

Software Agents and User Autonomy

Baty Friedman

*Mathematics and Computer
Science
Colby College
Waterville, ME 04901, USA
+1 207 872 3572
b_friedm@colby.edu*

Helen Nissenbaum

*University Center for Human Values
Marx Hall
Princeton University
Princeton, NJ 08544, USA
+1 609 258 2879
helen@phoenix.princeton.edu*

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of ACM. To copy otherwise, or to republish, requires a fee and/or specific permission. Agents '97 Conference Proceedings, copyright 1997 ACM.

ABSTRACT: Software agents comprise a new area for research and soon will be embedded and ubiquitous in modern computing systems. In this formative phase, it is important to develop comprehensive measures of excellence for evaluation. To criteria in the literature -- competence, completeness, correctness, efficiency, reliability, and trust -- we propose as an essential complement: user autonomy. In this paper we: (1) provide a definition and rationale for user autonomy in relation to software agents; (2) describe a framework for understanding how user autonomy can be promoted or undermined through the design of software agents; and (3) consider justifiable limitations on user autonomy.

KEYWORDS: Agents, autonomy, ethics, social computing, software agents, social impact, standards, user-centered design.

INTRODUCTION

Some critics of technology contend that a small, privileged group of individuals use technological advances to diminish the autonomy of others. This consequence is surely possible. Yet it can be otherwise insofar as those who fund, design, and build technology take into account effects on human autonomy. Thus, in this paper we: (1) provide a definition and rationale for user autonomy in relation to software agents; (2) describe a framework for understanding how user autonomy can be promoted or undermined through the design of software agents; and (3) consider justifiable limitations on user autonomy.

THE CRITERION OF USER AUTONOMY

Good software agents must measure up against a set of criteria for excellence. Some of these criteria are drawn from a larger set that measures the quality of software in general -- criteria such as correctness, completeness, efficiency, reliability, and safety (see [9]). Other criteria, like competence and trust, are particularly relevant to agents [6, 7]. Elsewhere, we have argued that this set of

criteria should be expanded to include those of social and ethical significance [1, 2]. Our focus on user autonomy extends this line of investigation.

By autonomous individuals we mean individuals who are self-determining: they construct their own goals and values, and are able to decide, plan, and act in ways they believe will help to achieve their goals and promote their values. Autonomy is fundamental to human flourishing and self-development [4, 5]. When people are denied autonomy, such as slaves or indentured laborers, we judge them to have been morally violated. They are prevented from realizing themselves, not respected as human beings, and treated as means to someone else's ends. Moreover, those who are not treated and viewed as autonomous individuals cannot be held morally responsible for their actions.

To recognize that user autonomy is a separable criterion from those noted above, consider the widely used metaphor of the personal assistant. A person takes on an assistant to perform work that he would like, or needs to have done and, for whatever reasons, chooses not to do himself. Perhaps the work is tedious, difficult, dangerous, or time-consuming. The assistant acts for, and on behalf of, the person. Ideally the assistant satisfies the above set of criteria of excellence: he is trustworthy, competent, reliable, and so forth; but is this always enough?

We think not. To convey our concerns, consider three cases where user autonomy is undermined.

Case 1: In Harold Pinter's screenplay for the movie *The Servant*, an effete, young, English gentleman hires a butler. The butler is as competent and reliable as the gentleman is frivolous and irresponsible. The butler solicitously does the master's bidding until the relationship begins to take a strange and sinister turn. The gentleman, though master, becomes increasingly dependent on the servant. The servant, who continues to fulfill the master's desires becomes more and more domineering in defining just what these desires ought to be. By the end of the film there has been a strange turnaround with the master as servant and servant as master. This screenplay illustrates that by engaging with a personal assistant, one's goals can unknowingly come to be reshaped, if not twisted. The important point here is that the change can occur unknowingly and, thus, diminish the charting of one's own goals and the direction of one's life.

