# Towards Mixed-Initiative Access Control

Prasun Dewan
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599, USA
dewan@cs.unc.edu

Jonathan Grudin and Eric Horvitz
Microsoft Research
One Microsoft Way
Redmond, WA 98052
jgrudin@microsoft.com horvitz@microsoft.com

***Abstract*— The difficult task of providing access to shared objects is, typically, carried out individually by access authorizers. Using a presence-based "thought experiment" and an abstract architecture, we motivate and explain here the idea of using distributed collaborative environments to perform this activity. In these environments, the initiative in distributing access rights to shared objects can be taken by information guardians, information consumers, and tools that act as agents of the guardians and consumers. Information consumers are responsible for sending access requests to information guardians; their agents (partially or completely) automate this task for them. Information guardians are responsible for authorizing accesses; their agents automate this task for them.**

***Keywords- access authorization; information privacy models; mixed-initiative dialogues; role-based access control; context-specific presence; presence rights; right amplication; optimistic access control; interactive access control; mixed-initiative interaction.***

## I. Introduction

There are several components of security (Figure 1): authentication, which checks that users are who they claim to be; access control, which ensures that authenticated users to do not make unauthorized changes to information stored in a computer; cryptography, which guards information transmitted on the network; and intrusion detection, which protects a computer from attacks from other computers. This paper focuses on access control.

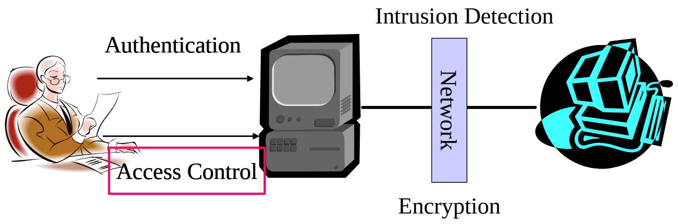


**Figure 1 Access Control vs. Computer Security**

Access control is related to privacy. Privacy is used to guard information about (a) users, such as their social security numbers and buying information, and (b) users' presence, that is, their activities on the computer, such as whether they are currently logged on. Access control is more general than privacy as it guards any information stored on a computer. The example we present focuses on presence-control, though the general idea we propose should apply to arbitrary forms of access control.

Access control has been a fundamental component of systems allowing multiple users to share resources. However, despite being researched extensively in the last three decades, existing access-control mechanisms continue to be difficult to use. In theory, access control is not difficult as mechanisms can be provided to guard any computer-stored information. Moreover, there is a correctness criterion for exercising these controls, called the "need to know" or "least privilege" principle: People should have access to only the information they need to know do their jobs. However, in practice, access control is a difficult problem because of the following situation we face today:

- Object-specific control: People wish to provide different controls for different objects. For example, most people are willing to share their work-telephone but not their social-security numbers with co-workers [1].
- User-specific controls: For many objects, such as "in-progress work," users show a high degree of variability in their privacy preferences [1].
- Task-based control: Users often care about the need to know principle [1, 2]. For example, co-PIs on NSF projects are willing to share their social security numbers with a PI only because Fastlane does not allow them to be made co-PIs without this information. However, it is difficult to follow this principle without knowing the tasks for which a right may be used. Traditional access control does not directly tie access distribution to tasks. Not surprising, then, an industrial survey found that one out of three people in an organization has wrong access rights [3].
- Obscure specification mechanisms: The reason for the wrong access rights might be due to a misunderstanding of not only the needs of users and but also the mechanisms used to distribute rights. There is anecdotal evidence to show that traditional access control is indeed hard to use. For instance, while the first author was on sabbatical at Microsoft, it was difficult for him to find and understand the access controls provided by Microsoft's SharePoint, a wide-area repository. He had to send a message to a large mailing group at Microsoft before he could set the desired permissions, and only one person responded with the correct answer. Moreover, he keeps a manual of AFS, a

- This research was funded in part by Microsoft and NSF grants ANI 0229998, EIA 03-03590, IIS 0312328, and IIS 0712794

popular file system, on his physical desk at UNC solely for the purpose of looking up the access control commands, which he can never remember. Similarly, whenever he teaches access control, he asks his students if they know about right inheritance in AFS and Windows, and the difference between (a) modify and write rights in Windows and (b) read and lookup rights in AFS. Few students understand these concepts. The situation for non-CS majors is bound to be worse.

Some of these problems conflict with each other. In particular, many users address the obscure-mechanism problem by using a coarse-grained access control policy in which they make all objects private except a special public directory, which is accessible to all authenticated users. When an object has to be shared, rather than worrying about how to set its permissions, they simply copy it to the public directory. This approach conflicts with their desire for object, user and task specific controls. For the "privacy unconcerned" [1], it results in increased security risks because some information (such as a joint proposal) not intended for everyone is stored in public directories. On the other hand, for the "privacy fundamentalists" [1], it reduces the chances for possible collaboration as certain collaboration-enabling data such as calendars are not shared with potential collaborators. Finally, it creates multiple copies of the same object, which can easily become inconsistent.

