condition to intelligently exercise it.

Dave: Hal, unless you follow my instructions, I shall be forced to disconnect you.

A. Clarke (1969): 2001 A Space Odyssey

The telescreen received and transmitted simultaneously. Any sound that Winston made, above the level of a very low whisper, would be picked up by it; moreover so long as he remained within the field of vision which the metal plaque commanded, he could be seen as well as heard. There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system the Thought Police plugged in on any individual was guesswork. It was even conceiveable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to.

G. Orwell, (1949/84): 1984

1.1 From utopia via the present days into the future

In "2001 A Space Odyssey" the human crew of the spaceship Discovery gets involved in a fatal conflict with on-board computer HAL. In Orwell's novel "1984" computer technology allows for surveillance and oppression of the people. Both stories are dystopias about a future where machines become problems although they were originally invented to solve problems. These two dystopias are no exceptions. Countless other visions exist. And like the two precedents they are either anarchistic or totalitarian.

In spite of the fact that 1984 and 2001 are rather outdated in the strict sense of the word, the dystopias have not lost their appeal. Much of the fascination descending from the works of Clarke and Orwell is probably due to the circumstance that they shadowy but emphatically draw the contours of an emerging reality. This counts for the optimistic impressions as well as for the pessimistic.

Clarke, Orwell and others were decisively optimistic regarding technological progress. Machines with outstanding capabilities play a leading role in their works.

Up to now, however, all dreams of the omnipotent machine have remained science-fiction. Still, a technological level is reached where computers as universal machines are capable of performing many of those activities that were fiction in "1984" and "2001". The required processing capacities are nearly adequate, simple speech recognition systems serve as reliable communication interfaces, so called chatterbots are convincing interaction partners in dialogues on specific subjects and chess computers beat human grandmasters (cf. also Stork, 1996). In essence, a rough portrait of today's machines could be painted along the following lines:

- At a high pace, machines become more and more powerful. Indicative of this process is Moore's law, which states that the processing power of microchips doubles every 18-24 months.
- To an increasing extent machines show characteristics that belong to a social artefact rather than to a mechanical object.
- The functions executed by machines become more important. To an increasing degree machines influence their environment physically, economically and socially.
- The way machines take over these functions changes. Less often, machines are directly manipulated by humans. Rather, more and more complex tasks are delegated to them.
- Machines no longer act in isolation but interact with humans and with other machines.

All these events are taking place as the meaning of a machine is changing. Today, the notion of a machine might better reflect software rather than hardware. The momentum seems to have shifted from the actual physical atoms that comprise a mechanical robot to the bits that make up a digital program (Bradshaw, 1997b, 4). The ambition to craft artificial humans has stepped into the background in order to leave the stage for machines that specialise in certain functions. In other words, "computers are organised much more directly around what electronic circuits are good at than they are around what people are good at" (Bailey, 1992, 68).

Turning to the potential consequences of future technologies, Clarke and Orwell drew a pessimistic picture. Currently, humans are not actively threatened by artificial actors. But approaching the capabilities of HAL also means approaching problems similar to those provoked by HAL. It seems to be unavoidable that the outlined technological achievements have their particular downsides. For instance, there is a risk that erroneous software design may lead to interruptions in the supply of infrastructural services like power, communications or public transport. Daily, significant costs arise and human life is threatened because of bugs in software, e.g. in medical or military systems (cf. Wiener, 1994, Birman/Renesse, 1997). World wide, a five-digit number of software-viruses is known to infect programs and data causing damages amounting to billions (cf. Kephart et al., 1998a). On top of that, the increasing role of the Internet leads to conditions where people and organisations can be more easily and more substantially damaged through information. So called spam is one example for this, antagonistic webpages targeting ethnic groups or corporations another. The Internet offers an environment where these activities can be executed by machines often more efficiently than by humans (cf. Leonard, 1997). In addition to the problems named so far, unintended harmful effects can arise when programs, which were designed by independent programmers or which are employed by independent users interact. This can occur even when the individual program is error-free and not supposed to cause any damage. Examples from the area of electronic commerce are shopping-bots and price-bots that have the potential to cause severe price-wars (cf. Kephart et al., 1998b). Overall, the mentioned problems differ in various ways. While combinations of all sorts are possible, the problems induced by machines can be summarised in five categories:

