Distributed Cognition and Joint Activity in Collaborative Problem Solving

Paul P. Maglio (pmaglio@almaden.ibm.com)
Eser Kandogan (eser@almaden.ibm.com)
Eben Haber (ehaber@almaden.ibm.com)
IBM Almaden Research Center, 650 Harry Road
San Jose, CA 95120 USA

Abstract

Troubleshooting large software systems is often highly collaborative. Because these systems consist of complex infrastructures with many interdependent components, expertise is spread across different people and organizations. Those who administer such systems are faced with cognitive and social challenges, including the establishment of common ground and coordination of attention, as they troubleshoot in collaboration with peers, technical support, and software application developers. We take a distributed cognition approach to interpreting a specific instance of problem-solving in administering a web-based system, examining the movement of representational state across media in a single system administrator's environment. We also apply the idea of language use as joint activity to understand how discourse attributes affect what is accomplished collaboratively. Our analysis focuses on information flow among participants and other sources, and how these affect what information is attended to, transmitted, and used.

Introduction

Millions of users of online services such as banking and shopping rely on instant transactions, round-the-clock access, and foolproof record keeping. The computer system infrastructures needed to support such applications consist of diverse components, such as database management systems, web servers, and application servers, all of which work together in complex ways to deliver fault tolerant, scalable, secure applications. Yet with such systems growing ever larger and ever more complex, *manageability* is fast becoming an obstacle to system administration: Administrators who install, configure, maintain, and support such systems must handle ever larger and ever more complex tasks (Anderson, 2002; Woods, 1988).

Large-scale systems contain many interdependent components that are not always designed to work together. Expertise and responsibility for different components is typically spread across people and organizations. Administrators are faced with daunting cognitive and social challenges. Complexity and scale are such that administering a complete system is usually beyond the abilities of a single person, making collaboration among team members and outside experts crucial to completing many tasks, especially time-critical tasks such as troubleshooting. Administrators have developed many heuristics for problem-solving and practices for collaborating with others to do their jobs effectively.

Collaborative troubleshooting involves coordinating activity and information from people and other sources. In this paper, we take a distributed cognition approach (Hutchins, 1995) to understand problem-solving in system administration, focusing on issues of trust and its relationship to the management of attention. Distributed cognition treats certain arrangements of people and artifacts as cognitive systems, effectively computing functions by transmitting representations (e.g., language, computer commands) across media (e.g., air, computer screens). The idea is that the cognitive computation can be (partially) understood by tracking propagation of representations.

We combine distributed cognition with the joint activity theory of language use (Clark, 1996) to interpret the way discourse attributes affect problem-solving in system administration. On the joint activity view, people use language to create and complete projects together, such as the project of coming to a mutual understanding (e.g., agreeing on a problem and its solution) or the project of accomplishing some other task (e.g., detecting a problem and determining a solution). By seeing language use as joint activity, we can discover why people interact the way they do (see also Fairburn, Wright, & Fields, 1999).

In what follows, we examine the process of solving a single problem that occurred during normal maintenance of a web-based system. This episode lasted three hours and involved nine people using many different collaboration tools. By tracking the movement of representational state across media in one administrator's environment, and by analyzing how joint projects are established, we examine how discourse attributes and information flows affect what information is attended to, transmitted, and used.

Study

Our data come from field studies conducted to develop knowledge of software system administrators' culture, organization, collaboration, work styles, problems, strategies, and tool use. Our overall goal is to improve products, practices, and processes of administration. For the data reported here, we observed administrators in a computer services group that hosts customer web applications. We used several techniques to gather data, including surveys, observations, video recording, formal and informal interviews, and material (hardcopy and online) collection.