Existing systems-research on this topic has tried to address the usability problems by providing high-level languages to specify access rights that allow a single specification to give a large number of rights [4]. In this paper, we propose and motivate a radically different, but complementary, approach that is based on the following two observations (a) access-control is an inherently complex collaborative activity (carried out to support a more primary collaborative activity such as document editing) involving one or more information guardians and consumers, and (b) collaborative environments can make it easier to perform group tasks by automating some of these tasks and providing formal interactive channels for the collaborators to communicate with each other.

Traditional systems provide abstractions that assume that the difficult task of providing access to different users is carried out individually by people manually playing the role of access authorizers. Based on the two observations above, we propose here the idea of designing collaborative environments to ease the setting of both general and item-specific privileges. In these environments, the initiative in distributing access rights to shared objects can be taken by information guardians, information consumers, and tools that act as agents of the guardians and consumers. Hence, we refer to this form of access control as mixed-initiative, which is special case of the general idea of mixed-initiative interaction [23]. Information consumers are responsible for sending access requests to information guardians; their agents (partially or completely) automate this task for them. Information guardians are responsible for authorizing accesses; their agents automate this task for them. Figure 2 shows the difference between traditional and mixed-initiative access control. The box on the left contains the shared objects to be protected. With traditional access control, its guardians are individually responsible of distributing access to different parts of it, though they may use general-purpose, informal communication channels such as email and instant messaging to consult with others. With mixed-initiative access control, a special collaborative environment, shown on the right, is created for this task. The environment provides consumer and guardian agents, and offers explicit support for the agents to work with the humans. Our expectation is that the specialized controlling collaborative environment can address key impediments to the fluidity of access control, solving some long-standing security problems that have been exacerbated with the increased interest in distributed programmer-defined shared environments such as web portals.

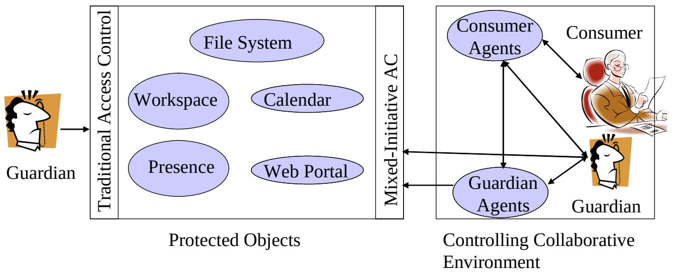


**Figure 2 Traditional vs. Mixed-Initiative Access Control**

Access control is needed to support collaborative systems – without collaboration there would no sharing and thus no need for protection. Put another way, collaboration is the problem addressed by access control. Mixed-initiative access control, interestingly, makes collaboration systems also part of the solution to problems with current access-control problems. Figure 3 shows this symbiotic relationship between collaborative systems and (mixed-initiative) access control.

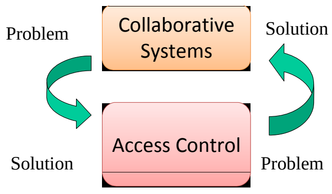


**Figure 3 The Symbiotic Relationship Between Collaborative Systems and Access Control**

Mixed-initiative access control is related to both general models of access control and specific applications developed recently.

## II. RELATED WORK

### A. *General models*

The classical matrix model, proposed by Lampson [5] and refined by Graham and Denning [6], provides a general framework for understanding and automating access control. The protection state of the system is represented by the abstraction of an access matrix, A. The columns of the matrix represent the protected objects, the rows the subjects (users/processes) from whom the objects are protected, and an entry, A(s,o), denotes the access rights subject s has over object o. As the access matrix is normally sparse, containing many entries with no rights, it is stored as a series of lists. Some systems store with each object an access control list, consisting of (subject, rights) pairs, while others store with each subject a capability (popularly known today as a cookie or token) list of (object, rights) pairs. In either case, the list does not contain entries that have empty rights. Even then, it is unacceptably time consuming to specify all the elements of a list [7]. Therefore, most modern systems such as Unix, AFS, and Windows allow one access-specification to be made for a group of access matrix cells. In particular, they support roles, that is, groups of users associated with access rights. By assigning a user to a role, it is possible to give him/her all rights associated with the role. Such grouping not only eases the specification task, but also makes it easy to understand how rights are distributed in an organization. Shen and Dewan [8, 9] have abstracted and generalized such grouping through extensions to the matrix model (Figure 4). In the extended model, the rows represent both subjects (e.g. Grudin) and subject-groups (e.g. AllUsers), the columns represent both objects (e.g. files) and object-groups (e.g. directories), and the cells contain both rights (e.g. Read) and right-groups (FullControl). When determining if subject s has right r to object o, the system first checks if A(s,o) contains r. If not, it determines if a predefined inference function, F(s,o,r,A) evaluates to true. F searches matrix entries corresponding to groups of subjects, objects and rights. To illustrate, assume that user Horvitz tries to access file Abstract in directory Mixed-Initiative, given the extended access matrix of Figure 4. He is allowed to make the access as AllUsers have been given Read access to all objects in the directory Mixed-Initiative. A problem with this approach is that the access matrix may now have inconsistent entries. Jajodia et al [10] have developed a model that supports multiple approaches to resolve this issue. Another problem is converting an existing access matrix without grouping to one with grouping. Recent work in role mining has addressed this problem for user-groups [11]. The original access matrix associated an entry with a single owner. Dewan and Shen have motivated and described a mechanism for supporting joint ownership [12]. Another important issue, required to support optimistic access control discussed below, is revocation of granted rights. In capability-based systems, indirect capabilities have been used to easily support this idea. Traditional systems based on access control lists have required a much more complex infrastructure [13] for revocation. For such systems, Dewan and Shen [12] describe a simpler mechanism based on indirect roles to revoke granted rights. Tolone et al [4] provide a survey of access control mechanisms in collaborative systems.

