The Symbiotic Relationship between the Virtual Computing Lab and Collaboration Technology

Prasun Dewan
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599, USA
+1-919-962-1823
dewan@cs.unc.edu

Sasa Junuzovic
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599, USA
+1-919-962-1734
sasa@cs.unc.edu

Govindaraj Sampathkumar
IBM Corporation
3901 S. Miami Blvd.
Durham, NC 27703-9315, USA
+1-919-224-1190
gsampath@us.ibm.com

ABSTRACT
The Virtual Computing Lab allows its users to remotely access computing resources spread-out across multiple sites. Collaboration technology allows users to work with each other from distributed sites. These two technologies have a symbiotic relationship wherein each potentially improves the capabilities of the other. Collaboration technology can be used to make users of the virtual computing lab aware of each others' activities, thereby providing a sense of community and allowing sharing of research results. More subtle, collaboration technology can be used to create a shared environment that allows resource users and grantors to fluidly communicate with each other to ensure high and safe utilization of the resources. Even more subtle, the virtual computing lab can be used to conduct experiments that evaluate the scalability of various distributed collaboration architectures. Even more interesting, these three ways to integrate the two technologies can themselves be integrated.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems – distributed applications.

General Terms
Management, Design, Security.

Keywords
Virtual Computing Lab, CSCW, Workspace Awareness, Access Rights Management.

1. INTRODUCTION
Many traditional computer lab resources, such as computers and software that are not shared, can be accessed only by users who are physically present in the lab. In particular, in order to perform some task using a lab machine, a user must go to the lab, wait for the desired resource to become available, perform the task, and then release the machine before leaving the lab. One issue with such labs is the necessary physical presence of the user in the lab. Sometimes the user may not have a means of getting to and from the lab. For example, a student who relies on public transportation to commute between school and home may not be able to reach the lab when the public transit system shuts down for the day or weekend. Moreover, there may be times during which a person feels uncomfortable getting to the lab. For example, an undergraduate student may not want to walk alone to the lab late in the evening. An even more intrinsic problem is that a lab user is confined to machines available in the lab, which determines the nature and number of machines that can be used in an experiment. This, in turn, can lead to inefficient utilization of the available computing resources. For example, available computers in a different physical lab that the user could leverage remain idle because the user cannot combine resources from different labs for a single experiment.

The concept of the Virtual Computing Lab (VCL) provides a solution to these problems by allowing users to reserve and use computers distributed among multiple locations. There has been a significant amount of research in realizing this vision. Here we provide a new angle to this research by exploring how collaboration technology can be married to this concept.

Conference'04, Month 1–2, 2004, City, State, Country.

2. TOWARDS A LAB COMMUNITY
Most of the complementary projects on the VCL reported in these proceedings treat a lab as a collection of only machines (Figure 1 (left)). The challenge in these projects is to distribute the lab machines in a way that efficiently and fairly allocates lab resources to the distributed users. However, when in use, an actual lab is a collection of machines *and people* (Figure 1(right)). Presence of and interaction with others can positively impact the lab experience in many ways. Seeing others in the lab may motivate one to continue working. Lab members can exchange pleasantries or work related information. These conversations may lead to them to exploring new avenues for research. Lab members may get help from lab attendants, friends, and co-workers. In addition, a lab member can interact with other members indirectly by putting post-it notes on machines explaining the member's current or future use of the machines. Team members doing pair programming or some other form of joint project may wish to share a single set of input and output devices. A lab participant may wish to share a particularly exciting result with a co-located user. People who solve difficult problems may build reputations that not only give them a sense of pride but also help them

Figure 1. Two Views of the Physical Lab Being Simulated by VCL: (left) Lab = Collection of Machines and (right) Lab = Collection of Machines and People

advance in their fields. Moreover, lab members interested in the same kind of problems can form groups which lead to a sense of community and opportunistic collaborations. It is our hypothesis that if a virtual computing lab is to be effective, it should virtually support the above and other collaboration scenarios.