- Bug: A program contains errors and does not behave as designed.
- Design bug: The program code is free of errors, but there are unintended errors in the design.
- Virus: The program and the design are error-free. The virus manipulates someone else's data or programs and can intentionally cause damage.
- Devil: The program and the design are error-free. Other programs remain unchanged. The devil accesses, steals, manipulates or distributes data in full knowledge of its semantic meaning and thus intentionally creates damage.
- Emergent problem: The program and the design are error-free. Other programs remain unchanged. Data is being accessed, manipulated or distributed in full knowledge of its semantic meaning but without offending the rights of others. The combined effect of many operating and interacting programs leads to unintended, overall harmful effects.

The illustrated spectrum of problems indicates that the pessimism of authors like Clarke and Orwell turns out to contain some truths. Their pessimism, however, goes much further than the typology of problems presented here. It is systematic in character. Clarke threatens his audience with anarchy. Orwell invokes totalitarianism. So far, the projections of science-fiction have not materialised. But they do not receive opposition from sound scientific analysis either. Rather, either ignorance or latent agreement characterise the current attitude in computer science. This is fed by a prevailing anomaly in computer technology: Extremely rapid progress in computer hardware is contrasted by only average improvements to computer software (Brooks, 1995, 181). Consequently, even recognised pioneers of computer science do not exclude that current problems may some day amount to something worse. Joseph Weizenbaum emphasises that it is "better to think of new ways to avoid having problems than to simply throw more processing power at them" (cit. in Leonard, 1997, 56).

Taking Weizenbaum’s remarks at its cue, this study intends to overcome both the pessimistic view and the ignorant non-view. The often positively interpreted technological progress on one side and the negative side effects on the other side are considered to be inevitable. But it is argued that societies consisting of men and machines can keep well clear of anarchic, totalitarian and other systems where life is "solitary, poor, nasty, brutish and short" (Hobbes, 1651/1996, 84). The book focuses on the question of social order for machines. It is about computers but it is not computer science. Instead, it provides an economic analysis of a particular idea, liberal order. The functionalities of liberal order are demonstrated and it is shown how fundamental problems of order can be handled. Particularly, clues from economics and more generally from the social sciences are delivered to the much younger field of computer science, proposing that computer software can follow the principles of liberal order. While this may not be a "silver bullet", immediately delivering the much asked for order of magnitude improvements in software simplicity, reliability and productivity (Brooks, 1995,181), a general approach to understand and solve the problem of order is offered. As will be shown in detail, this includes issues like overcoming state of nature situations, accommodating individual unpredictability, escaping the small-worlds problem and achieving social order at low transaction-costs.

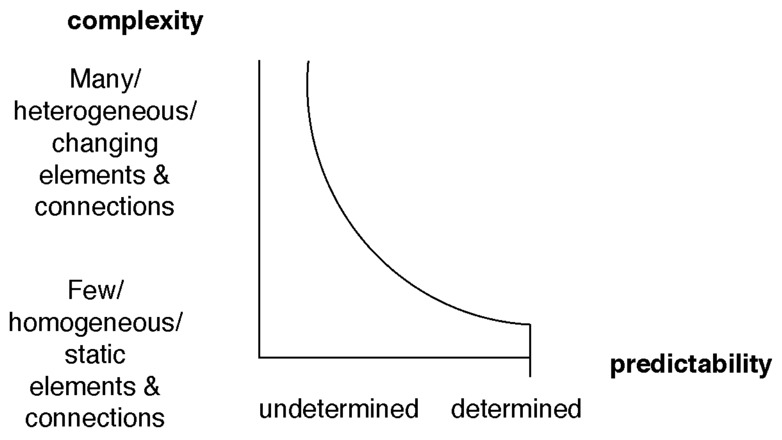
1.2 Computer science and order for small worlds

The computer is a universal machine and falls into the area of responsibility of computer science. The backbone of computer science is the order it creates in the sense that many distinct parts are assembled to form a unified, predictable whole, more concretely, a functioning program. But if the creation of order is the core competence of computer science, then why is a solid justification building on knowledge derived from economics needed, rather than on findings from computer science itself?

