Cartoon Heroes
Critical Reflections on Software Agents

GISLE HANNEMYR; INSTITUTE FOR INFORMATICS, UNIVERSITY OF OSLO

We are what we're supposed to be
Illusions of your fantasy
All dots and lines that speak and say
What we do is what you wish to do.
—Aqua: Cartoon Heroes (2000)

Since the early nineties, software agents have been subject to considerable interest. Advertised as a solution to several of the many problems that annoy modern computer users, they have attracted industry funding, research talent and considerable media interest. What they have not attracted is many users and real life applications. This essay first reviews some of the research literature dealing with agents, before moving on to a cursory examination of user experiences with real-life software agents. It finds real-life software agents much simpler than their counterparts in the literature. However, even at a comparably simple and transparent level, users have problems relating to the model of the computational process represented by a software agent.

Software agents are a class of computational systems that are characterised by being capable of acting autonomously on behalf of the user, in a context- and user-dependent way (Dehn and van Mulken, 2000).

In the literature of software agents, it is frequently suggested that this technology is capable of solving a number of the very visible problems facing users of modern, highly interconnected computers, including: the information overload problem; the resource discovery problem; excessive user interface complexity (Genesereth, 1994, Maes, 1994, Maes, 1995, Johnson, 1997, Maes, 1998). The same literature also portrays software agents as significantly different from conventional computational systems. Among the chief difference listed is that they are long-lived and expected to self-evolve and adapt to the environment over time. Further, they are characterised by having embedded knowledge, goals, sensors and effectors. (Maes, 1998)

Proponents of this technology see software agents not only as a replacement for command-driven and/or direct manipulation human-computer interfaces, but also as an alternative way to create computer applications. Instead of having computer programs carry out their actions either algorithmically, according to pre-set rules, or under the control and guidance of a human operator, software agent based computer applications are expected to learn from experience, adapt to context, and even be pro-active and carry out actions on their own initiative.

The idea of autonomous software is not new. In 1959 AI pioneer John McCarthy sketched such a system called *Advice Taker* (McCarthy, 1958), but the modern concept of autonomous computer-based personal assistants did not arise before nearly thirty years later, in 1987, when Apple released a five minute video entitled *The Knowledge Navigator*. The Apple video showed a college professor making use of an animated digital valet, "Phil". A friendly human face positioned at the upper left corner of a futuristic looking computer represented "Phil", who assisted the professor by helping

Citation: Hannemyr, G. (2000) Cartoon Heroes. Critical Reflections on Software Agents, In: *NIK'2000* (ed. Fallmyr, T.), Norsk Informatikkonferanse, Bodø, 20. - 22. nov. 2000, pp. 79-90.

him locate data in remote databases, and also by performing a number of other mundane tasks.

In 1994, Communications of the ACM published a special issue on software agents, including Pattie Maes classic paper on "Agents that reduce work and information overload" (Maes, 1994), and in 1995, in Scientific American's 150 Anniversary Issue (subtitled *Key Technologies for the 21st Century*) featured another paper by Pattie Maes where she pronounces:

> Agents will bring about a social revolution: almost anyone will have access to the kind of support staff that today is the mark of a few privileged people. As a result, they will be able to digest large amounts of information and engage in several different activities at once. (Maes, 1995)

Since then, agents and agent technology has been featured in a large number of books, newspaper reports, television shows and scientific papers. In addition, there exist several annual or bi-annual conferences to cover recent development in the field, including the *International Conference on Multi-Agent Systems* (ICMAS), the *International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* (PAAM), and the *International Conference on Autonomous Agents* (AA). There is also a *Journal of Autonomous and Multi-Agent Systems* (JAMAS). In 1998 Microsoft released an Agent toolkit that greatly simplified the task of creating agent-based applications for the World Wide Web and for the Microsoft Windows operating system (Microsoft Corp., 1998).

Apple's The Knowledge Navigator, featuring "Phil"

Software agents have been the subject of some controversy. For instance, direct-manipulation proponent Ben Shneiderman believes that autonomous software agents may just be used as an handwaving excuse for not designing proper user controls in the first place (Shneiderman and Maes, 1997). It is just easier for a programmer to say that a piece of software is "autonomous" because then it has the "right" to be quirky. User interface expert Donald A. Norman (Norman, 1994) warns that indirect administration of computers may lead to user's feeling that they lose control, and that the

anthropomorphism that characterises some agent systems may result in a mismatch between the systems real capabilities and the user's expectations of it.