Such support can be support mechanisms simulating physical awareness and communication channels, providing users with a feeling of "being there." A popular example of such a channel is an audio/video window of a remote collaborator. However, a computer system can only provide an approximation of being there. If direct simulation of face-to-face interaction modes was the only goal of collaboration technology, then it would always be the second choice to the real thing, taken when time and cost considerations make face-to-face interaction impractical. For this reason, Hollan and Stornetta have argued that to be truly successful, such technology should go "beyond being there," offering interaction features not directly supported in face-to-face collaboration [5]. When this happens, people with use such technology even when face to face interaction is a practical option. A simple example of such an interaction feature is IM, allowing users to carry out multiple private conversations with others.

There has been much work in identifying mechanisms that attempt to meet the dual goals of collaboration technology. While these mechanisms have not been explored in the context of lab-based communities, they have been applied to virtually simulate a (a) general-purpose community center [7], (b) a place to gather after playing the same game, providing "down time" for the players to relax, discussing the game and other issues, after a difficult quest [3][4][8][9], (c) a group exchanging bookmarks [6], (d) and users rating movies for each other [1]. In some of these cases, social interaction became the primary reason for using the system rather than the activity such as gaming around which the interaction was designed. This is not surprising given that many people with a choice of telecommuting prefer to physically commute to work.

Based on the techniques and experience in these communities, we can construct some of the elements of lab-based collaboration technology, outlined below. When a group of users decides to start/end an experiment, notifications can be sent to others informing them about the event and experiment. An architecture can be created for placing the various lab resources in a virtual space, and the users of these resources can be represented by avatars. Users waiting for resources can be put in waiting rooms, and/or a virtual shuttle can take them to various parts of the lab. Audio/video/text-based communication tools together with screen sharing facilities can be used for users to work together on the same or related experiments, share social pleasantries, and ask for and provide help. Browsable profiles can be created for the users of the lab, which can list their personal information, the projects on which they work, and the people they have helped. Prizes can be awarded to the most helpful participants or those who have performed the most interesting experiments. Polls can be used to identify the recipients and solicit their experience with the lab. Based on the interests listed in the profiles, guilds can be created, which can be associated with discussion forums and blogs. If these elements can be supported by the lab, people will want to visit a virtual lab even when they have access to a physical lab with equivalent resources.

To concretely illustrate and define some of the above functionality, consider a scenario in which Alice, Bob, and Charlie are working together on a programming assignment. They are using a collaborative programming environment installed on the virtual lab computers. In the programming environment, there is a special pane that shows Alice's, Bob's, and Charlie's presence status, as in Jazz [2]. Suppose that Alice has a question about the function she is currently writing and she would like either Bob's or Charlie's input. She looks at Bob's presence status and sees that it is "Busy." Charlie's status is currently set to "Available" so she decides to start an audio video conference with him. The session controls and the video feed are shown in a new pane in the programming environment. In addition, next to her own coding window, Alice opens a window showing the code Charlie can currently see on his screen. Conversely, Charlie opens a window showing what Alice can currently see in her coding window. Alice and Charlie can now use the coding windows to synchronize their locations in the code. As they discuss Alice's question, they can communicate using the audio and video links as well as the code itself. For example, if Charlie types something in his coding window, the new text will be displayed in the remote coding window on Alice's computer. If Alice agrees with Charlie's changes, she can synchronize her code to his. Alternatively, if Charlie was only prototyping ideas, his changes can be rejected, and Alice can implement her own solution. Audio

