Agentization in Computing: How to Ameliorate the Consequences of the History Today?

Kai K. Kimppa[1], Janne Lahtiranta[2] and Markku I. Nurminen[3]

[1] Turku Centre for Computer Science (TUCS), University of Turku,; kai.kimppa@it.utu.fi
[2] Turku Centre for Computer Science (TUCS), University of Turku; janne.lahtiranta@it.utu.fi
[3] Turku Centre for Computer Science (TUCS), University of Turku; markku.nurminen@it.utu.fi

Abstract: In this article, we proceed by pointing out some significant events in the history of information systems that have contributed to the phenomenon which causes users to experience computerized systems as agents. Some issues discussed in relation to the phenomenon are the creation of master files (all data of one object class was collected together) and the use of integrated databases (multiple master files were integrated to an integrated conceptual schema). The increasingly intertwined functions of storing, processing, and transmission confused the picture further. Finally, we try to trace the reason to this tendency to animate or anthropomorphize information systems. A review of textbooks through historic periods is used to get support or counter-arguments to this hypothesis. We will also look into agentization and unintended subjectification of computer artifacts, and consider whether they have an impact on today's concept of the computer as an agent.

Keywords: Agentization, history of computing, information systems design, software agents

1. Introduction

A consistent look into the history of agentization in computing is missing and people do not understand its effects on the use of computer artifacts. One reason for this is that the concept itself is a rather vague one; it applies notions such as autonomy, reactivity, and social ability to animate or inanimate objects alike. Throughout the history of humanity, people considered living beings and even natural phenomena as agents of one sort or another. Agentization in computing is a more modern phenomena and it is present, amongst others, in popular media and research (e.g. when computers were referred to using terms like 'electronic brains'), and in unintended subjectification of information systems (e.g. when the user perceives that the computer has a will of its own).

Were it so that we could create life into inanimate objects, would they become agents? The stories such as that of a Golem built by Rabbi Judah Loew (a.k.a. Löw) from the mud of the Vltava River [7, pp. 119-121 and 10, pp. 205-206] and Mary Shelley's Frankenstein [10, p. 206] depict a situation in which a humanoid construct is brought to life as a mocking figure of a human. Stories like this lay a

basis for later stories of robots, such as those written by Isaac Asimov (1920-1992) where he describes a very human-like robots and the three Laws of Robotics or Philip K. Dick's famous novel 'Do Androids Dream of Electric Sheep?' (1968), which was later filmed as 'Blade Runner' (1982). Similar stories have been written of artificial intelligences (AIs) becoming sentient, e.g. by Arthur C. Clarke '2001: A Space Odyssey' (1968) in which the computer 'HAL' becomes sentient.

Mainstream media has also built this image (e.g. 'electronic brains') as the popular press anthropomorphized the computer already in the 1940s [7, p. 121]. As presented in the Figure 1, there has also been an anthropomorphized image of a computer wearing a navy captain's hat on the cover of the Time Magazine in January 1950 and choosing a PC as the "Man of the Year" (or Machine in this particular case instead of a traditional Man of the Year) in January 1983 [2, p. 1]. This raised questions such as '[c]ould a computer be smarter than Man?' [7, p. 121]. Partly this kind of popular publicity can be explained as efforts of marketing or gaining acceptance. It is, however an expression of a deeper cultural meaning assigned to computers [11].

Figure 1. Time Magazine covers from January 23, 1950 and January 3, 1983.

The vision in the field of AI was at the time, that it 'would soon rival the human intellect in many areas' [2, p. 213]. Although there were critics, such as J.C.R. Licklider, who thought that the view was utopian (ibid.), programs such as the famous Eliza [13] were created. In anticipation of actual AIs, Alan Turing proposed the Turing's test: if a person cannot tell whether they are talking with a machine or not, then the machine is intelligent [7, p. 122]. As Levinson [10, p. 209] pointed out, Eliza could only fool humans for a limited amount of time that it was another human (as many of us have undoubtedly experienced).