Virtual reality pioneer Jason Lanier has been one of the most vocal opponents of agent technology, stating:

> The idea of "intelligent agents" is both wrong and evil. I also believe that this is an issue of real consequence to the near term future of culture and society. As the infobahn rears its gargantuan head, the agent question looms as a deciding factor in whether this new beast will be much better than TV, or much worse. (Lanier, 1995)

The gist of Lanier criticism is that he believes that the agent community is incapable of delivering what they are selling, and what they currently are delivering is harmful. Because most of the fundamental questions of artificial intelligence remain unsolved, maintains Lanier, the so-called "intelligent" agents are actually very dumb. In order for them to learn, adapt and evolve as promised, *users* will be forced to modify their behaviour to accommodate for the dumbness of the software:

> In order for me to interact well with an agent program, I'd have to lower myself to its level. I'd have to start being more obvious. This is a very roundabout way of getting a program to do something. And demeaning. Imagine living your life so that a moron could understand your interests. (Lanier and Maes, 1996)

A recent joke suggests that AI (artificial intelligence) has tried to revitalise itself by reversing the letters of its abbreviation to IA (intelligent agents). At least, the IA field seems to share a number of fundamental problems with the AI field, and many of the missives directed towards "classic" AI in Hubert L. Dreyfus 1972 criticism (Dreyfus, 1972) seem just as relevant today, when applied to the field of IA.

In fact, the problem of creating an agent with autonomy, adaptability and learning capabilities conjectured by Pattie Maes and others proponents of agent technology seem to be "AI-complete". This term (coined by Phil Agre of UCLA and modelled on the term "NP-complete" in the theory of computation) implies that for an AI-complete problem to be solved, all other outstanding problems in the field of AI must be solved beforehand. Pattie Maes goes a long way in acknowledging this, when she en passant notices that so far:

> knowledge engineers […] have been unable to build a base of all common sense information that an agent might need to operate in the world at large. (Maes, 1995)

She then makes a note that such a base is now under construction in the shape of Douglas B. Lenat's CYC (enCYClopædia) project (Lenat, 1995), before cautiously concluding:

> It is too early to tell whether a CYC-based agent would have all the knowledge it needs to make appropriate decisions […] (ibid)

As a lexicographic aside, it should be noted Dr. Lenat is one of a very select group of living persons that are honoured by an entry in that great encyclopaedic work of computer folklore, *The Hacker's Dictionary.* An article describing the workings of the

bogometer (a mythical instrument used to measure *bogosity* – the quality of being conceited and useless at the same time) contains the following passage:

> The agreed-upon unit of bogosity is the microLenat (µL) or one-millionth of a Leant, in honor of computer scientist Doug Lenat. The consensus is that this is the largest unit practical for everyday use. (Steele jr. et al., 1983)

At the time of writing (spring 2000), we must conclude that the question posed by Maes in 1995 (Will a CYC-based agent have all the knowledge it needs to make appropriate decisions?) is *still* undecided, because CYC is not yet in a state where it would be meaningful to use it as a basis for an intelligent agent (and possibility that it ever shall be is diminishing).

But let us for a moment assume that the CYC project has succeeded in achieving its goal of having a complete mesh of all sorts of encyclopaedic information about the real world stored and semantically linked in a giant computer, and a hypothetical software agent with access to all this data is subsequently constructed and put at our disposal. How useful would this hypothetical agent be when ordered to perform various mundane tasks on my behalf, such as, for example, making the necessary planning arrangements for an overseas journey?

When I make travel arrangement for a business trip, and when I make arrangements for a pleasure trip, I have different, even complementary preferences on a number of different issues. An "intelligent" autonomous software agent, faithfully watching my moves to infer my habits and preferences when I plan different types of trips would have a hard time trying to make sense of the seemingly contradictory decisions unless it had a very deep knowledge about the circumstances surrounding each specific trip, circumstances that may not even be manifest. Just a cursory examination of my travels in the past year makes it very clear that this is not an issue that can be boiled down to two or more disjunct and complementary rule sets marked (say) "business", "pleasure" and so on. Some trips are characterised by abrupt switches between the modalities at different legs of the journey, while other trips take place somewhere in the continuum between different modalities.