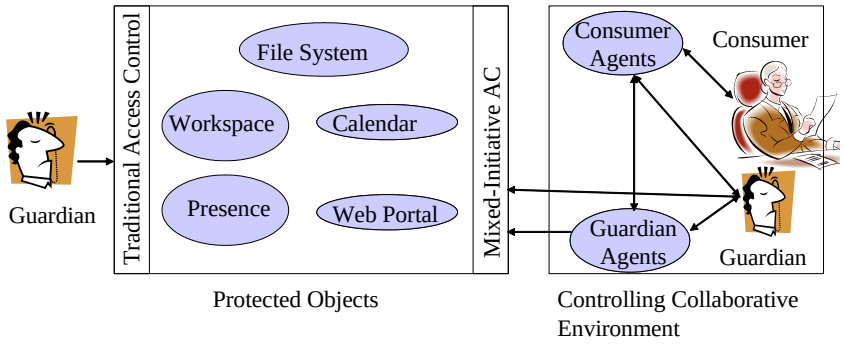

Figure 2. Traditional vs. Mixed-Initiative Access Control

channels can be turned off and instant messaging can be used instead, if Alice and Charlie do not wish to reveal their discussion to other people in the lab. For example, suppose that Alice, Bob, and Charlie are working on a programming project for a class, and they know that at least some of the other people in the lab are working on the same homework. In order to avoid unwillingly sharing the solution with the rest of the class, Alice and Charlie use instant messaging as the communication channel.

3. COLLABORATIVE ACCESS CONTROL

The previous scenario raises interesting problems regarding the issue of access control. In particular, students in other groups should not be able to join in the conversation between Alice and Charlie. On the other hand, if Bob decides to jump in on Alice's and Charlie's chat, he should be allowed as he is their teammate. In this section, we discuss how collaborative access control can be used in the virtual computing lab to solve the problem. In general, a virtual lab with community support must control the communication channels available to the lab members. A virtual lab with or without community support must also support an access-control mechanism that allows administrators to grant computing resources to users. Compared to the number of lab resources, which can be considered constant, controlling the number of possible communication channels, which is exponential with respect to the number of users, can be a fairly involved task. Thus, protecting these resources and communication channels raises usability issues.

Existing research on this topic has tried to address usability problems with access control by providing high-level languages to specify access rights that allow a single specification to give a large number of rights. A radically different, but complementary, approach 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 exchanging experiment results) 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, it is exciting to investigate the idea of designing collaborative environments to ease the setting of both general and item-specific privileges. In