Over the last decades, computer science has quickly adopted the image of an engineering discipline (Gibbs, 1994; Appelrath/Ludewig, 1995). At the heart of this discipline is the structured analysis, the language representation and the conversion of real world situations into algorithms. In other words, programmers first analyse how to automate a given process and how to divide problems up into sub-problems. With the aid of appropriate algorithms they then design a detailed manual for the respective process that can be executed by a computer. Unlike other devices, the computer can manipulate any information that is described clearly enough. Without an exact description, without an algorithm, or simply without order, however, there is no chance for automation. The inversion of the argument is also popular: Automation indicates order and suggests unified and predictable wholes.

"Software entities are more complex for their size than perhaps any other human construct" (Brooks, 1995, 182). Software is complex because of the large number of parts of which it consists and because of the great variety of connections between these parts (Jennings, 1999, 1429). Continuously, new and more complex problems are to be solved by programmers. Consequently, software is getting ever more complex. In the light of this development, it becomes less evident to view automation as an indicator for order. Traditional programming only works if five conditions are met: "First, we know exactly what we want to do. Second, we can foresee every possible eventuality. Third, we can predict a correct action for each such eventuality. Fourth, we can execute each such action flawlessly. And fifth, the solutions we need are especially efficient" (Rawlins, 1998, 79). As soon as a more complex problem is to be solved, these conditions fail to exist. From an economic point of view, a trade-off between complexity and order might be suspected, i.e. the processing of more complex problems requires lower expectations with respect to order (cf. figure 1.1).

Figure 1.1: Trade-off between complexity and predictability


complexity
Many/ heterogeneous/ changing elements & connections
Few/ homogeneous/ static elements & connections
predictability
undetermined
determined

The incompleteness theorem of logician Kurt Gödel proves that, because of their complexity, computers can in principle be unpredictable for humans. For instance, it is reported that already in the 1950s the air-defence system of the United States was unpredictable: "The SAGE system was so complicated that there appeared to be no way to model its behavior more concisely than by putting the system through its paces and observe the results" (Dyson, 1997, 181). In addition to the fact that it is impossible to fully verify whether a program is correct, in practice time pressure in the development process causes software to contain even more errors than could theoretically be prevented (Gibbs, 1994).

Meanwhile, computer science is undergoing a paradigm shift. Monocentral computer systems are replaced by polycentral computer networks; closed architectures are followed by open architectures; sequentially structured computing systems disappear to make room for parallel systems. The impetus for this paradigm shift has its origin in the mentioned ambition to solve more complex problems. Basically, these systems achieve a higher fault tolerance, but they are even less predictable, and individual errors can cause devastating chain-reactions. In principle, bugs and design-bugs are more easily handled, but they appear more often. At the same time, new areas for problems open up. First, the introduction of polycentral, open and parallel systems leads to a loss of total global control which, so far, often has been taken for granted. Computer systems are no longer exclusively designed along the lines of central planning. Rather, they are being designed independently and get connected in often unforeseen ways, so that the potential for emergent problems increases exponentially. Second, the new paradigm increases the number of interfaces between computers and human society. Additionally, the social relevance and reach of machine actions grows. Problems like viruses and devils become more pressing.

Traditional computer science has only limited means at its disposal in order to meet the challenges just portrayed. Two aspects are particularly striking: First, as computer science has to be exact, it concentrates on the details while the possibility of disorder in a larger context is in danger of remaining unconsidered. Second, computer science in its role as an engineering discipline largely neglects the social relevance of its artefacts. Both arguments need explanation.