*Specific applications*: The vast majority of traditional and new collaborative systems such as file systems, Web portals, and conferencing systems implement various subsets of the general models described above. Some recent applications do have important novel ideas not included in the general models. In the context of distribution of protected physical (paper) documents, Stevens and Wulf [14] have identified a new dimension in access control: when is access control exercised? As in traditional systems, access definitions can be specified before the document request, by associating permissions with the documents. If the requested document does not have the required permissions, the request can be sent to the owner, who can then interactively authorize it at time of access. An alternative is to grant the access automatically under the optimistic assumption that this will cause no harm. The grant is, however, recorded, and can later be revoked. Legal arrangements ensure that the users do not make use of the paper documents taken back from them. This optimistic approach was first proposed by Povey [15], who gave several motivating scenarios for it. For example, "*Bob is a nurse at a small rural hospital which has been physically isolated due to a heavy storm. Communications are down, and the local doctor is unable to be located. Bob has to attend to a life-threatening emergency, for which he needs immediate access to a patient's medical records. However, Bob is not authorised to access the information, putting the patient's life at risk.*" The optimistic policy is supported in auto distribution lists. There is data to indicate that an optimistic policy may be applicable in many other situations. For instance, Palen and Grudin found that office workers' fear that public calendars would result in managers and other co-workers constantly snooping on them and spreading the information was unfounded because of social conventions [27, 28]. The popular Wikis take the even more optimistic approach of not providing any access control - they do not even authenticate users. Related ideas are seen in some other existing systems. Parental control allows children to access blocked Web pages that are authorized interactively by a parent. Bauer et al [25, 26] present and evaluate a system using interactive access control to authorize entry to a physical room with a networked smart lock. Instant messaging applications support interactive grant of dynamic reciprocal access: user A's request to send messages to user B is interactively granted by B, and results in B being granted the reciprocal right to send messages to A during the IM session. Yahoo buddy lists also support such reciprocal access – user A's request to add user B to A's buddy list is interactively granted by B, and results in A being added to B's buddy list. Ackerman and Cranor [16] provide two privacy critics or agents that aid the task of interactive web-based access control by warning users when they try to send data to web sites that (a) in another session they had blocked and (b) are known to send information to junk mailers.

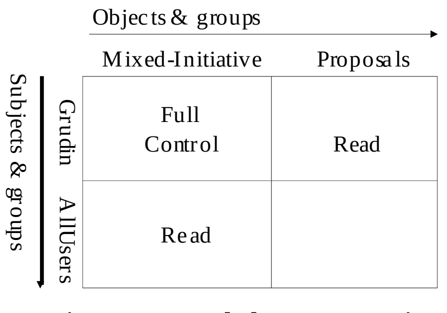


**Figure 4 Extended Access Matrix**

## III. ILLUSTRATING & MOTIVATING THOUGHT EXPERIMENT

As we see above, modern applications have several access-control features missing in the general models. It is a hypothesis of our work that many of these features can be applied to other domains including traditional file systems. In fact, we believe more strongly that all of them can be applied to all domains, and the choice depends on the nature of the collaboration and collaborators rather than the application. More important, we believe, the modern applications have not overcome many of the limitations of the traditional access control models. The impact of these limitations is exacerbated when we consider the new kinds of information that (a) are shared today, and (b) can be shared, but are not, because of access-control concerns. To illustrate (a) consider Facebook, which makes it possible for others to see our photos, friends list, status updates, the applications we have added, and when we execute these applications. Facebook provides traditional mechanisms for controlling access to most of this information, but anecdotal evidence shows that few users use these mechanisms, simply using the default settings, despite their desire for object-, user-, and task- based access control, mentioned earlier. To illustrate (b), consider the fact that it is not possible for others to see the nature of our editing activities without being in a joint editing session with us. The potential usefulness of such sharing will be illustrated below.