2. Nordic Information Systems Research and Agentization

The Nordic information systems (IS) research and development has throughout its history called for critical and user centered approach for IS development [1]. When we look into the agentization of computer artifacts, we should consider whether the users have been sufficiently involved in the design of the artifacts they use. It can be argued, that had they been involved in the design process, the inner workings or at least the fundamental operation principles of the artifacts would be evident to the homogenous majority of the users (i.e. to users who come from similar background and do similar work under similar circumstances). Since this is not the case, the users start to experience the software 'doing things on its own' instead of responding to their needs and wants.

The fundamental problem with this kind of view to agentization as a design flaw (of a sort) is that considering the modern computer artifacts, such as operating systems and office applications, it is nearly impossible to identify any kind of homogenous user groups or majorities. Furthermore, complexity and user interface techniques of the artifacts create additional layer of complexity into the mix. It may be extremely difficult to unearth the operating principles of an artifact; they are buried below complex, and sometimes even extremely anthropomorphic, or human-like, user interfaces. However, it is in the spirit of the Nordic IS research to questions whether this needs to be and what could one do to ameliorate the situation – if it needs to be ameliorated.

3. Control and Computer Artifacts

People employ computers when they perform their work tasks by means of information systems. The better control they have over the systems, the less need they have to regard them as agents rather than tools. For an individual user, such control eroded not only by the seemingly 'intelligent' behavior of the computer. The loss of control is further amplified by the complicated structure of the IS, and its semantics and pragmatics. These factors have supported the tendencies to regard the IS as a subject; such tendencies go back to the beginning of the computer era, or even to the Hollerith technology.

Our presumption in this article is that agentization moves the control of the system from the human to a third party. This move can be the information system itself (although in this case the 'control' is illusory), or the complex system can be used (either intentionally or unawares) to lessen the feel of security and control of the workers over their working situation. One good test to find this phenomenon is to ask the users to check whether the dubious information they received from the system is correct. If they can identify the other users that are responsible for it, the IS remains as a tool, but if they ask help from the system operators (who cannot have a semantic and pragmatic touch to the information), the system has grown to a 'subject'.

These cases hint towards a more profound problem where end users even today consider the computer as an active agent, which they do not feel in control

of in their activities either at work or at home. This is something that needs consideration in the design of information systems even today. One possible explanation could be that information systems have properties that have resemblance with actors simply because it has been the dominant paradigmatic notion among designers.

One of the reasons why agentization of computer artifacts is a particularly present problem in Nordic countries is the early adaptation of information technology in these countries. The situation is becoming even more relevant now in the advent of ubiquitous and pervasive computing. As by its definition, ubiquitous computing is always present without being directly visible. Ubiquitous computing can take away control from the user without the user even noticing the shift of power from the user to the system. Pervasive systems, again, as per the definition, pervades the living and working environments of the users making the world 'easier to handle' and 'easier to access'. At the same time, pervasive systems and embedding computing into the environment leave less and less room for the control of the system to the user, e.g. by making the system invisible to the user – what you do not know is there, you cannot control.

In the European Union (EU) at large and in Finland, policies regarding information and communication technologies (ICT) emphasize that quick adoption of ICT technologies is valuable per se, and to shorten the 'gap' between the EU countries and leading ICT provider countries it is paramount to 'be at the leading edge'. Unfortunately, this reasoning is problematic. Firstly, it promotes an idea that technologies in themselves would be valuable (which they are not) and secondly, it gives an impression that it does not matter how they close the 'gap' between the leading countries. Issues such as having a technology that is transparent to the user are forgotten and information systems are given an ever-greater access to control in our (private) lives.