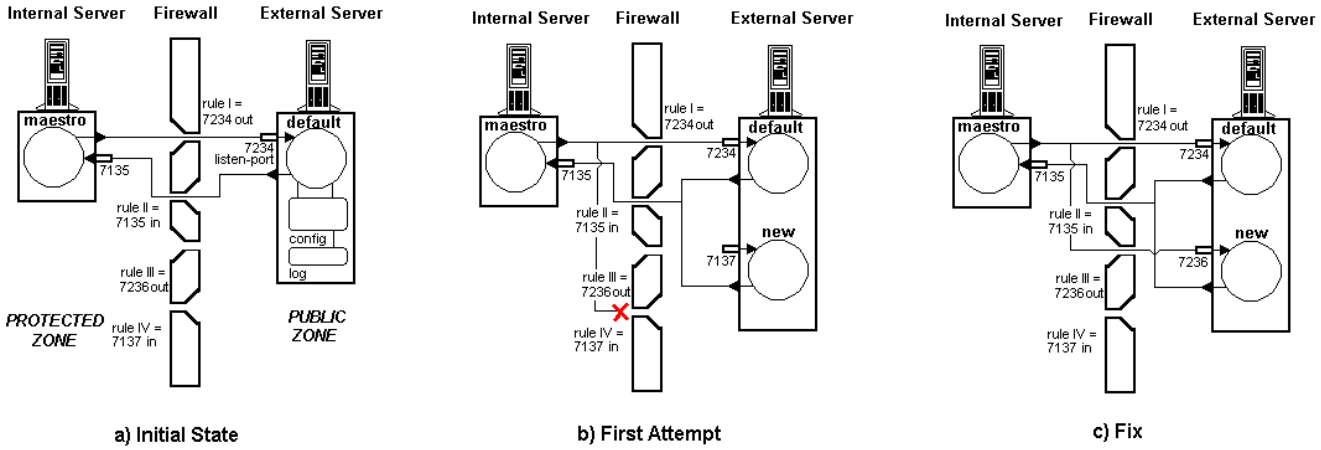

Figure 1. (a) Initial state of the system. Note that as part of preparations for the operation, two new rules were added to the firewall; (b) System state after the first attempt to add a new player instance. The maestro could not contact the new player instance, however, as the new player was configured to accept messages on port 7137 whereas there was no firewall rule to allow messages from internal to external server on port 7137 (only vice-versa); (c) System state after the fix. The player was configured to accept messages from the maestro on port 7236, which the firewall allowed.

We detail a specific problem-solving episode, analyzing how access to information and how aspects of discourse influence administrators' collaboration practices and problem-solving effectiveness. The descriptions of agents, representations, and representational activities that follow are restricted to only this single episode. We first describe (a) some technical details of the task; (b) the people, computers, and information sources involved; and (c) the kinds of representations that were used.

Task

A customer installation included a certain software product for managing the flow of data between the public internet and the customer's protected internal network. This software had two parts, a *player instance* running on a server in the public zone and a *maestro instance* running on a server in the protected zone. Connections between player and maestro were regulated by a firewall (Figure 1a). The customer requested that a second player be added in the public zone.

The task involved creating a second player instance, configuring the maestro to allow the new instance to access certain resources, and configuring the firewall to permit communication between the maestro and the new player instance. Communication between systems is done through *ports* (represented by integer port numbers). The firewall regulates communication using a set of rules that define allowed port numbers and communication direction.

People and Computers

Many individuals from many groups were involved in the problem-solving episode. Primary actors included our main administrator (hereafter, ***Admin***), the project architect (***Archi***), technical support for the product (***Tech***), and ***Admin***'s colleague who had access the same systems (***Colle***). Less important contributors included ***Admin***'s officemate, the customer relationship manager, the project executive, ***Archi***'s friend who was the product developer, and ***Colle***'s manager.

Systems that received, processed, and transmitted information during the episode included an *internal server* machine that ran the maestro application in the protected zone of the network, an *external server* machine that ran the player instances outside the protected zone, a *firewall* that regulated messages between internal and external servers, the *maestro* process that orchestrated message passing, the *default player* process that processed incoming messages, and the *new player* process that the customer wanted added on the external server.

Representations and Actions

As ***Admin*** and collaborators worked on the problem of adding a new player instance to the external server, they used various methods of communication and tools for interacting with systems. Their communication involved verbal exchanges face-to-face or over the phone, and textual exchanges through email and instant messages.[1]

When interacting with computer systems, administrators relied mainly on commands typed directly into the system's *command line*, a general human-computer interface that requires the user to know precisely the names and parameters of specific commands for the computer to execute. Command line users are typically very experienced. Commands can control processes and machines, and can display state and configuration information.