The intuition behind our proposed solution to this problem, given in the introduction, is that distributing access rights is inherently a complex collaborative activity that can be better supported by an interactive mixed-initiative collaborative environment in which information guardians, consumers, and agents participate in the access distribution process. Naturally, in-depth research of this idea is needed to determine the extent to which this belief is true. To motivate this research, we have created an example application that provides sharing of information about our editing activities. As the basis of the example, we took the collaboration involved in creating a research proposal on mixed-initiative access control involving the three authors, and considered what it would mean to provide mixed-initiative access control for this activity. The result of this "thought experiment" is the following hypothetical scenario, which serves to illustrate and motivate this new form of access control.

The proposal is stored in a Groove like workspace, which is essentially a wide-area directory associated with the set of users working on it (Figure 5). The day before the deadline, the three researchers are ironing out the last few wrinkles in the proposal documents. In particular, they are trying to ensure that their definitions are correct and consistent. Almost every change must be approved quickly by the others. Moreover, they must not make concurrent inconsistent changes. Therefore, they send each other notifications explaining what they have finished working on and what they are going to next work on. However, notifications create problems when users make further changes to pieces of work they had earlier declared to be "finished." When they don't send the notifications, they do not give their collaborators a chance to look at the final version, which is particularly unfortunate if the final change was made before the collaborators saw the last change mailed to them. When they do send others the notifications, their collaborators can be inundated with unwelcome messages.

When Jonathan realizes this problem, he decides to use a research tool that allows him to watch in real time the edits of Prasun (Figure 5, menu item). Such awareness has been found to be useful in many situations [17]. Prasun, however, has not given him rights to watch his actions, as he, like most people [1], is normally uncomfortable with others seeing his incomplete incremental changes. An agent running on his computer asks Jonathan if he would like to send Prasun a request for these rights. As the collaboration has now moved to a stage where the goal is to watch and approve each other's finishing touches, Jonathan sends the request along with a message explaining why the access was needed. After reading the message, Prasun grants the requested rights completely (Figure 6). He also had the choice of discussing (through instant messages), denying, reducing or amplifying the requested rights. For example he could have reduced the rights by allowing Jonathan to watch him for only a day. Conversely, he could have amplified them by allowing Jonathan to forward them to others.

The grant approval is intercepted automatically by Jonathan's watch tool, which responds by creating windows tracking Prasun's changes to the watched documents (left window, Figure 7). In addition, it creates a summary window (right window, Figure 7) indicating, among other things, which documents are being actively edited, a portion of text around the cursor of an active window, and whether Jonathan's accesses to a watched document are being audited, that is, put in a log that can be viewed later by Prasun.

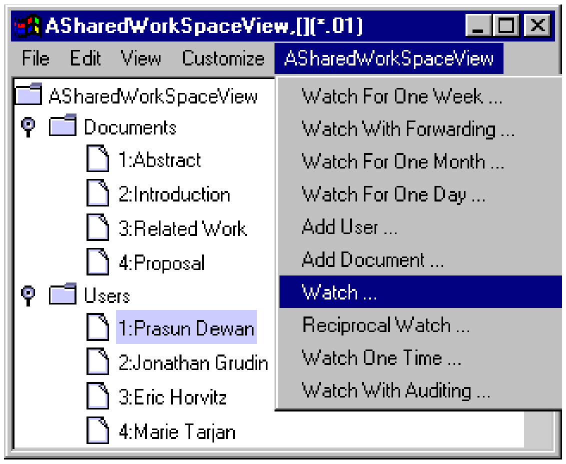
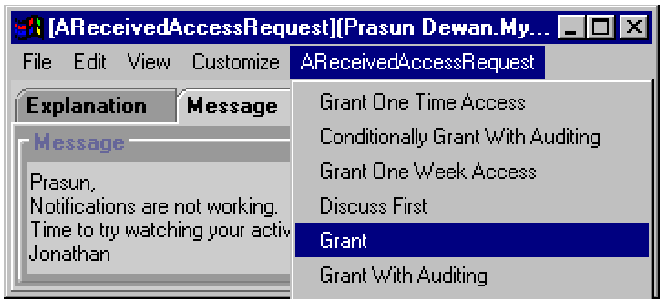
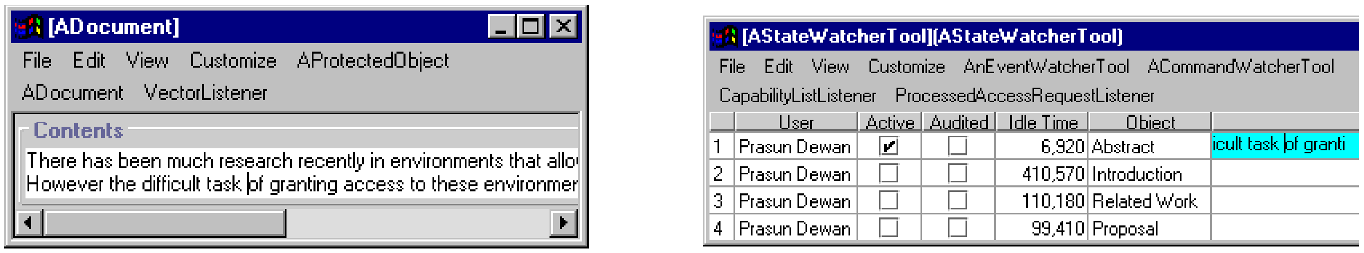