Case 2: It is understood that people usually know their own goals better than others, even if their self-knowledge may be imperfect. Thus, to promote autonomy one allows people to seek advice from others, but ultimately leaves the decision-making (even if it is the decision to delegate decision-making) up to the individual. For example, imagine that a woman with cancer seeks advice from her physician. The physician, focusing on biological concerns to sustain life, recommends an intensive chemotherapy program that will likely extend her life for another 4-5 years. Forgoing treatment, the physician predicts about one year of further life, largely painfree. From the woman's perspective, it is plausible that psychological concerns with quality of life outweigh length of life. In other words, while the medical professional recommends chemotherapy, the autonomous patient declines because she is best privy to and acts on her own values and goals. Had the physician the authority to mandate the chemotherapy, the patient's autonomy

would have been undermined.

Case 3: Autonomy is centrally concerned with self-determination -- making one's own decisions, even if those decisions are sometimes wrong. It is this aspect of decision-making that allows us to be responsible for the consequences of our actions. Continuing with the example of the patient and physician above, let us imagine that after some time the patient realizes that she had misjudged her own goals and values -- that, indeed, for her longevity would have outweighed quality of life. As an autonomous individual, her unfortunate decision was not only hers to make, but responsibility for the decision rests with her.

In the above three cases, the personal assistant is another human being. Yet, we think that the relationship of a user to digital agents can be distinguished along similar lines, namely, those that preserve, or even enhance user autonomy, and those that do not. The examples show that user autonomy may be distilled as a distinct characteristic. Its preservation should be addressed explicitly through design and not simply left to chance.

Moreover, when the personal assistant is a technological tool two considerations make the case for user autonomy even more pressing. First, in contrast to a human personal assistant with whom it is possible to negotiate (and re-negotiate) so that autonomy which may have been undermined can to some extent be regained, such negotiation with a software agent is currently unlikely, if not impossible. Second, in the case of human personal assistants, any loss of autonomy while important to the individual affects only that individual; in the case of widely disseminated software agents the loss of autonomy can be systemic and pervasive. Thus, it becomes important to move toward a framework for understanding how user autonomy can be promoted or undermined through the design of software agents.

ASPECTS OF SOFTWARE AGENTS THAT CAN PROMOTE OR UNDERMINE USER AUTONOMY

It might seem that designers could maximize user autonomy by following the simple dictum that more control leads to more user autonomy. After all, if autonomous individuals need to have freedom to choose ends and means, then it could be said that wherever possible and at all levels, designers should provide users the greatest possible control over computing power. On closer scrutiny, however, we see a more complex picture and a less direct relationship.

User autonomy seems to have less to do with simply the degree of control and more to do with what aspects of an agent are controllable, and the user's conception and knowledge of the agent (cf. [8]). In the case of a software agent to help with creating consistent and well formatted documents, for example, a typical non-technically minded user will have little interest in explicitly controlling lower levels of operation of the agent even though they will appreciate control over higher level functions -- little interest, say, in controlling how the agent executes a cut and paste operation or embeds formatting commands in the document, and more interest in controlling the efficient and effective formatting of the document. In this case achieving the higher order desires and goals, such as efficiently producing a good-looking document, will enhance autonomy, whereas excessive control over all

levels of operation of the agent may actually interfere with user autonomy by obstructing a user's ability to achieve desired goals. In other words, autonomy is protected where users are given control over the "right things at the right time." Of course, the hard work of design is to decide just what and when these are.

Toward this end, we identify five aspects of software agents that can promote or undermine user autonomy.

1. Agent Capability

User autonomy can be undermined when the software agent does not provide the user with the necessary technological capability to realize his or her goals. Said differently, user autonomy can be undermined when there are states the user desires to reach but no path exists through the use of the software agent to reach those states. To illustrate this dimension, consider a mail agent. A reasonable user request for such an agent might be, "I don't want to see any mail from Martin unless it's really important." This is the sort of thing people routinely request of their human personal assistants. But technically it may not be possible to specify this level of intention to a software agent. For an end-user programmable agent, such as CyberDog, it can be easy to program the agent to "trash all mail from Martin." And fairly simple to add the qualifier "except when the subject heading says 'Important!.'" But what if the subject heading was to say "READ ME!!!"? Or "HELP!?" Or any number of other headings that a human personal assistant would key into as meaning something special that needs attention? True the user could add any number of additional rules to improve CyberDog's performance, but there is no way to anticipate, let alone logically specify all the possibilities in advance. Our point is this: A lack of technical capability on the part of the software agent -- to be able to accurately represent the user's intentions -- can lead to a loss of autonomy for the user.