[1] Email and instant-messages are often used simultaneously; email is more persistent, yet must be explicitly received and read by a recipient whereas instant messages instantaneously "pop up" in a special window on the recipient's screen.

Information representations included *configuration files*, *log files*, online and paper instruction *manuals*, and *port listings*. There were separate configuration files for each server process, which included settings that allow administrators to change port numbers. Likewise, each process had its own log files that report errors and warnings that occur while the process is running.

We now turn to the details of the problem-solving episode in which ***Admin*** worked with many others and consulted many information sources to add a new player instance to the external server. All observations were taken from ***Admin***'s perspective.

Observations

We begin with an overview of the three-hour problem-solving episode. We then focus on several specific interactions that illustrate how constraints on propagation of representational state and how discourse attributes affect what information is attended to, transmitted, and used.

Overview

Initially, ***Admin*** received an email message describing the steps required to add a new player instance to the external server. ***Admin*** began by requesting the firewall team open two ports: port 7137 from external server to internal server, and port 7236 from internal server to external server (see Figure 1a). After the firewall team completed the job, ***Admin*** began to follow the email instructions.

First, ***Admin*** copied the command to create a new player instance from the message and pasted it onto the command line of the external server:

```
m_web create {instance} -m {internal-port}
```

He then proceeded to substitute "`new`" for `instance` and "`7137`" for `internal-port` by directly editing the text on the command line, resulting in

```
m_web create new -m 7137
```

This was typical of ***Admin***'s style—whether the command was copied and pasted from email or from other sources—he would substitute directly into the command.

Admin executed the command, resulting in the configuration shown in Figure 1b. He then copied from the email message a command to configure the maestro server to permit the new player access to certain resources. When he filled in the parameters and executed this command, the following error appeared on his screen:

Cannot reach server: Error 1231A

Note the ambiguity in the text: Which server cannot be reached, the maestro server or the player server? To try to understand the error, ***Admin*** engaged in phone, email, and instant-message conversations with ***Archi,*** the application architect, and ***Tech***, the technical support person. As time passed and the problem remained unresolved, ***Admin***'s office-mate, colleague (***Colle***), and a developer friend of ***Archi***'s all joined the conversation. At several points, the customer relationship manager and the project executive requested updates from ***Admin*** on the state of problem resolution.

The pattern of interaction among ***Admin***, ***Archi,*** and ***Tech*** had ***Admin*** in control of the systems, with ***Archi*** and ***Tech*** asking him to run commands or transmit to them aspects of configuration and system state. By contrast, ***Colle*** could access the system so his work on the problem was more independent, reporting back to ***Admin*** his findings and suggestions for a solution.

In these conversations, various representations of system state, including error numbers, configuration file entries, and portions of log files, were exchanged over the telephone, instant messages, and email. As information was transferred, it was typically transformed from abstract descriptions, such as `internal-port`, to specific names and numbers, such as `7137`, and vice versa.

The problem was resolved after nearly three hours by ***Admin*** and ***Colle***. The problem turned out to be a misunderstanding of the meaning of `internal-port`, as specified when creating the new player instance. The `internal-port` is used for communication from maestro to player instance—rather than the other way around, which was what ***Admin*** had originally believed. Thus, using another port, in this case `7236` rather than `7137`, solved the problem (see Figure 1c).

Communication from all player instances to maestro was handled on a standard or default port (`7135`), which was specified in the configuration file as `master-port`. This specification was overlooked by ***Admin.*** At the center of the problem were specific transformations carried out by people and computers on various representations of system state. The time taken to resolve the issue was affected by these transformations during attempts by participants to reach a common understanding of the semantics and syntax of system components and their representations.

We now turn to three particular interactions in more detail. These relate to problem diagnosis and problem resolution, illustrating how attributes of communication between participants influence how information flows through the system.

Do you have the manual?