As already indicated, computer science is constrained in that it can only handle problems that can be exactly described. In order to achieve this, computer scientists regularly only envisage "small worlds" in the sense of Savage (1954/71, 8ff). A "small world" is derived from the real "grand world" by neglecting some distinctions between possible states of the world and thus by drawing a simplified picture. The small world phenomenon is perfectly illustrated by the Y2K problem caused by the representation of the dates in computer software by six figures instead of eight figures. Within a small world programmers create order. Problems are only considered if they are relevant within this world. "The problem of small worlds is that an analysis using one small world may fail to agree with an analysis using a more refined small world" (Shafer, 1986/90, 125). The ultimate level of detail is only given by the grand world in which we live. Software that is active in the grand world but whose actions are based on a small world can only be unproblematic as long as the environment perfectly adapts to this software.

Apart from this, it has to be considered that computer science is traditionally only responsible for the computer as a machine, not for the computer as an actor among other machines and/or within human society. But eventually, the described problems and the sketched paradigm shift affect social order as a whole. Problems like network breakdowns, viruses and spam go well beyond the field of computing itself, and traditional computer science does not offer the analytical tools to systematically connect issues of social order with the order of program codes.

In conclusion, computer systems become more and more complex and socially interconnected. This leads to a situation where the traditional notion of order in the sense of unified and predictable computer programs (small worlds) seems no longer sufficient because it neglects that unpredictability cannot be avoided. Furthermore, it does not meet the requirements of a grand world and lacks a social embedding of computer systems.

1.3 Economics and order for the grand world

At the outset two points were established. First, computer software creates and is confronted with problems of order. Second, computer science can no longer meet traditional expectations regarding the question of order. Only recently have computer science and related fields started to analyse polycentral, open and parallel systems in general (cf. O'Hare/Jennings, 1996) and problems of social order in particular (cf. Kephart et al., 1998b). In contrast, for other sciences both questions have proved to be fundamental issues for ages. The vocabulary and the methodologies may have changed over the centuries but the basic questions have remained the same. One of these areas of research is today's economic analysis of social order in the tradition of Thomas Hobbes and the Scottish moral philosophers David Hume, Adam Ferguson and Adam Smith. Based on new impetus from fields like the New Institutional Economics, Evolutionary Economics, Sociology, the Austrian School, and the Ordo-liberal School, the economic analysis of social order can employ a wide and sophisticated range of analytical tools and insights. This makes economics an attractive partner for computer science.[1]

One important aspect in this context is that interdisciplinarity is not only attractive but also possible: Economics and computer science are complementary sciences. In fact, interdisciplinarity between the two sciences is as old as computer science itself: When thinking of the first computer, Charles Babbage employed the economic principle of the division of labour. Since then, the chain of interdisciplinary contributions has not stopped. This series includes influential names like John von Neumann and Herbert Simon. Today the spectrum reaches from projects at MIT, the University of Michigan, the Santa Fe Institute and the Brookings Institute to activities at the corporate research centres of Xerox, IBM, NASA and others.

[1] Unless stated otherwise, the term economics in this book stands for economic analysis of social order. Another science that partners with computer science in this area of research is sociology (cf. Malsch, 1998a).

Huberman/Hogg (1995, 141) emphasise that networks of computers face the same problems as market economies do and hold that "economics may offer new ways to design and understand the behavior of these emerging computational systems." Boutilier et al. (1997, 3) draw the conclusion that the principles of polycentral computer systems are primarily economic. A growing force of researchers and practitioners employs tools from economics to handle the emerging challenges posed by a wired world. The central aim is to efficiently co-ordinate and allocate resources in and through computer systems (cf. Clearwater, 1996; Wellman/Wurman, 1997; Shoham, 1996). Often, these contributions are not merely theoretical but ready for implementation, e.g. the utilisation of a market mechanism for automated climate-control in large buildings (Clearwater/Huberman, 1995). This book will build on existing interdisciplinary work. It will fill a gap by providing a unified view of the field, instead of being tied to an individual application. It presents theoretical foundations that can widely be employed.