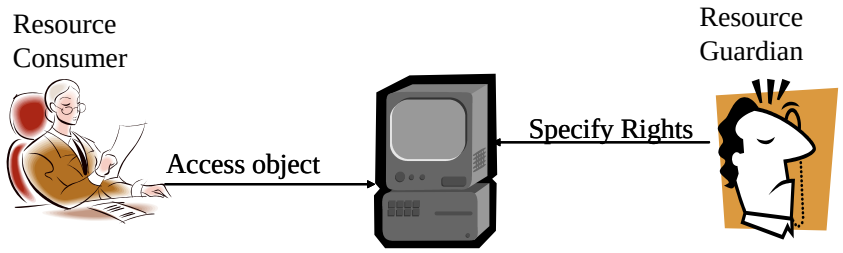

Figure 3. Traditional Access Control

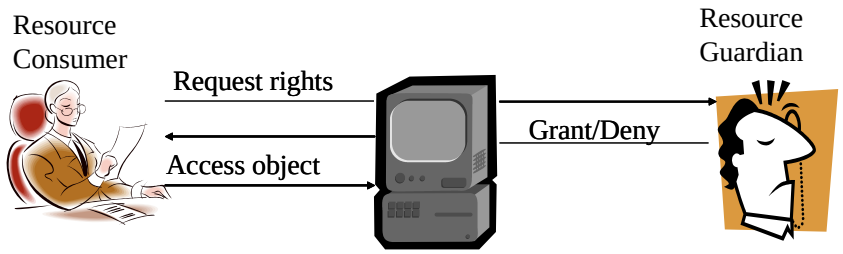

Figure 4. Interactive Access Control

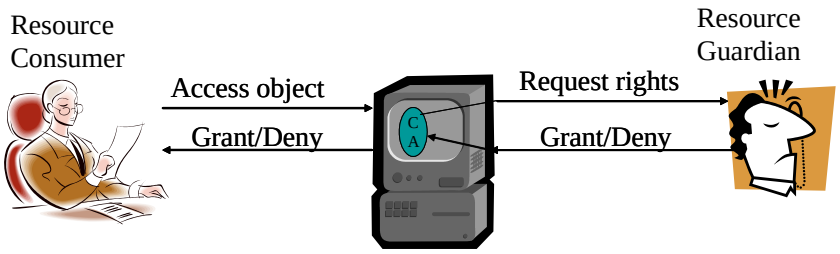

Figure 5. Consumer Agent Automatically Composes and Sends Request

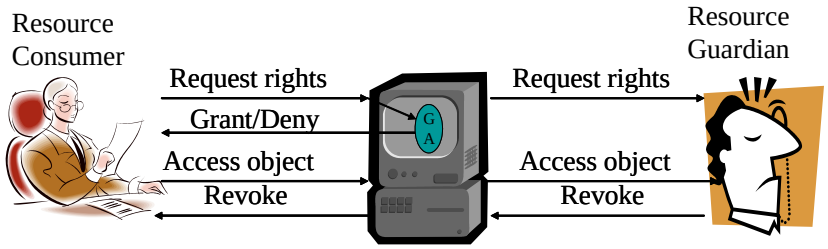

Figure 6. Guardian Agent Automatically Grants Revokable

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. Information consumers will be responsible for sending access requests to information guardians; their agents will (partially or completely) automate this task for them. Information guardians will be responsible for authorizing accesses; their agents will 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.

The primary or protected collaborative environment sets the context for controlled collaboration. For example, a course instructor for the course COMP-101 could define a collaborative environment specifically for it, or for a specific assignment in the course. The specification of the environment would include things such as the software configuration for the machines, the assignments and projects made available to the students accessing the environment, and the kinds of controls are enforced on the collaboration channels. Once a collaborative environment is defined and instantiated, students would be able to access it, just as any resource that is available in the VCL. Depending on the degree of control specified and enforced on the collaboration channels within the environment, students would be permitted or restricted from sharing artifacts and objects created within the context of the collaborative environment with other students. The role of the secondary or access-control environment is to exert this control collaboratively, allowing the various parties to negotiate the nature of the control.

Our expectation is that the specialized controlling collaborative environment will address key impediments to the fluidity of traditional access control, solving some long-standing security problems that have been exacerbated with the increased interest in distributed programmer-defined shared environments such as a community-based VCL.

Let us show more concretely the difference between traditional and mixed-initiative control. The difference between the two is when accesses are granted and the parties involved in creating the grant. In traditional access control, a resource guardian is solely involved in specifying rights to a resource, and he/she must do so before an authorized consumer of the resource uses it (Figure 3). In interactive access control [10], the resource user sends a formal request for access to the information guardian, who can then grant it to automatically change the resource access rights (Figure 4). A requester with granted rights can later access the resource, as in traditional access control. Thus, in this scenario, the consumer and guardian collaborate to create the grant – the consumer composes the request and the guardian approves it. This is very similar to a person sending a formal meeting request to another user, who can then accept it to automatically update the calendars of both users. In a variation of this idea, instead of explicitly composing an access request, a consumer can simply try to access the resource. A consumer agent automatically composes the request, and sends it to the guardian (Figure 5). A dual of the scenario is one in which a guardian agent automatically grants the request, which triggers logging of the consumer's actions. Later, the guardian can examine the log and revoke [10] the grant if necessary (Figure 6). In mixed-initiative access control, all of these schemes are supported by allowing consumers, consumer agents, guardians, and guardian agents to collaborate with each other to grant accesses.