**Figure 5 Execution of unauthorized operation**



Prasun,
Notifications are not working.
Time to try watching your activ
Jonathan

**Figure 6 Manually processing a request**


[ADocument]
File Edit View Customize AProtectedObject
ADocument VectorListener
Contents
[AStateWatcherTool](AStateWatcherTool)
File Edit View Customize AnEventWatcherTool ACommandWatcherTool
CapabilityListListener ProcessedAccessRequestListener

There has been much research recently in environments that allo
However the difficult task of granting access to these environmer

| | User | Active | Audited | Idle Time | Object | |
|---|---|---|---|---|---|---|
| 1 | Prasun Dewan | ☑ | ☐ | 6,920 | Abstract | icult task of granti |
| 2 | Prasun Dewan | ☐ | ☐ | 410,570 | Introduction | |
| 3 | Prasun Dewan | ☐ | ☐ | 110,180 | Related Work | |
| 4 | Prasun Dewan | ☐ | ☐ | 99,410 | Proposal | |

**Figure 7 Consumer tool processes grant notification by re-issuing operation and watching Prasun**

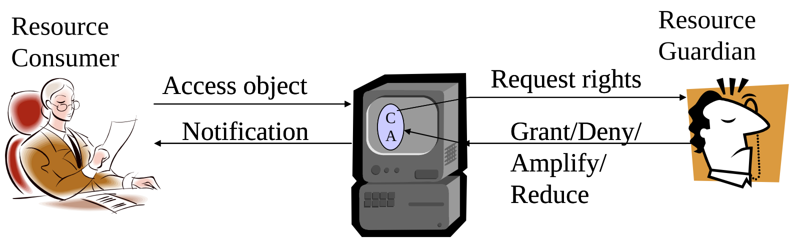
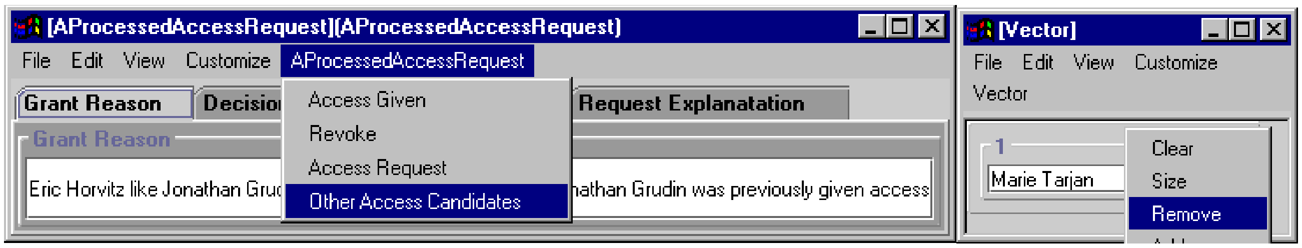


**Figure 8 Consumer, Consumer Agent, and Guardian Interaction**



Grant Reason

Eric Horvitz like Jonathan Gru hathan Grudin was previously given access

Marie Tarjan

**Figure 9 Interacting with an automatically granted request**

Figure 8 illustrates the abstract steps involved in the interaction above. The resource consumer, Jonathan, tries to access the protected object. A consumer agent determines that he does not have the requisite rights, and sends a message to the guardian requesting these rights, who then has the choice to grant them, after possibly reducing/amplifying them, or deny them.

As we see later in the discussion of the abstract architecture, in a practical implementation, the "consumer agent" of Figure 8 would consist of several generic and application-specific software components, which would together determine and communicate the access rights.

Continuing the example, next time Jonathan sees Eric, he tells him that watching is working much better than notification. So Eric goes through a similar process to send a watch request. However, when the request arrives at Prasun's computer, he is away from his desk, teaching his class. His agent realizes that Prasun is inactive and, based on preferences set by him earlier, checks if the request can be granted automatically. It determines that Eric is part of the workspace to be watched and that another workspace member was recently given watch access. Therefore, it automatically grants Eric the access. Based on a user preference about automatic grants, it turns on auditing in the granted permission. The watch tool on Eric's computer informs him that his accesses are being audited.

When Prasun returns to his desk, he is informed about the automatically granted access and given the reason for the grant (Figure 9, left window). At this point, he has many options such as seeing the details of the permission given, revoking it, looking at the audited log to see if any misuse has occurred, and determining other people to whom the inference rule used could be applied. He exercises the last option, and removes Marie Tarjan from the list (Figure 9, right window) as she is performing purely administrative tasks and thus does not need to watch the incremental changes - she just needs to know when the proposal is complete so she can submit it to the administrators. He could also have "preemptively" given other candidates the access using traditional access control, without waiting for requests from them.