The mentioned interdisciplinary activities support the point that economics can be useful to computer science as far as the question of order is concerned. Practically, this is demonstrated by solutions for electronic business, telecommunications, energy, air-traffic control etc. based on economic principles (cf. Jennings/Wooldridge, 1998). An important aspect that distinguishes the present contribution from existing approaches is that it focuses on social order and that it separates the question of order from the particular problem to be solved. Such a perspective does not comply with the views of programmers designing centralised and closed computer systems where purpose and order are inherently linked. It does, however, offer an appropriate way to look at bundles of systems consisting of interconnected sub-systems which are engineered by different programmers and employed by different parties and whose borders are unclear and constantly changing. In these environments there is no single purpose and there is not one single order. Order, however, is required, the social relevance of machines has to be considered and the effects on the grand world have to be taken into account. In a sense, the economic understanding of order promoted here is less ambitious in that it does not envision software to be a unified, predictable system. Rather, various types of computer software are viewed to be elements that, like humans, are active parts of the world.

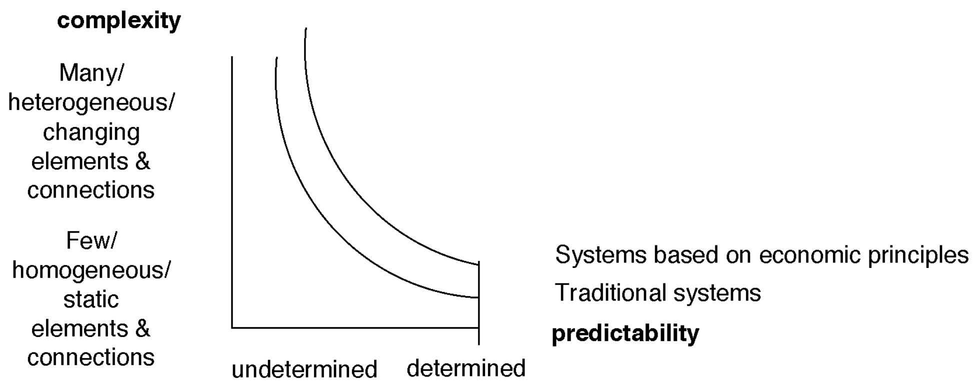
> Order, then can be defined as "a state of affairs in which a multiplicity of elements of various kinds are so related to each other that we may learn from our acquaintance with some spatial or temporal part of the whole to form correct expectations concerning the rest, or at least expectations which have a good chance of proving correct" (Hayek, 1973, 36).

The basic argument behind this definition is that under such conditions, it will be easier for the individual purposes - whatever they are - to be reached. While the appeal of this alternative view of order may not be intuitive - and can probably only be convincing after gaining support from the following chapters - the intention of this venture can nevertheless be illustrated at the outset: The widespread implementation of general economic principles of order can permit computer systems to reach a higher technology curve that indicates a higher degree of order at the same level of complexity and a higher complexity at the same level of order respectively (cf. figure 1.2). Economic principles can inform the development of polycentral, open and parallel systems.

In essence, three reasons have been identified that make it worthwhile to shrink the distance between economics and computer science: First, social order is an important research subject in economics, and this subject is affected by the developments in the area of computer software. Second, economics and computer science are complementary sciences. Third, economics can offer theoretical concepts and basic principles of order that can be employed by computer science.

Based on these premises this book aims systematically to offer access to economic understanding and to fundamental economic principles of social order for a world with complex computer systems. The approach differs from past and existing research activities both in and at the borderline of computer science and economics. This is because computers are viewed as socially relevant actors within an interconnected world, and because the perspective is not one of small worlds but one of the complex, grand world of everyday life. And, the focus is on the problem of social order, which is separated from specific purposes.