Nearly two hours into the session, ***Tech***, technical support, and ***Admin*** exchanged instant messages [1:46:00]:[2]

`Tech:` `Can you verify listen port 7234 or 7237 is listening?`[3]

Tech asked the right question to find the source of the problem. 7234 was the listen port for the default instance, and 7137 was the listen port for the new instance (although ***Tech*** wrote "7237" he likely meant "7137"). ***Admin*** determined the listen ports using the command line, which

[2] Timestamps in brackets indicate elapsed time from the beginning of the episode; in this case, 1 hour, 46 minutes, and 0 seconds in.

[3] Transcripts displayed in a `fixed width font` indicate communication via instant messages; *italics* indicate voice.

showed ports 7137 and 7234 among others. To himself, ***Admin*** muttered [1:46:35]:

> **Admin:** *7137 and 7234. This is the problem! Huh. Oh, no wait! Hmm, that should be fine.*

Admin might have realized that 7137 should not be a listen port, but he focused instead only on 7234 and filtered other information from the port list when reporting to ***Tech*** [1:47:10]:

> **Admin:** It is listening 7234…is it ok that it listens on the same port as the default instance?

Tech responded negatively, which might have led ***Admin*** directly to the solution [1:48:15]:

> **Tech:** Don't think so.
> **Tech:** Do you have the manual?
> **Tech:** I'm trying to find it... working from home today.

But on seeing these messages, ***Admin*** spoke with ***Archi*** by phone [1:49:20]:

> ***Admin****: You got to be kidding me! Oh God, this support guy is asking me for the manual.*

Archi told ***Admin*** that he knew someone else who could help, and eventually brought a developer into the discussion.

It seems clear that ***Admin*** lost faith in ***Tech***. Yet just before asking the question about the manual, ***Tech*** had asked a question that would have quickly led to resolving the problem (about which port was listening where). Although ***Admin*** continued to discuss the problem with ***Tech*** for a while longer, ***Admin*** showed his disinterest in communicating with him. In fact, ***Tech*** later pointed out where to change the listening port for the new instance, and ***Admin*** responded verbally (to no one) [1:58:25],

> ***Admin****: This guy is totally useless.*

From this point on, ***Admin*** ignored ***Tech*** completely. The instant message windows that contained the exchange with ***Tech*** became covered over. ***Tech***'s last message arrived after a long period of no communication [2:13:00],

> Tech: What is happening?

Analysis. All system state information flowed through ***Admin***. For ***Tech*** to help solve the problem, he had to get ***Admin*** to discover and report information about the state of the system. The flow of information between the new player instance and ***Tech*** was filtered by ***Admin***'s understanding of the system: what was reported was not the same as the results displayed on ***Admin***'s screen. Yet just as ***Tech*** started to extract critical information from ***Admin***, ***Tech*** asked for the manual. From ***Tech***'s perspective, this can be seen as initiating a joint project with ***Admin*** to discover whether the new instance ought to be set up to listen on the same port as the old instance. From ***Admin***'s perspective, this appeared to be an inappropriate joint project, as he seemed to believe that ***Tech*** should simply know the answer without needing to refer to the documentation. Thus, ***Admin*** did not take up the project.

What are you talking about?

A series of exchanges between ***Admin*** and ***Colle*** that led to the resolution of the problem came soon after communication with ***Tech*** broke down. ***Colle***, a close colleague of ***Admin***'s, was told by the customer relationship manager to help ***Admin***. In fact, ***Colle*** checked with ***Admin*** in person (walking into the office and discussing the issue with him) about one hour into the session, and stayed in contact with ***Admin*** on and off via instant messages. ***Colle*** worked in the next office, where he had access to the same servers. Eventually, ***Colle*** discovered that the internal server was trying to communicate with the external server over port 7137 [2:02:15]:

> **Colle:** We were supposed to use 7236. Unconfigure that instance and …
> **Admin:** Can't specify a return port… you only specify one port

Admin's response indicates that he did not know how to specify the port connecting internal to external server. ***Colle*** explained how he came to this conclusion (to use 7236 rather than 7137) by pasting into instant messages the commands he ran to test communication from internal to external server, attempting to persuade ***Admin*** that he was correct. The exchange became more heated [2:02:20]:

> **Colle:** You specified the wrong port.
> **Admin:** No, I didn't.
> **Colle:** You did it wrong. Yes, you did. You need to put in 7236.
> **Admin:** we just didn't tell to go both ways. The other port has nothing to do with this.
> **Colle:** Well, all I know is what I see in the conf file
> **Admin:** we thought that was the return port. That is not a return port.
> **Colle:** there currently is no listener on <internal-server> on 7137. So use 7236. DO IT!