To illustrate the role of such control in a VCL, consider a class that needs to do projects on the VCL. Students accessing this collaborative environment would experience something similar to entering a physical lab on a campus – they would be able to "see"

Table 1. Systems with different shared layers and employing different architectures

System	Layer Shared	Mapping
Screen Sharing in LiveMeeting, VNC, Webex, SameTime	Screen	Centralized
App Sharing in LiveMeeting, Webex, SameTime	Window	Centralized
Groove PowerPoint, Webex PowerPoint, SameTime Whiteboard	Model	Replicated
LiveMeeting PowerPoint Sharing	Model	Centralized

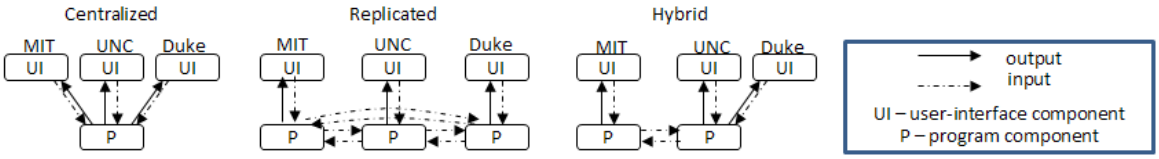

Figure 7. (left) centralized, (center) replicated, and (right) hybrid architectures

who else is there, chat with them, and so on. Depending on the degree of control specified and enforced on the collaboration channels within the environment, students would be permitted or restricted from sharing artifacts and objects created within the context of the collaborative environment with other students. Let us assume that the professor uses traditional access control to create a group containing the students, and give the group the required access. Later, let us say that a particular student, Bob, needs some help debugging an assignment. Another student, Alice, might join the debugging session by placing her avatar in front of Bob's computer. However, this is a restricted operation requiring appropriate rights. Therefore, Alice's consumer agent determines these rights and sends them to the professor's guardian agent. The professor is not online, but has instructed his/her guardian agent to automatically grant these to Alice and other trusted students. The resulting collaboration session would be automatically audited by the system. Later, the professor can look at the logs to see if Alice did indeed follow the honor code by not showing her solution to Bob.

As the VCL is intended to enable institutions to share each others' computing resources, one issue is dealing with cross-institutional security policies. For example, how does a student at NCSU access a resource physically located at UNC? Facilitating seamless collaboration in a secure way across institutional boundaries would require several additional aspects to be worked out, from a security point of view. Users in one autonomous institution will not be known in a different institution. For example, user bob@ncsu.edu has certain privileges and attributes associated with him, all within the context of his institution, NCSU. Within this context, user bob@ncsu.edu is known to be a 6th semester undergraduate student in Computer Science. If bob@ncsu.edu is allocated a resource physically located in the UNC campus, the security enforcement modules in UNC have to recognize that bob@ncsu.edu is an NCSU student, and that he has certain privileges that transfer over to the UNC environment. A collaboration environment is defined within an institution, such as by a course instructor. The definition of this has to span institutional boundaries so that physical resources could be allocated anywhere throughout the VCL. Further, the collaboration channels have to be opened up across the institutional boundaries as well.

4. MEASURING PERFORMANCE OF COLLABORATION ARCHITECTURES

A collaboration environment would be used only if its performance is tolerable. To illustrate, consider the following scenario. During candidates' day at UNC, the computer science department invites a number of students for demos of the research projects within the department. Unfortunately, some of these candidates are also invited to the candidates' day at Duke and MIT, both of which happen on the same day. Therefore, to give all invited students a taste of research at UNC, UNC shares the demo applications using an application-sharing system. This allows the students visiting the other schools to remotely try the demos as long as they have with them Internet-connected portable devices, such as laptops and PDAs. The interactivity of the demo is of the utmost importance. The shared demo application must respond to the operations by the students at Duke and MIT quickly and notify the student of any operations by other users in a timely fashion; otherwise, the student may get bored and quit the demo, which could result in the student not coming to UNC. Even worse, they may never try collaboration technology again!