As explained by von Neuman, what makes software so efficient is that we can leave it to complete automatically a given sequence of commands. For example, if in a web store we program a software 'agent' to execute buy and sell commands upon meeting certain conditions, we know for whom the system and the agent are working. However, when the software applications or 'agents' become more complex and ubiquitous, the actual human actor can disappear. For example, an anthropomorphic robot designed for home care of the elderly people such as the Pearl [6], can replace some of the functions of home care personnel and it 'does' different automated tasks, but for whom? If software 'agents' become extremely complex and they are left to execute series of orders, similar effects arise as those described in the following case from Sherry Turkle's book "The Second Self: Computers and the Human Spirit" (1984). In the first part of the book (Chapter 1): "Child Philosophers: Are Smart Machines Alive?" where the children ponder whether the computer 'dies' when it is turned off and especially why does the computer 'do' the things it 'does' – because 'it wants to' or because 'it is told to', and especially by whom is it told to 'do' these things. [12]

The question whether the computers are beyond our control is also a relevant problem. Even their programmers do not understand the complex programs of today. The reasons for this can be numerous such as lack of time available for

familiarizing with the source code and the number of programmers working with the application. Some of the systems today are so complex that even though one has sufficient time to study the inner workings of the system, no one can really understand the big picture. Ironically, Langefors [9] already used the concept "imperceivable system" as a cornerstone of his theoretical analysis. The complexity of the systems, however, does not need to be a major reason behind agentization of the software. As long as the user feels to be in control of the system, the operation logic of the system is visible, the user can at least understand the causality behind the operation logic. The user can anticipate certain results and fulfill one's expectations – or at least the user understands why they are not fulfilled (if that is the case).

Computer programs can be complex and we cannot understand them for other reasons as well. One such reason is that all programs must go through a compiler transition and they end up as binary, as a collection of zeroes and ones. The compiler itself is a program that translates programs – and depending on how we implement the program and the compiler, the results may wary. A more relevant problem however is the learning programs. Who can be responsible of the results of the 'actions' a learning program does. Transparency is one solution for this dilemma; even though 'actions' of the program are difficult to predict, we can explain the logic behind the operation.

The agent metaphor itself, and uncritical use of it, are problematic. We often see 'agents' doing searches without giving a second thought to the activity. The use of the term might propel us into new and stimulating directions, but if we do not exit the metaphor from time to time and look into the usage critically, we may become its prisoners instead of it aiding us in the development of helpful software.

We humans seem to have a tendency to anthropomorphize different things such as software, animals, and even natural phenomena. In computing, this may lead to excess agentization of programs. Trying to simplify complex issue by using a metaphor may actually distort the concept itself to something else altogether, thus not explaining the actual phenomenon at all. Agentization, at least in part, may result from misunderstanding or it may be just a defense mechanism of a human mind. It seems natural to humans to do this. Starting from simple animism of things such as lightning being 'a spear from Zeus', the human mind finds within it a simple 'analogy' in real life as a parent or leader physically reprimanding a child or subordinate to today's agentization of complex software. The phenomenon of lightning must have been quite as complex to understand for the early man as the information systems of today are to a typical user. To avoid this, we might also want to consider actually trying to explain the working of software to the best of our ability instead of using analogies or agentizing it. Of course, the explanations can be too cumbersome for everyday use, but we must not forget them, lest we anthropomorphize unnecessarily.

What is a software agent? Is an actual software agent even possible? In AI research, it would seem that at least the aim is to create an actual agent. The current 'agents' however hardly qualify. Commonly demonstrated examples of 'AI' include autonomous consumer products such as the automated lawnmower

(the Robomow[1]) or vacuum cleaner (the Trilobite[2]), which are rather similar in behavior. A real agent chooses; more specifically, it can choose differently. If element of choice does not exist, a real agency cannot either. Maybe we should talk about the anti-agentization of users. When the users do not question the 'decisions' the system 'makes' but rather attribute to a 'choice' to the system – "it did that" – both the 'end users' and the operators diminish in their capacity as decision makers and become unwitting objects for the system. The users actually degrade to the level of a machine themselves by doing things they are told to do even though they do not know the reason, only because the system told them. A similar situation holds for the operators of the system, where suddenly it is the system that 'needs' or 'wants' things instead of operators. The various issues introduced in this chapter have created a fertile ground for the development of information systems which made the actual agent disappear for the user.