2. Agent Complexity

In some instances, software agents may supply users with the necessary capability to realize their goals, but such realization in effect becomes impossible because of complexity. That is, a path exists to the state the user desires to reach but negotiating that path is too difficult for the user. Pattie Maes [6] points to this problem in her critique of the end-user programmable approach to software agents. She writes: "The approach requires too much insight, understanding and effort from the end-user, since the user has to recognize the opportunity for employing an agent, take the initiative to create an agent, endow the agent with explicit knowledge (specifying this knowledge in an abstract language), and maintain the agent's rules over time (as work habits or interests change)" (p. 32). Similar problems for user autonomy arise with trainable software agents in instances that require extensive and complex training to achieve good performance. In effect, from the user's perspective, these software agents are untrainable. Thus problems of agent complexity arise from a mismatch between the user's abilities -- in terms, for example, of skill level, memory, attention span, computational ability, and physical ability -- and the user abilities assumed by the software agent to maximize its use.

3. Knowledge About the Agent

Sometimes, in order to use the services of an agent as desired, a user must know

how the agent goes about its task. When the designer of a software agent does not make information accessible to the user, then the user's autonomy can be undermined. For example, an intelligent search agent for a space like the World Wide Web (e.g., WebCrawler) can provide the user with important information but hide critical assumptions about how that information was collected and filtered. For example, a user might need to know what sites were contacted, how comprehensive was the search, and what search algorithm was used. Or consider a trainable agent. The user may have very good performance results but no knowledge of or access to the underlying rules and database that the agent uses to guide that performance. Imagine such a mail agent. If the user receives no mail from a particular individual for an extended period of time, there is no way -- save going outside the system to ask that individual directly -- for the user to discover if the agent has been trashing mail from that person or if that individual has simply not sent the user any mail for a long time.

To their credit, some researchers have explored "explanations" provided by the agent as a means to help the user understand how the agent has determined a course of action. Pattie Maes [6], for example, writes that:

the particular learning approach adopted allows the agent to give "explanations" for its reasoning and behavior in a language the user is familiar with, namely in terms of past examples similar to the current situation. For example, "I thought you might want to take this action because this situation is similar to this other situation we have experienced before, in which you also took this action" or "because assistant Y to person Z also performs tasks that way, and you and Z seem to share work habits." (pp. 32-33)

From the standpoint of user autonomy, the problem with this sort of explanation is that it hinges too much on an unelaborated notion of similarity. In other words, the user has no sense of the criteria being used to establish similarity.

4. Misrepresentation of the Agent

Users can also experience a loss of autonomy when provided with false or inaccurate information about the agent. Consider two diverse examples. First, imagine the package copy for a news filter that states "this news filter is as good as the best personal assistant." Given the state of the field, such hyperbole will mislead a user who believes the package copy and, thus, develops inaccurate expectations of the software agent's ability to realize the user's goals. Second, consider a chatterbot in a MUD that people have entered with the mutually understood goal of meeting and interacting with other visitors. If the chatterbot represents itself to the user as another person visiting the MUD, the user will be lead to engage socially as if with a person. Once discovered, the effect of such deception is to cast doubt on future interactions; such doubt, in turn, undermines the quality of those interactions and ultimately effects the user's original goals for visiting the MUD.