An important interactivity metric illustrated in the above scenario is the *local response time for an input command*, which is defined as the time that elapses from the moment a user enters an input command to the moment that user sees the output for the input command. Another, related interactivity metric illustrated is the *remote response time for an input command*, which is defined as the time that elapses from the moment a user enters an input command to the moment a different user sees the output for the input command. One factor that impacts the interactivity of an application-sharing system is the collaboration architecture it uses.

A *collaboration architecture* defines the logical system components, their physical distribution, and the interaction between them. Collaborative systems assume that the shared application is divided into the program and user-interface components. When the application is shared, the user-interface component is always replicated on all the users' machines. The program component, on the other hand, may or may not be replicated. Regardless of the number of program component replicas, each user-interface component must be mapped to a

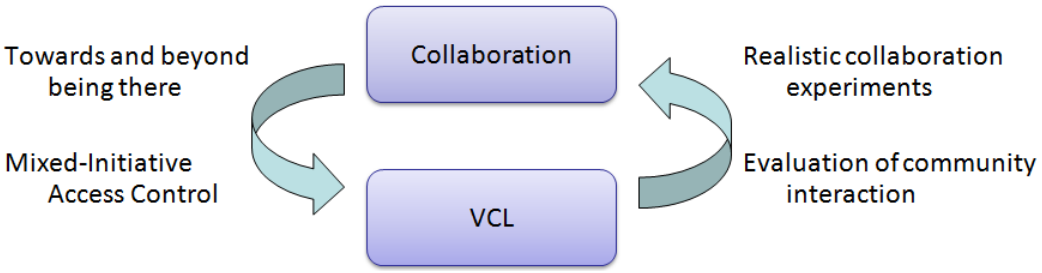

Figure 8. The Symbiotic Relationship between VCL and Collaboration Technology

program component to which it sends input commands and from which it receives outputs. To keep the program component replicas synchronized in this many-to-one mapping of user-interfaces to program components, each program component must send input commands it receives from the user-interfaces mapped to it to all the other program components.

Three popular mappings have been used in the past: centralized, replicated, and hybrid (Table 1). The *centralized mapping* maps all user-interface components to a single program component. The *replicated mapping* maps each user-interface to its local program component. All other mappings are *hybrid mappings*. To illustrate the three kinds of mappings, consider the above candidates' day scenario in which two students, one from Duke and one from MIT, are using PDAs to remotely join the application-sharing session. The professor demoing the application is also in the session and is joined using a desktop computer. A centralized mapping in which the program component on the professor's computer is used is shown in Figure 7 (left). The replicated mapping is shown in Figure 7 (center). A hybrid mapping in which the user-interface on the Duke PDA is mapped to the program component on the UNC desktop and the other user-interfaces are mapped to their local program components is shown in Figure 7 (right).

The choice of mapping can impact the interactivity of the shared application. Suppose that the professor is demonstrating an exceptionally good but computationally expensive Checkers AI and is inviting the students at Duke and MIT to challenge it. What mapping should be used to provide good local response times to Checkers moves entered by the two students? As UNC and Duke are near each other, the network latency between them is low. Suppose that the network latency between UNC and MIT is also low. Assume that the professor's desktop is much more powerful than the PDAs. Since the AI algorithm is computationally heavy, a centralized mapping (Figure 7 (left)) in which the UNC desktop runs the program component may offer the best local response times. The reason is that it pays for both PDAs to incur the round-trip costs between them and UNC for using the desktop as a high-speed computation server. If the network latencies were high, then the high round-trip times between the PDAs and the desktop would annul the benefit of using the desktop as a high-speed server. In this case, the replicated mapping shown (Figure 7 (center)) could give optimal local response times.

Experimental results are necessary to verify the intuitive conclusions made above about the relative performance of the various architectures in the collaboration scenario described above. To illustrate the nature of and problems with these experiments, consider a specific experiment a student carried out recently on this subject. He reserved, at around 4am, 8 public computers, got temporary permission to disable firewalls that prevent Java RMI server from running, and spent four hours running a log he gathered from an internet chat session! A dedicated lab in which the computers are constantly updated would alleviate some of these problems, but would not allow us to scale our experiments to a large number of computers. Such scalability studies need to be performed, for instance, to answer how many users can IBM SameTime efficiently support in a presentation given to the whole company, and can this number be increased by a different architecture, and can the architecture be dynamically changed to adapt to the number of users, the power of their computers, and the network connections among them. Therefore, the virtual computing lab is the solution.