The problem, in this particular example, is that the encyclopaedic knowledge embedded in the CYC database does is only marginally relevant to the problem at hand, and the software agent would therefore be deaf and blind as far as the specific situation and specific context surrounding the specific problem it is enlisted to solve.

(Goodwin and Duranti, 1992) uses the term *focal event* to identify the specific phenomenon being contextualized. They argue that such an event cannot be properly understood and interpreted unless one goes beyond the event itself and take into account other phenomenon within which the event itself is embedded – in this case the particular and personal circumstances surrounding each particular trip.

In contrast, a good human travel agent would learn about these circumstances through verbal interaction with the client, i.e. through a much less autonomous and much more direct approach than the indirect manipulation proposed for autonomous software agents.

Even such a simple task as searching for information on the Internet appears to be highly contextualized. My perception of what would be a suitable document is radically

different when I am searching for information in my role as a research scientist, and when I am searching for a bedtime story to read to a small child — and the rules for assessing document quality and relevance changes accordingly.

The way genre contextualises communication should not be underestimated. Genres are typified communicative actions that evolve in time in reciprocal interaction between individual actors embedded in various contexts and situations (Yates and Orlikowski, 1992). Having an agent perform communicative acts on my behalf without access to the context and therefore unable to adopt the appropriate genre for communication shall most likely result in very poor, or even dysfunctional, communication.

It is hard too see how access to encyclopaedic information about the real world will help overcoming these obstacles.

In fact, speculations about what access to vast amounts of encyclopaedic information may do for a software agent's ability to perform is way over the top. Even a fairly limited environment, where all the relevant information in principle is available online, may present an intelligent agent with obstacles that renders it useless.

As an example, let us consider what it entails to entrust a software agent with the task of a journey. This being the age of the Internet, we are happy to note that much of the information required for arranging such a trip is readily available on line. Airlines, railways, bus companies, hotels, and so on, publish a wealth of information on their web pages. All these companies also have provision for online booking on their web pages. In *principle*, all the necessary information for planning such a trip is already available online, on the Internet.

Still conducting a thought experiment, let us speculate what it entails for an intelligent software agent to assist me in planning this particular trip.

The software agent must first obtain information such as timetables, fares and seat availability from the various carriers serving the destination. To do this, the agent obviously needs to know where these resources are located. Given the frequency that major companies re-organise their web-sites, this information cannot be hard-coded into the agent, it must be discovered by the agent when needed. That is, the agent must first figure out *where* on the service provider's web site the required information is kept, and then figure out *how* to extract it. A cursory examination reveals that this is not a trivial task. None of the service providers I examined keep this information in a straightforward, easy to extract format (such as a table). Usually, all the interesting data are kept in a database, but the raw data are for a number of reasons (all of them commercial) not immediately available, but must be extracted by the information requestor through some sort of query process that often appear to be designed just as much to extract information *from* the requestor as supplying it. If the requestor succeeds in navigating this obstacle course, he is rewarded with a snapshot of the information requested, but idiosyncrasies abound. One carrier seems to return schedules that is always six hours earlier than the requested time slot, another keeps saying "nothing available" unless the requestor manages to specify a timeslot within one hour of an available departure time (and with only one service from, say, Oslo to Malaga scheduled per week, it requires a lot of tries to actually find out when that particular plane leaves).

Price lists and schedules seem to be kept well separated as a matter of principle – probably because the carriers operate multiple pricing schemes that price adjacent seats on the same journey different, depending upon the context and circumstances of the sale.

To engage in the query process needed to extract, say, a timetable fragment requires mastery of a surprising range of skills. One carrier, for instance, displays a map of the destinations it serves in the form of a bitmapped graphic image. Users are expected to select destinations by clicking on points on this map with the mouse. This is not very difficult to do for a human user, but a real challenge in pattern recognition for a software agent.

Information about special offers, bonuses, combo packages and so on are also available on the same websites, but again in a multitude of different shapes and formats. Often the presentation format, language and other structural elements in these specials are changed every time a new offer is launched.