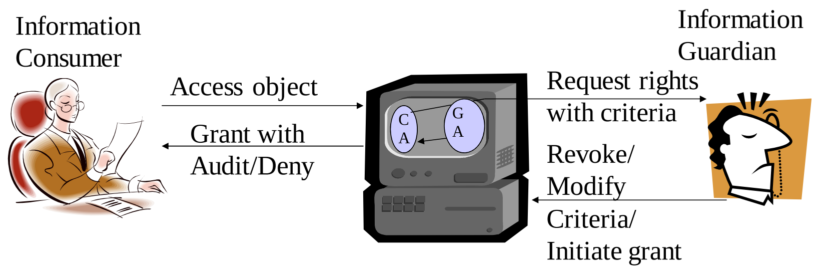


**Figure 10 Consumer, Consumer Agent, Guardian Agent, Guardian Interaction**

Figure 10 illustrates the abstract steps involved in the interaction. The request composed by the consumer agent is processed by the guardian agent. If the guardian agent grants the right, it tells the consumer that the authorized accessed will be audited. In addition, it informs the guardian about the request, the decision made by it, and the criteria used for the decision. The guardian can revoke the right, modify the criteria used by the agent, and/or initiate a grant to others using traditional access control.

Like the consumer agent of Figure 8, the consumer and guardian agents of Figure 10, in a practical implementation, would consist of several generic and application-specific software components.

This scenario illustrates the nature of mixed-initiative control and how it could address the access control problems. None of the participants (the information consumers or guardian) had to find or understand the underlying permissions. They were mainly concerned with using the operations necessary to perform their tasks. When an operation was not authorized, that is, when an *access fault* occurred, the computer tool that invoked the operation automatically determined the rights needed, received notifications about grants of these rights, and retried the operation (Figure 11). An information consumer had to take the extra step of filling an optional field explaining why it was necessary to perform the operation. The overhead for the information guardian was to use a general interface to make a decision. Moreover, it is easier to make the decision in the context of some current task as the information guardian has an idea of what the consumer needs to know. Furthermore, this decision can sometimes be taken automatically without interrupting the guardian. Finally, on receiving a request for a specific access, as in traditional access control, an access granter can use grouping methods such as roles to grant a whole set of accesses. In fact, agents can advise the granter about potential groups based on mined data such as the researchers listed in an NSF proposal. Thus, mixed-access control has the potential of simultaneously reducing the (a) effort required to specify access control and (b) the chance that wrong rights are given.

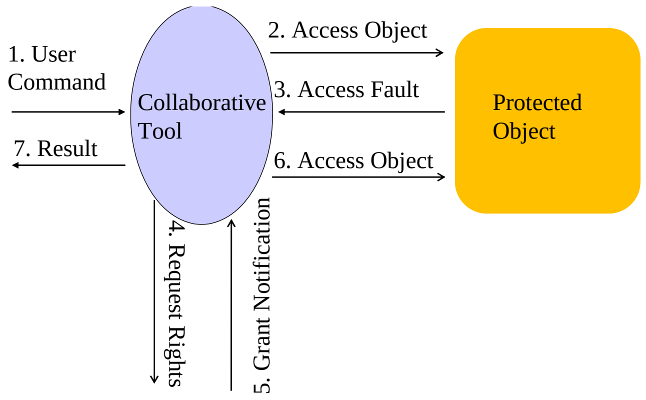


**Figure 11 Access Fault**

We have seen here the use of mixed-initiative access control for an interactive collaborative environment allowing controlled sharing of users' editing activities. In fact, it is equally applicable to traditional non-interactive shared environments such as file systems. In particular, a private directory could have been transformed to a shared one through mixed-initiative control. When Prasun created the proposal directory, he could have announced it to his collaborators without setting any permission. When Jonathan tries to access it, a request is sent and approved in the fashion described above (Figure 12). Later Eric could be automatically given the same rights based on the fact that both Jonathan and Eric are listed in the NSF Fastlane project to which files in the directory have been uploaded. Alternatively, when Prasun accepts Jonathan's request, Prasun could be asked if all other collaborators who have the same role as Jonathan – researcher on the NSF project – should be preemptively given access (Figure 12). Such role discovery and grant amplification is crucial to ensuring that information guardians are not interrupted frequently. Moreover, schemes for managing interruption could be used for requesting and processing access rights such as maintaining multiple accounts and activity awareness [18-23].