4. The Invisible User

The argumentation around artificial 'intelligence' sometimes dominates the discussion around agentization. Therefore, the focus is in the behavior of the computer: Can the computer outperform human actors' abilities or capabilities? To play chess or to proofread a document can give an example of this. In most cases, the more or less 'intelligent' behavior is an outcome of deliberate purposeful objective and effort to create such an artifact. There is, however, another path to go towards ICT artifacts with agent-like properties. We seldom plan these paths consciously. What happens is that human actors gradually disappear from the sight and finally nobody seems to be in control of these artifacts.

This development is often due to the increasing complexity of information systems. The issue is not only in programs or in algorithms that may have characteristics with resemblance of intelligent behavior. We excessively use information technology for performing other functions as well. Most important functions on the side of processing are storing and communication. In what follows, we analyze the three main functions in the time perspective. We identify important events in the history of ICT and we highlight their possible contributions to making the human actor invisible.

During the batch-processing era, the agents and their roles were distinct. We delivered batch runs to the computer centers, where the operator performed activities in predetermined sequence. The operator could report about the status of different jobs, and no process executed unless the operator was in charge of it.

The situation changed slightly with the advent of multiprocessing and time-sharing. One processor could divide its capacity between different jobs by giving each of them a slice of time in their turns. First occurrences of this were background jobs such as pronging that occurred during such slices when the

[1] http://www.friendlyrobotics.com/

[2] http://trilobite.electrolux.se/

processor was less busy. In a time-sharing environment, a program could be loaded in the memory, ready for action but idle, like sleeping until it awoke by a particular trigger. Current graphical computer interfaces are essentially based in such waiting loops. When a program is running, in some way it has a life of its own, creating an illusion of an agent. Since the number of processors in our artifacts has increased dramatically since the first time-sharing operating systems, people are likely to expose themselves to this illusion more and more easily.

In storing, the illusion of agentization does not reflect intelligent behavior in such extent as in processing. Rather it highlights the role of the knowing agent. In the era of batch processing, the dominating sequential files did easily associate with a knowing subject. In fact, manual card files were better suited for such repository of a search date than the files on magnetic tapes or even on punched cards. Such a file was directly coupled to some application and the connection was pragmatically established. Gradually master-files replaced such application-specific files. The master-files collected all attributes of a certain object class to one collected file even if not all those data items were useful in any single application. We strengthen this change when the database concept integrated the conceptual schema between different object classes. Direct access and continuous availability of the database finally created a unit, which we could regard as a knowing subject for inquiry. On the other hand, it was difficult to identify and find human actors who stood and could take the responsibility for the semantics and pragmatics of various segments in the database. For some reason, this has not been a central design issue in information systems development.

Another problem that quite often confuses the users of a centralized information system is the dualistic nature of data storages. They are storages that document states and events from one point of time to another. This is the archiving function. However, in addition, they also use stored data. When one user writes a message and another user is reading it, it creates a communication link between them. Therefore, we can interpret an integrated information system as a communication network. However, such a network is nontransparent, due to three reasons:

- The volume and scope of a database are so large that the structure is unperceivable;
- The receiver often delays or triggers the message delivery, which mingles the archiving and communicating function within each other;
- The processing function is also involved. The receiver may get a message that was not entered by anyone. Hence, the message may be a report summarized by a piece of software.

All these three factors together indicate that it, indeed, is difficult to reconstruct the human agency structure embedded in an integrated information system. It is just natural that the users in most cases give explanations like “I received this information form the database” or “I have to enter this number because the system wants me to do it”.

Many forms of electronic communication keep the user visible; in fact, they make the agentization concern even worse. For example, e-mail is rather visible

and transparent form of communication. Even unwanted (junk mail) or anonymous senders of e-mail support the actor nature of the ICT artifact. On the other hand, most electronic services are based on the absence of the service provider. The service provider has made and started a program, an agent, which can respond meaningfully to the requests of the customer. However, electronic services must maintain the connection to human actors, because otherwise it would be very problematic to recover from errors and exceptional situations.