This really boils down to a problem of ontology. For the software agent and the various travel service providers systems to be able to communicate and interact, they must share definitions and meanings of a number of very complex subjects. This is very difficult even given an incentive to co-operate, but in the world of e-commerce, commercial actors often have more incentive to compete than co-operate, and as a result, attempts to create "bargain hunter" software agents are on regular basis sabotaged by commercial actors who exploit the obvious limitations in the software agent's cognitive capacity to "poison" its internal state with false or misleading information (Hannemyr, 2000).

At this point, I would like to leave the purely academic battleground of AI (or IA) vs. no AI (or no IA), and also the disputes over the lofty claims made about and against agent technology, to look at some points closer to home. How do software agents fare in actual use?

Doing this, however, was not an easy task, because such material is curiously absent from the literature. When *M.D. Computing,* a journal than mostly covers practical developments in the field of medical informatics published a paper on "intelligent agents", the section titled "Case Study" starts with the following passage:

> Over the years, my colleagues and I have investigated several of the agentbased approaches discussed above. […] and studied agent design, user interaction and performance support. At present we are attempting to construct an ambitious group of agents […] This work is part of a larger effort to develop open-software standards-base "middleware" which can interconnect any supplier's data sets and the like. (Silverman, 1998)

What emerges from this brief passage is disturbing on at least three accounts. First: The deception. What is presented as a "Case Study", is clearly not. It is a sketch of a yet un-implemented software pipe dream. Second: The hubris. The ultimate goal for the project (to develop software "which can interconnect any supplier's data sets and the like") is both grandiose and imprecise. There is nothing in the passage that suggests that the author is aware of the complexity and/or scale of this task. Third: The lack of context. All the major attributes of the proposed system (open-software standards-base "middle-ware") are computerese buzzwords. There is nothing here to suggest that the

undertaking require knowledge about things beyond computers, programming, communication protocols and so on. Reading the full paper, it very little to suggest that the proposed system have uses that are specific to medical informatics – it could just as well be a generic paper on "intelligent agents" in a computer journal.

It is curious that so much of the empirical literature dealing with agent technology seems to concentrate on lab experiments, often measuring very limited variables out of context, while apparently ignoring real-life usage situations. A recent survey of "empirical research" of the *persona effect* (the effect of having the software agent visualised as a more or less anthropomorphic figure) (Dehn and van Mulken, 2000) covering more than 100 experiment by various research groups does not list a single experiment measuring the effects of a production agent system in real-life usage situations. Instead various prototypes and mock-ups are examined in the laboratory, where for instance human forms presented on the computer screen are reported to be perceived as "more intelligent" than simple geometric forms. Dehn and van Mulken then offer the following wry reflection:

> In this study, however, subjects did not interact with the system and thus did not have any information on physical objects except for their physical appearance. Under this circumstance, it appears almost trivial that – corresponding to the subjects' pre-existing world knowledge – a geometrical object is rated as less intelligent than an anthromorphized shape. (Dehn and van Mulken, 2000)

This does not mean that software agents do not occur in in-use software. Software agents appear in various shapes and sizes in a number of much used commercial software these days, and for the rest of the present essay, I shall discuss some of the software agents that share the characteristic that they are implemented and in use.

One of the first real (as opposed to video-simulations and laboratory demonstrators) attempts to deploy agent technology commercially was Microsoft "Bob". "Bob" was launched in 1995 as an *alternative* and "user-friendly" interface to Microsoft Windows. Its target audience was computer novices and inexperienced users. It did not catch on.

While reviewing the scientific literature on agents in preparation to writing this essay, I was surprised that I did not come across a single empirical study of such a widely deployed system as Microsoft "Bob".

Searching for references to Microsoft "Bob" on the World Wide Web yielded little facts beside a few historical notices acknowledging Microsoft "Bob" as one the first software agent deployed in a commercial package. In fact most references to Microsoft "Bob" on the World Wide Web was jokes (such as the *Doctor Fun* panel shown below).

One of the stories related to "Bob" surfaced on the Usenet Risks forum (Risks, 1995). The story relates how the built-in software agent, trying to adapt to the user's behaviour and eager to please, assumed that if a user had mistyped his or her password three times in a row, the user must have forgotten it. The agent then proceeded to ask the user if he or she wanted to pick a new password. As the story goes, the agent certainly is *friendly* enough, irrespective of whether its client is the legitimate computer user or just a stranger passing through, but at the same time the agent emerge as somewhat lacking in the system security department.