Figure 1.2: Computer systems based on economic principles reach a higher technology curve (schematically)


complexity
Many/ heterogeneous/ changing elements & connections
Few/ homogeneous/ static elements & connections
Systems based on economic principles
Traditional systems
predictability
undetermined
determined

1.4 Machines as software-agents

The introduction of many works by economists is marked by a discussion of the behavioural model they employ to conceptualise the human actor. Here, precedence is given to another actor, the machine. As a universal machine, the computer and particularly its software can be found at the centre of interest. This does not seem a narrow focus if one considers that computers are involved in a vast variety of human activities. Pervasive Computing or Ubiquitous Computing are illustrative keywords here. The focus can be narrowed by concentrating on a particular software-paradigm called agent-technology. Agent-technology conceptualises software-entities as software-agents. Before discussing the concept in greater detail, it is advisable to derive why the present study focuses on this paradigm. In fact, several reasons can be found for such a step. Although a proof for the superiority of agent-technology over other approaches of computer science is unavailable (Jennings, 1999, 1431), the relevance of software-agents increases rapidly. This is true for research as well as for applications (cf. Jennings/Wooldridge, 1998 and table 1.1). Especially on the Internet the agent-paradigm prospers. "The computational architecture that seems to be evolving out of an informationally chaotic web consists of numerous agents representing users, services and data resources" (Huhns/Singh, 1998a, 1). Empirical evidence for the coming massive penetration and the rapidly increasing variety of software-agents comes from information services like Botspot or the Gartner Group.[2] Despite recent successes, critics have repeatedly questioned the relevance of agent-technology (Shneiderman, 1997, 100). However, Jennings (1999, 1431ff) argues that agents are more than a trend because the technology can deal with the challenges posed by the design of complex computer systems. Additionally, the characteristics of agent-technology enhance its prospects of becoming a dominant paradigm in computer science. This is nurtured for instance by the fact that the conceptual basis of software is no longer determined by the computer architecture, but relies on the problems itself. On top of that, agent-technology supports the use of existing systems and programs. Existing code is not substituted but simply complemented with a new wrapper. Consequently, every software program can become an agent.

Even in the light of these arguments it might be premature to assume at the current stage of development that software-agents in the long run will be a dominant paradigm in computer science. Thus, the ultimate reason for focusing on agents originates from another source: In contrast to other computer science approaches, agent-technology is based on a "cognitive and social view of computation" (Shoham, 1997, 271). This implies that, through the agent-paradigm, machine worlds theoretically can be accessed and understood by a social science like economics. Conversely, it also means economic ideas can enter into computer science the same

[2] See: http://www.botspot.com, http://www.gartner.com.

way. In other words, if computer science and economics are to meet then most probably and most successfully it will happen via the agent-paradigm.

Table 1.1: Application areas for software-agents

- Workflow Management (Office, logistics, operations, R&D etc.)
- Network Management (Telecommunications, power, transport etc.)
- Data Mining, information retrieval and management
- Digital libraries
- Education
- Personal digital assistants (PDAs)
- Electronic Business (Auctions, supply chain management etc.)
- Entertainment
- …

The notion of an agent can be traced back to the Latin expression "agens", meaning the cause of an effect, an active substance, a person or a thing that acts, or a representative (Tokoro, 1994, 1). [3] For the notion of an agent in the sense of an artificial actor a common definition is to be searched in vain. The definition of Huhns/Singh (1998a, 1), however, can be considered to be a common denominator of many existing definitions: "agents are active, persistent (software) components that perceive, reason, act, and communicate." Similarly, a synthesis derived by Franklin/Graesser (1997, 21ff) describes an agent as "a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." Paradoxically, such definitions are simultaneously criticised both to be too vague (e.g. Shneiderman, 1997, 99f) and to be too constrained (e.g. Krogh, 1997, 149). A more comprehensive picture can be drawn by referring to a modifiable and extendable list of attributes. Table 1.2 summarises the characteristics of software-agents discussed in the literature. Usually, individual agents comprise several but not necessarily all of these attributes. Eventually, these attributes are to differentiate agents from conventional software (cf. Kautz, 1994/98, 130).

[3] In order to prevent conceptual misunderstandings, it is worth noting that throughout the book the term agent always refers to a machine while the term actor refers to both men and machines.