The concern for invisible user and disappearance of responsible human actors is not only a theoretical or conceptual exercise. It leaves the user without help, alone, when he runs to the problem of an unusual or exceptional situation, without practically any chance to check the origin of the problem or the consequences of alternative options to deal with it. Some authors have even argued that this nameless character of technology lends itself very well as a means of technical control. This is what happened with assembly lines during the first half of the 20th century.

"Struggle between workers and bosses over the transformation of labor power into labor was no longer a simple and direct *personal* confrontation; now the conflict was mediated by the production technology itself." [3, p. 118]

"Control becomes truly structural, embedded in that hoary old mystification, technology." [3, p. 125]

The increased use of computers as a means of control created strong resistance in Scandinavian countries. One slogan was: "We refuse to be detail-controlled", "Vi vägrar låta detaljstyra oss!" [4]. Furthermore, the defense of professional skills was important. Later, among many others, they established a Nordic Research project UTOPIA (1981-1985) to protect the maintaining and development of the skills of graphical workers. This time the frontline dealt with the whole profession, but the Scandinavian spirit was still clear: "To become masters of the machine", "At bli maskinens herrar" [5].

5. Classes of Agency

The real agency can disappear in various ways into the system. Below we list the ways we can transfer agency to the system, based on the previous analysis.

- *Strong agency*: Makes genuine choices;
- *Weak agency*: The actor executes a program but cannot check or correct the outcome;
- *Lack of agency*: Everybody knows that the computer is not a real agent; no one is able to identify the real agent for different actions;
- *Coordinative agency*: Coordinates actions, but does not perform them; another formulation of the invisible hand (or the boss):
  - Edwards' Control policies
  - Personal (by capitalist)
  - Bureaucratic (by foremen by rules)
  - Technical (nameless and "objective": we all have to obey the System)

- *Replaced agency*: Imitation of the human actor's behaviour (e.g. modeling of their feelings);
- *Agent without responsibility*: Expert systems; eroded responsibility.

In addition, the following constitutes a set of different relevant acts:

- *Deliberate*: Not accidental;
- *Causative*: Aiming at a goal that may be different from the present;
- *Intentional*: Causation may be nondeterministic, not all shots hit;
- *Competent*: Knowledgeable (Does not try to fly aircraft without adequate training);
- *Motivation*: Purpose, benefit, duty (Why should the agent do it);
- *Consequences*: Expected;
- *Context*: Collaboration, coordination;
- *Responsibility*: Accountability.

We combine these ideas in Table 1.

Table 1. Relationships between agency and action[3]

Act \ Agency	Strong Agency	Weak Agency	Lack of Agency	Coordinative Agency	Replaced Agency	Agent without responsibility
Deliberate	Yes	Yes	Yes		No	No
Causative	Yes	Yes	Yes		No	Maybe
Intentional	Yes	No	No		No	Maybe
Competent	Yes	Maybe	Maybe	Maybe	Maybe	Maybe
Motivation	Yes	Maybe	Maybe	Yes	No	Maybe
Expected consequences	Yes	Yes	Maybe	Maybe	No	No
Context	Yes	Yes	Yes	Yes	Maybe	Maybe
Responsibility	Yes	No	No	Maybe	No	No

As can be seen, strong agent can perform any actions listed. At this time, a strong agent would be a human being, although it is – at least to a degree – imaginable that a system might eventually hold the same status. Whether that is desirable, is outside the scope of this paper. The typical situation is that of weak agency, e.g. in a web store where the software performs certain tasks instead of the actual human actor. The weak actor is never responsible for its actions nor can it perform intentional choices – it cannot decide that it does not want to perform an order given to it by the actual actor. Lack of agency is the same situation as weak agency – except that the actor behind the agent is unknown. This causes

[3] Please note: many of the agencies can overlap each other.

problems in many situations that can lead to agents without responsibility or even to a replaced agency. We can use coordinative agency on purpose or it can 'just happen'; it moves the agency via a bureaucratic or technical group from the users or leaders of an organization to e.g. an ERP system. Thus, the coordinative agency can function as if there was a lack of agency when there in fact typically is not. The three empty slots in the table (table 1) may be nonsensical as questions for a coordinative agency.