Unix gurus in hell

A typical Microsoft "Bob" joke

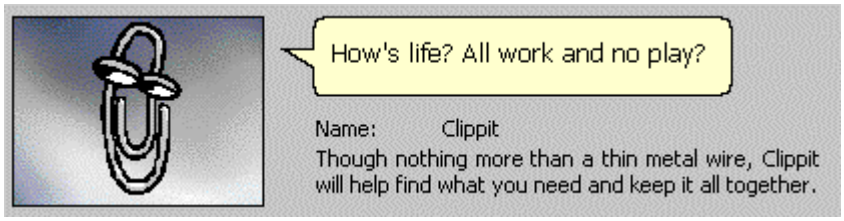
As told in the Risks forum, the story is presented as mildly humorous anecdote about the perils of (too) user-friendly software. But the story also serves as a blatant demonstration of what is the fundamental problem of knowledge representation in the artificial world within the computer. In short, the software agent has no knowledge about the difference between a legitimate user of the system and an intruder. The difference is of course quite obvious out in the "real world", but knowledge about the state of the real world is not available in the cartoon universe inhabited by Microsoft Bob. Hence, the software agent pursues the major goal "user-friendliness" with a rather silly response to a security risk as a result.

Apparently, Microsoft "Bob" lives on as the *Office Assistant* – the online user help system embedded in Office'97 and Office 2000. The *Office Assistant* is the paper clip Clippit and other cartoon characters (referred to by one Windows user who had *not* seen the above Doctor Fun cartoon as "that infernal paper clip") that pops up every time somebody using Microsoft Office'97 requests help from the system.

Most Office users are familiar with the Office Assistant, and it would be very interesting to conduct a real-life study of their interaction with it, to determine what sort of effects, if any, the inclusion of the animated agent has made to their perception of the Office online help system.

Norman (Norman, 1994) conjectures that anthropomorphism might result in a cognitive mismatch between the user's expectation and an agent's actual behaviour, and this may

be the cause of user's problem with using agents. Actually, many of the software agents available to users on the Internet today do not make use of human faces or any other pretence of being "human", but still seem to cause such cognitive mismatches.



The ghost of Microsoft "Bob"

For example, online stock brokers such as Datek and ETrade in the US, and Netfonds and Stocknet in Norway do not let their users buy or sell shares in the market directly. Instead, they provide a user interface to the trading system that can be viewed as a (very simple) software agent. To buy a set of shares, the user has to tell the agent his preferences: For instance, whether he would prefer to buy or sell the shares as fast as possible, no matter what the price is, or whether he rather would see that agent met a certain price objective, and not buy or sell any shares until the objective is met. In the latter case, the user could log off the system, and the agent would monitor the stock market and exercise the transaction as soon as the criteria established by the user is met.

On the simple level described above, the agent's behaviour seems to be totally predictable. There is no adaptation, no learning and no evolution involved. The agent is instructed to buy shares through a straightforward buy-imperative that the user may express as follows: "Buy such-and-such stock, provided the price you pay is below such-and-such level". And for selling stock, the agent can be instructed with a complimentary sell-imperative: "Sell such-and-such stock, provided the price you get is above such-and-such level".

Lately, one of the online brokers I study introduced another imperative as part of their agents' capabilities; namely a concept known as "stop-loss" among traders. The idea behind "stop-loss" is to minimise losses by selling a falling stock before it has fallen too far. Again, the command to the broker software agent is a simple imperative: "Sell such-and-such stock as soon as the price falls below such-and-such level".

The initial implementation of the "stop-loss"-imperative was flawed in not being aware of context at all. This meant that if a stock fell below the pre-set "stop-loss"-level – it was sold, and if it at a later time rose above the pre-set "sell"-level, it was sold again. Fortunately, this was a fairly rare occurrence, but when it happened it had the unfortunate consequence that the agent sold the shares twice, and as the user no longer owned the shares when they were sold the second time, the user had to repurchase the shares in the market to cover the shortage. Such purchases were likely to take place at a higher price than the price the agent had sold the shares for.

The above implementation of the stock trading agent is obviously just plain wrong, but demonstrates the point that even in what must be characterised as a very simple situation, it is possible for programmers creating agents to ignore even the most obvious of contexts.