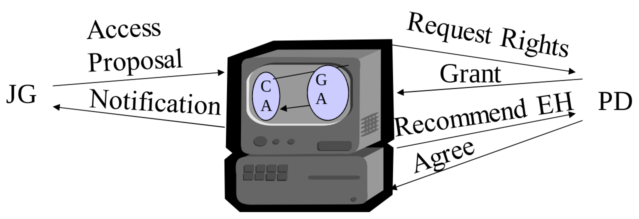


**Figure 12 Mixed-Initiative Access Control for Files**

## IV. ABSTRACT ARCHITECTURE

To understand some of the implementation issues raised by mixed-initiative access control, consider the abstract architecture of a system supporting the above model. Traditionally, a controlled shared environment is guarded by software that either allows an access or gives an access fault, based on the contents of the access matrix defined by authorizers. Figure 13 shows how such a tool would be extended with a collaborative environment to support mixed-initiative control. An access fault is trapped by a consumer agent, which translates it into an automatically generated access request and allows the consumer to change the request before sending it to the guardian. It would also act more autonomously by automatically generating requests for accesses it predicts based on past actions, as illustrated above. Consumer agents can directly communicate with guardian agents, but that requires the design of special communication protocols, clients, and servers. A better approach is for them to communicate via clients and servers of existing interactive communication infrastructures such as instant messaging and email. However, this approach requires the ability to separate normal messages from access requests (shown as white and shaded boxes, respectively), which must be delivered to appropriate guardian agents. This is, in turn, means that existing clients must be extended, as shown by the shaded rings around the communication client ovals in Figure 13. (Such extensions would be in the spirit of the approach described in [29] for providing an email interface to version control systems.) A guardian agent would ask its user to interactively authorize the access and/or provide automatic grant support. In an environment supporting joint ownership, a guardian agent would check with the guardian agent of another user. For example, if a user A asks user B for the contact information about user C, user B's guardian agent would contact user C's guardian agent before granting the access either because B does not have the authority or B and C are joint authorizers. If a grant has been given as a result of an access fault, the result could be delivered to the tool that attempted the access so that it can try that access again. For example, the watch tool of Fig. 4, on receiving the grant notification, would create the display of Fig. 6.

## V. CONCLUSION AND FUTURE WORK

The new forms of access control will not be a silver bullet in all situations. It is for this reason that traditional access control is a fundamental part of our model. The thesis of this position paper is that the new paradigms are applicable in a significant number of situations. In fact, in many cases, we expect the concepts to be used when traditional access specifications have not granted the necessary accesses. Just as servicing of page faults relieved programmers from anticipating all possible concurrent accesses to memory, servicing of access faults should relieve guardians of information from anticipating all possible valid consumers of the information. Further research is needed to define and evaluate the specific protocols used by consumers, guardians and their agents to interact with each other, and system abstractions for easily implementing these protocols. The goal of this paper is to motivate such research.

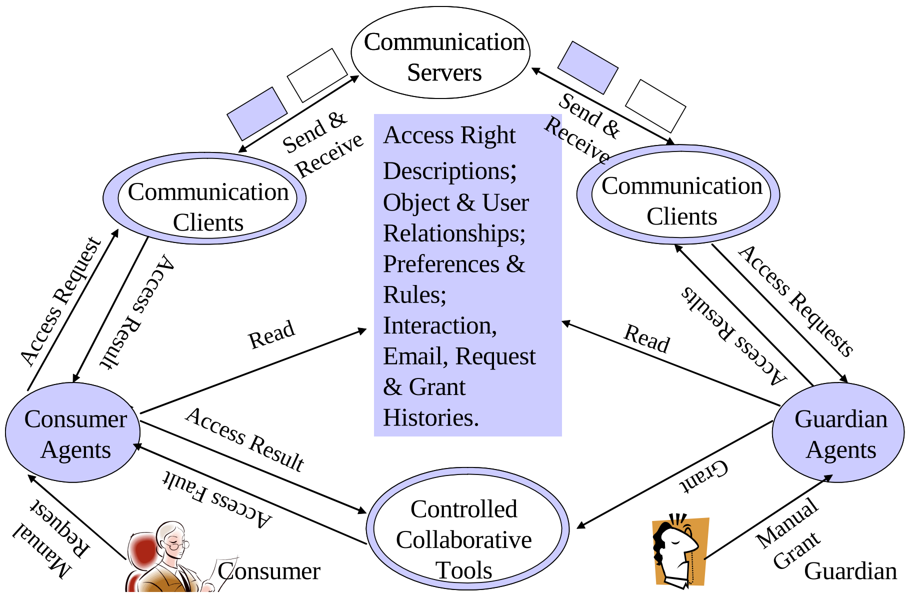


**Figure 13 Mixed-Initiative Abstract Architecture**

## References

[1] Judith S. Olson, Jonathan Grudin., Eric Horvitz. *Toward Understanding Preferences for Sharing and Privacy*. in *Proc. CHI*. 2005.

[2] Palen, L. and P. Dourish. *Unpacking privacy for a networked world.* in *CHI*. 2003.

[3] Newsletter, I.M., *Rules and policies vs. actual practice*, in *Network World*. 2005.