5. Agent Fluidity

Even if users' large overarching goals remain stable, smaller subsidiary goals are

likely to change. Thus, software agents need to be able to support and easily accommodate changes in users' goals. Consider this problem for user autonomy in light of software agents that provide filtering services. The filter works to prevent the user from retrieving or seeing information that the user does not want to review. Thus, there is a sense in which a filter acts as a censor. However, as the user's goals change over time, what was appropriately filtered out early on may become of interest over time. However, most software agents do not provide users with access to the information that was filtered out -- after all, that is what the user initially wanted removed from the user's purview -- but such information is critical if users are to recognize when their agents need to be reprogrammed or retrained to meet evolving goals. The point here is that user's goals will evolve; software agents need to take such evolution into account and provide ready mechanisms for users to review and fine-tune their agents as their goals change.

ACCEPTABLE LIMITS ON USER AUTONOMY

To argue that user autonomy is a valued criterion of software agents is not to argue that it is the paramount value. Just as when people pursue their own goals in acting autonomously they need to consider and frequently compromise with external factors, including the rights of others to achieve their ends, so might it be that a design will justifiably compromise user-autonomy in favor of other important criteria. The upshot, on these occasions, will be limiting control of a user over aspects of operation, even when this is at the expense of his or her autonomy.

By way of example the following two considerations might, at times, place limits on user autonomy. One consideration entails situations in which safety is at stake. When software agents are employed in contexts that involve human welfare, such as helping to coordinate efforts in an air traffic control system, we may need to restrict user autonomy to protect against a user with malicious intentions as well as well-intentioned users guided by poor judgment. A second consideration concerns standardization. Since user autonomy suggests greater flexibility and personalization, it may appear to run contrary to goals for standardization. On the one hand, some forms of standardization may restrict users in ways that are important to them in relation to their goals. On the other hand, standardization can free users from the burden of relearning how to work with an agent as they switch among systems (or switch among stations with the same system) and thereby, in effect, provide users with greater control over their systems. We suspect the two ideals of standardization and autonomy need not always conflict, but that good resolutions will depend on identifying the appropriate level of user control.

CONCLUSION

In the development of our technology, we continue -- and rightly so -- to delegate tasks to the technology. But, in so doing, we should be wary that through neglect we do not design technologies in general, and software agents in particular, which diminish user autonomy. Much work is needed to develop our understanding of user autonomy and to identify methods to aid with the design process. In our future work we hope to provide greater specificity in these directions. Here we but note that, as with other criteria such as reliability, efficiency, and correctness, we hold out the criterion of user autonomy as an ideal against which we judge the quality of our systems in use. To fall somewhat short of the ideal is within the realm of our practice; to fall too far short is to underestimate how all of us (as autonomous

individuals) can help shape the direction of our computing community.

ACKNOWLEDGMENTS

This work was funded in part by the Clare Boothe Luce Foundation and by a research grant from the Natural Science Division, Colby College. Aspects of the ideas presented in this paper were developed in a workshop on user autonomy at CHI 96 [3]. Thanks to workshop participants. Thanks are also extended to Research Assistants Mitchell Goodman and Brigette Krantz.

REFERENCES

1. Friedman, B. & Kahn, P. H., Jr. (1992). Human agency and responsible computing: Implications for computer system design. *Journal of Systems Software*, 17, 7-14.
2. Friedman, B., & Nissenbaum, H. (1996). Bias in computer systems. *ACM Transactions on Information Systems*, 14(3), 1-18.
3. Friedman, B., & Nissenbaum, H. (1996, April). User autonomy: Who should control what and when? Conference companion of the conference on Human Factors in Computing Systems, CHI 96 (p. 433). New York: Association for Computing Machinery
4. Gewirth, A. (1978). *Reason and morality*. Chicago: University of Chicago Press.
5. Hill, T. E., Jr. (1991). *Autonomy and self-respect*. Cambridge: Cambridge University Press.
6. Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, 37(7), 30-40, 146.
7. Maes, P. & Wexelblat, A. (1996, April). Interface agents. Conference proceedings for the conference on Human Factors in Computing Systems, CHI 96 (pp. 369-370). New York: Association for Computing Machinery.
8. Norman, D. A. (1994). How might people interact with agents. *Communications of the ACM*, 37(7), 68-71.
9. Riecken, D. (1994). Intelligent agents. *Communications of the ACM*, 37(7), 18-21.