After its introduction, the "stop-loss"-imperative has caused much grief for those who were bold enough to use it. Below is an excerpt from one trader's sad tale about losing

money through the stop-loss"-feature in the Stocktalk online chat group (*http://stock-talk.solbors.no/Forum2/HTML/012680.html*):

> I am a zealous follower of Frontline [a shipping company]. I have over the last weeks accumulated 10000 [shares] at an average cost of [NOK] 64 [per share]. I was of course overjoyed when they were valued at [NOK] 72.50 [per share] today. Checking my mailbox for the first time since Saturday, I was surprised to find a note saying they had all been sold on Monday at [NOK] 62 [per share]. Wondering what had happened, I finally realised that a SL [stop-loss] order I had intended to have active for a single day had not been cancelled and had survived [in the system] for two weeks.

For users, dealing with autonomous software clearly is a challenge. There is no strong AI or clever and obscure inference involved in this situation, just a straightforward imperative that in principle is both transparent and predictable. Still, the situation involving entrusting the software agent with buying and selling stock is radically different than when performing the same task through a human agent (such as a "real-life" stockbroker). In all likelihood, a human stockbroker would have understood that a two-week old "stop-loss" limit no longer was relevant to the situation at hand, and would have made inquiries before selling the stock at such a low price.

Online auction houses also provide their patrons with agents, in the shape of a mechanism known as proxy bidding. Users instruct proxy bidders to bid on items in their absence. They do this by instructing a software agent with the maximum they are willing to pay for an item. After that, the agent will monitor the auction and bid on the user's behalf until it wins or until it reaches the pre-set maximum limit. Now, if two or more software agents participate in the same auction, they will bid against each others at a furious pace, thereby increasing the going price of an item very rapidly. Users new to online auctions, perhaps only familiar with human proxy bidders sitting discretely at the back row at Sotheby's or Christie's are reportedly taken completely by surprise by this effect of enlisting the assistance of an software agent. Unless they are really set on obtaining an object, experienced participants in online auctions therefore report that they avoid using agents to bid for them. Instead they prefer to adopt a strategy known as "sniping". "Sniping" involves monitoring the closing minutes of an online auction live, and if the price is still is right, place a slightly higher bid seconds before the auction closes.

Another readily available type of agent is the "personalised" information collector that was pioneered by IBM in the (now defunct) *infoSage* project. This is a variation of the web *portal* (an access point to selected Internet resources such as newswire, entertainment and reports) that collects information from the Internet guided by goals implicit in a personal user profile and then uses push technology to deliver this information to the user at regular intervals.

Examples of real-life "personalised" information collectors include the Wall Street Journal *Personal Edition* and Individual.com's *My NewsPage*. Both let users set up and maintain a personal profile that typically includes the names of companies and industry sectors to watch, along with a set of keywords used to prioritise between alternate sources.

Preliminary findings indicate that user experiences with this type of services are mixed. Some users report that they find such services of value, and use them to watch competitors and important technology developments. Others say that they find their precision lacking and resolution too coarse, and that this type of service therefore tends to compound the problem of information overload rather than alleviating it.

The most successful (in terms of user acceptance) type of software agent deployed so far is what is generally known as collaborative filtering or recommender systems. Collaborative filters provide users on the World Wide Web with guidelines for locating products (e.g. books, videos, CDs) or resources (e.g. Usenet articles, websites) that the system predicts the user will like, based upon the preferences of other users with similar tastes. For example if it is known that a significant number of persons who like the music of Pete Seeger also like the music of Woody Guthrie, the system would predict that a new user who reveals that he likes the music of Pete Seeger also would buy the music of Woody Guthrie – and adopt a marketing strategy targeting that particular user accordingly.

One notable feature of recommender systems is their hybrid nature. They work by enrolling the human users of the system in the task of classifying resources and then translate their contribution into what is presented as "intelligence".

One of the best known collaborative filters in use is the "personalised" product recommendations provided by the *Amazon.com* Internet store, but other popular collaborative filtering systems include Direct Hit (for searching the World Wide Web) (*http://www.directhit.com/*) and GroupLens (for searching Usenet discussion forums) (*http://www.cs. umn. edu/Research/GroupLens/*).