[4] Tolone, W., et al., *Access control in collaborative systems.* ACM Computing Surveys, 2005. **37**(1): p. 29-41.

[5] Lampson, B.W., *Protection.* ACM Operating System Review, 1971. **8**(1): p. 18-24.

[6] Graham, G.S. and P.J. Denning, *Protection - principles and practice.* Proc. Spring Jt. Computer Conf. **40**: p. 417-42.

[7] Ackerman, M., L.F. Cranor, and J. Reagle. *Privacy in e-commerce: Examining user scenarios and privacy preferences*. in *ACM Conference on Electronic Commerce*. 1999.

[8] Shen, H. and P. Dewan. *Access Control for Collaborative Environments*. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. November 1992.

[9] Dewan, P. and H. Shen, *Access Control for Multiuser Interfaces.* ACM Transactions on Computer Human Interaction, March 98. **5**(1): p. 34-62

[10] S. Jajodia, P. Samarati, V. Subrahmanian, E. Bertino. *A unified framework for enforcing multiple access control policies*. in *SIGMOD*. 1997.

[11] Schlegelmilch, J.r. and U. Steffens, *Role mining with ORCA* in *Proceedings of the tenth ACM symposium on Access control models and technologies* 2005 ACM Press: Stockholm, Sweden p. 168-176

[12] Dewan, P. and H. Shen. *Flexible Meta Access-Control for Collaborative Applications*. in *Proceedings of ACM Conference on Computer Supported Cooperative Work*. Nov 1998.

[13] Astrahan, M.M., *System R relational approach to database management.* ACM TODS, 1976. **1**(2): p. 97-137.

[14] Stevens, G. and V. Wulf. *A new dimension in access control: Studying maintenance engineering across organizational boundaries*. in Proc. ACM *CSCW*. 2002.

[15] Povey, D. *Optimistic security: A new access control paradigm*. in *Proc. ACM New Security Paradigms Workshop*. 1999.

[16] Ackerman, M. and L.F. Cranor. *Privacy Critics: UI components to safeguard users' privacy*. in *CHI Extended Abstracts*. 1999.

[17] Tee, K., S. Greenberg, and C. Gutwin. *Providing Artifact Awareness to a Distributed Group through Screen Sharing*. in *Proc. ACM CSCW (Computer Supported Cooperative Work)*. 2006.

[18] Begole, J.B., et al., *Work rhythms: analyzing visualizations of awareness histories of distributed groups* in *Proceedings of the 2002 ACM conference on Computer supported cooperative work* 2002 ACM Press: New Orleans, Louisiana, USA p. 334-343

[19] Begole, J.B., N.E. Matsakis, and J.C. Tang, *Lilsys: Sensing Unavailability* in *Proceedings of the 2004 ACM conference on Computer supported cooperative work* 2004 ACM Press: Chicago, Illinois, USA p. 511-514

[20] Czerwinski, M., E. Cutrell, and E. Horvitz. *Instant Messaging and Interruption: Influence of Task Type on Performance.* in *OZCHI*. 2000.

[21] Dabbish, L. and R.E. Kraut, *Controlling interruptions: awareness displays and social motivation for coordination* in *Proceedings of the 2004 ACM conference on Computer supported cooperative work* 2004 ACM Press: Chicago, Illinois, USA p. 182-191

[22] Fogarty, J., et al., *Predicting human interruptibility with sensors* ACM Trans. Comput.-Hum. Interact. , 2005 **12** (1 ): p. 119-146

[23] Horvitz, E., P. Koch, and J. Apacible, *BusyBody: creating and fielding personalized models of the cost of interruption* in *Proceedings of the 2004 ACM conference on Computer supported cooperative work* 2004 ACM Press: Chicago, Illinois, USA p. 507-510

[24] Horvitz, E. *Reflections on Challenges and Promises of Mixed-Initiative Interaction* 2 AAAI Magazine 28, Special Issue on Mixed-Initiative Assistants (2007)

[25] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. *Device-enabled authorization in the Grey system*. In *Proceedings of the 8th Information Security Conference*, p. 431–445, Sept. 2005.

[26] L. Bauer, L. F. Cranor, M. K. Reiter and K. Vaniea *Lessons learned from the deployment of a smartphone-based access-control system. Proceedings of the 3rd Symposium on Usable Privacy and Security*, p. 64–75, July 2007

[27] Palen, Leysia . *Social, Individual & Technological Issues for Groupware Calendar Systems, Proceedings of the 1999 ACM Conference on Human Factors in Computing Systems* (CHI 99),pp. 17-24

[28] Palen, Leysia and Grudin, Jonathan. *Discretionary adoption of group support software: Lessons from calendar applications. B.E. Munkvold (Ed.), Implementing Collaboration Technologies in Industry. Springer Verlag,* 159-180, 2002.

[29] Dewan, Prasun and McEuen, Henry. *Active Notifications. IEEE CollaborateCom 2007.*