The situation becomes truly problematic when there is an agent without responsibility or when one replaces the agency. When we cannot identify the entity (e.g. a medical expert system) as a system in the first place [see e.g. 8] or we cannot identify the agent behind the system, who – or what – is the responsible party? Is the question of 'what' being the responsible party even sensible? How could a system be accountable for something? Thus, the agent can – and does – function without anyone in charge. It is easy to give examples when this would be problematic (e.g. a ubiquitous system that transfers the 'responsibility' of the wellbeing of an elderly person). We now replace the agency, previously held by home care personnel, by the system; yet, the system is not responsible for anything.

6. Conclusions

To counter phenomenon of agentization and anthropomorphism, we need to take a critical view towards the new, especially ubiquitous software. The user needs to make decisions and must be in control. The users must decide what the system does and what the users want to do themselves. This, however, is becoming ever more difficult. A typical example is a ubiquitous home care system. The system collects information and 'acts' in the every day environment, but where do the commands come from? Who collects the information supplied by the system? For what purpose does one use that information? Presumably, the commands come from the home care or health care personnel and the information is used by them for the benefit of the user. Nevertheless, if the users are unaware of the usage they cannot control the 'actions' of the system or the use of the information. The system has become a semi-invisible tool for power over the user—a system 'out of control'.

References

[1] Bjerknes, G., Ehn, P. & Kyng, M. (editors) *Computers and Democracy*, a Scandinavian Challenge. Aldershot: Avebury, 1987.

[2] Campbell, M., Aspray, K & Aspray, W. Computer: *A History of the Information Machine*, New York: Basic Books, 1996.

[3] Edwards, R. Contested Terrain: *The Transformation of the Workplace in the Twentieth Century*. New York: Basic, 1979.

[4] Ehn, P., Erlander, B. & Karlsson, R. *Vi vägrar låta detaljstyra oss!*, (We refuse to be detail-controlled.) Rapport från samarbetet mellan Statsanställdas Förbund, avdelning 1050 och DEMOS - projektet, Stockholm, Arbetslivscentrum, 1978.
[5] Ekdahl, L. *At bli maskinens herrar* [To become masters of the machine]. UTOPIA report no. 19. Stockholm: Arbetslivscentrum, 1984.
[6] Jajeh, P. Robot nurse escorts and schmoozes the elderly. August 24, 2004, online at: http://www.thematuremarket.com/SeniorStrategic/Robot_nurse_escorts_schmoozesthe_elderly-5260-5.html accessed 13.04.2007.
[7] Kennedy, N. *The Industrialization of Intelligence: Mind and Machine in the Modern Age*, London: Unwin Hyman Limited, 1989.
[8] Lahtiranta, J. & Kimppa, K. Telemedicine and responsibility: why anthropomorphism and consent issues muddle the picture. 5th International WE-B (Working for e-Business) Conference, Fremantle, Australia, 2004.
[9] Langefors, B. *Theoretical Analysis of Information Systems*. Student Litteratur, Lund, 1970.
[10] Levinson, P. *The soft edge: a natural history and future of the information revolution*, Routledge, London, 1997.
[11] Suominen, J. *Sähköaivo sinuiksi, tietokone tutuksi,* [Getting familiar with the electric brain, getting to know the computer]. Jyväskylä 2000.
[12] Turkle, S. *The Second Self: Computers and the Human Spirit*, London: Granada Publishing Limited 1984.
[13] Weizenbaum, J. *Computer Power and Human Reason*, San Francisco: Freeman, 1976.