Many users express amazement and even delight when first exposed to collaborative filters. At the moment, they are the only software agents that are known to "work" fairly well. Unfortunately, collaborative filters are not without problems of their own. In general, they do not work unless the user agrees to part with generous amount of personal information raising obvious privacy concerns, and they are subject to a number of purely technical problems such as *The Early Voter Problem*, *The Freeloader Problem* and *The Critical Mass Problem* (Kjeldaas, 2000). Last, but not least, the collaborative nature of collaborative filtering means that systems based on this particular technology only works well as long as the actors in the network chose to collaborate (Hannemyr, 2000).

In conclusion, software agents has been proposed as a solution to a number of important current research questions, including: the information overload problem; the resource discovery problem; excessive user interface complexity.

However, with the exception of personalised information collectors and collaborative filtering systems, little progress in addressing these research questions has been made since 1995. The few software agents that exist are for all intents and purposes cartoon heroes, inhabiting cartoon worlds and only capable of cartoon character reasoning. I.e.: they do not learn, adapt or evolve. Looking back at the list of agent characteristics near the beginning if this essay, we find that the only agent-like quality present in software agents that are deployed and in-use is the capability for autonomous action. How well the users' interests are served by these agents remains, however, an open question.

References

Dehn, D. M. and van Mulken, Susanne (2000) The impact of animated interface agents: a review of empirical research, *International Journal of Human-Computer Studies,* Vol. 52, pp. 1-22.

Dreyfus, H. L. (1972) *What computers can't do : a critique of artificial reason,* Harper & Row, New York.

Genesereth, M. R. (1994) Software Agents, *Communications of the ACM,* Vol. 37:7, July, pp. 48-53,147.

Goodwin, C. and Duranti, A. (1992) Rethinking context: an introduction, In: *Rethinking context. Language as an interactive phenomenon*, (Eds, Goodwin, C. and Duranti, A.), Cambridge University Press, Cambridge, pp. 1-42.

Hannemyr, G. (2000) The Internet is not your friend, *Bibliothek Forschung und Praxis*, (forthcoming).

Johnson, S. (1997) *Interface culture: how new technology transforms the way we create and communicate,* HarperCollins, San Francisco.

Kjeldaas, A. (2000) *Personalized Search Engines,* Hovedoppgave; Fakultet for fysikk, informatikk og matematikk, NTNU, Trondheim.

Lanier, J. (1995) Agents of Alienation, *Journal of Conscious Studies,* Vol. 2:1, pp. 76-81.

Lanier, J. and Maes, P. (1996) *Brain Tennis: Intelligent Agents = Stupid Humans*, last updated: *15-24 July 1996*, <http://hotwired.lycos.com/braintennis/96/29/index1a.html>.

Lenat, D. B. (1995) CYC: A large-scale investment in knowledge infrastructure, *Communications of the ACM,* Vol. 39:11.

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

Maes, P. (1995) Intelligent Software, *Scientific American,* Vol. 273:3, September, pp. 84-86.

Maes, P. (1998) *Software Agents*, last updated: *Feb. 18 1998*, <http://lcs.www.media.mit.edu/courses/agents98/Tutorial/index.htm>.

McCarthy, J. (1958) Programs with Common Sense, In: *Teddington Conference on the Mechanization of Thought Processes,*.

Microsoft Corp. (1998) *Developing for Microsoft Agent,* Microsoft Press.

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

Risks (1995) *The Risks Digest*, last updated: *13 May 1995*, <http://www.infowar.com/-iwftp/risks/Risks-17/risks-17.12.txt>.

Shneiderman, B. and Maes, P. (1997) Direct manipulation vs. interface agents: excerpts from debates at IUI'97 and CHI'97, *Interactions,* Vol. 4:6, November-December, pp. 42-61.

Silverman, B. G. (1998) The Role of Web Agents in Medical Knowledge Management, *M.D. Computing,* Vol. 15:4, July/August, pp. 221231.

Steele jr., G. L., et al. (1983) *The Hacker's Dictionary,* Colophon Books, New York.

Yates, J. and Orlikowski, W. J. (1992) Genres of Organizational Communication: A Structural Approach to Studying Communication and Media, *Academy of Management Review,* Vol. 17:2, April, pp. 299-326.