5. CONCLUSIONS AND FUTURE WORK

There are at least three ways in which collaboration technology and the idea of a virtual computing lab can be integrated. Collaboration technology can virtually simulate and augment the sense of community in a physical lab. In addition, it can create rich communication channels to allow resource guardians to grant access to us resources to resource consumers. In these two cases, collaboration technology is being used to improve the nature of a virtual computing lab. Conversely, a virtual computing lab can be used to perform scalability experiments comparing the performance of various architectures used to implement collaborative applications. This is graphically illustrated in Figure 8.

Each of these directions can be pursued by an independent project. Interestingly, however, these three ways to integrate the two technologies can themselves be integrated in a single project. Imagine an IM-like tool that provides status about the activities of all of the current users of the lab and allows users to communicate with each other. Such a tool can be used not only for regular communication but also access control requests and grants, much as an email tool is used for ordinary communication and adding events to the calendar. For example, Alice, once she finishes an assignment, can use a special command provided by the tool to request the professor rights to help others with their assignments. Before granting the right, the professor might open a video, audio and/or text channel to Alice to set some limits on the help provided. The first version of the tool requires the professor to manually determine which of these channels can be expected to give tolerable performance. In addition, it requires the system administrator to configure the architecture. This tool is used to conduct controlled performance experiments with the tool, which recursively, requires the use of the tool to request the computers needed in the experiments. The results of the experiments are used to create a new version of the tool that, given measured user and

system parameters, automatically determines the architecture of the tool and the channels that can be efficiently supported.

Future work is needed to explore the various symbiotic relationships shown in Figure 8, independently and in an integrated fashion. What we have identified above is a set of mechanisms potentially exciting and useful for some set of lab participants and resource guardians. Further work is needed to augment/reduce this set and determine the kind of participants that finds the various feature in the set useful. The main goal of this paper is to motivate such research.

6. ACKNOWLEDGEMENTS

This research was funded in part by *IBM* and NSF grants ANI 0229998, EIA 03-03590, and IIS 0312328. Discussions with Diane Pozefsky and Andy Rindos contributed to these ideas, and the comments of the reviewers helped improve the presentation. The idea of lab community was inspired by the comprehensive paper Julia Grace wrote on online communities.

7. REFERENCES

[1] Beenen, G. et al. Using Social Psychology to Motivate Contributions to Online Communities. CSCW 2004.

[2] Cheng, L.T., et al. Jazzing up Eclipse with Collaborative Tools. OOPSLA workshop on eclipse technology eXchange. 2003.

[3] Ducheneaut, N. and Moore, R.J. The Social Side of Gaming: a Study of Interaction Patterns in a Massively Multiplayer Online Game. CSCW 2004.

[4] Ducheneaut, N., et al. "Along Together?": Exploring the Social Dynamics of Massively Multiplayer Online Games. SIGCHI 2006.

[5] Hollan, J. and Stornetta, S. Beyond Being There. CHI 1992.

[6] Lee, K.J. What Goes Around Comes Around: an Analysis of del.icio.us as Social Space. CSCW 2006.

[7] Millen, D.R. and Patterson, J.F. Stimulating Social Engagement in a Community Network. CSCW 2002.

[8] Nardi, B. and Harris, J. Strangers and Friends: Collaborative Play in World of Warcraft. CSCW 2006.

[9] Seay, A.F., et al. Project Massive: a Study of Online Gaming Communities. CHI 2004.

[10] Stevens, G. and V. Wulf. A New Dimension in Access Control: Studying Maintenance Engineering Across Organizational Boundaries. CSCW 2002.