Admin then called ***Colle*** on the phone [2:03:45]:

> ***Admin****: What are you talking about? 7236?*
> ***Colle****: Yeah?*
> ***Admin****: We thought that it came in on 7137 and went back on 7236, but we were wrong, that 7236 is like an ACTPS listener port or something?*
> ***Colle****: It will still come in on 7135 to talk to maestro server apparently...*
> ***Admin****: right?*
> ***Colle****: What's happening is it's actually trying to make a request back, um, through the 72... well actually trying to make it back through the 7137 to the instance...*
> ***Colle****: .. and it's not happening.*
> ***Admin****: I know. I know that. But I can't tell it to...*
> ***Colle****: .. just create it with the 7236. Trust me.*
> ***Admin****: Why? That port's not, that's going the wrong, that's only one way too.*
> ***Colle****: Trust me.*

***Admin**: It's only one way. Do you understand what I am saying?*

***Colle**: Cause it's the maestro talking back to the player server instance.*

***Admin**: Yeah, but how does the player instance talk to maestro to make some kind of request?*

***Colle**: 7135 is the standard port it uses in all cases. So we had it wrong. Our assumption on how it works was incorrect.*

***Admin**: All right, all right.*

***Colle**: If it doesn't work you can beat me up after*

***Admin**: I want to right now. (Laughter on both sides)*

Analysis. In this case, all system state information did not flow through ***Admin***. Because ***Colle*** had access to the same systems as ***Admin*****,** he could examine system state directly (as ***Colle*** said, "`all I know is what I see in the conf file`"). Communication centered on ***Colle***'s instructions for solving the problem by configuring the new player instance to use a different port. When ***Admin*** did not immediately take up the project to fix the port settings, ***Colle*** explained the problem. ***Colle*** shared the commands that showed him which ports were listening. Again, ***Admin*** did not take up the project proposed by ***Colle***. When the conversation shifted from instant messages to phone, ***Admin*** finally accepted ***Colle***'s project to change the port settings, but only after ***Colle*** stated that their understanding of how the system worked had been incorrect all along. Because ***Admin*** was upset, ***Colle*** made a special effort to appease him by jokingly agreeing to be physically harmed if his hypothesis turned out to be wrong. In both admitting prior misunderstanding and joking, ***Colle***'s discourse was not about the business at hand, the establishment of common ground about the state of the system. Rather, ***Colle***'s statements served a different communicative function: establishing a different joint project that would enable ***Admin*** to follow ***Colle***'s directions. ***Colle*** found that rather than debugging ***Admin***'s knowledge of the state of the system (repeatedly explaining what the port settings should be), he had to debug ***Admin***'s model of the system itself (explicitly stating "our assumption was wrong" about the direction of the ports).

I've got too many #@&! people annoying me!

Throughout, ***Admin*** maintained multiple channels of communication (phone, email, instant messages, and face-to-face) with others. ***Admin***'s information environment was filled with many demands for his attention. One striking instance occurred near the end of the session. By this point, both ***Colle*** and ***Archi'***s friend had suggested the same root cause, and ***Admin*** had agreed to the try the solution. ***Admin*** and ***Colle*** spoke by phone [2:05:60]:

***Colle**: Actually, you can create a new one.*

***Admin**: Yeah, that's what I'm gonna do. (sighs)*

***Colle**: I'm telling you man, this is what's happening. You can see by the connection it's trying to make. There is no 7137 listener on maestro right now, so what is it going to try to connect to?*

***Admin**: Yeah, I understand what you're saying.*

***Colle**: You know sure, we can see this in the logs, but I think we're already there where we've found out what the issue is.*

***Admin**: All right, all right.*

***Colle**: It's trying to make a return port.*

***Admin**: All right!*

***Colle**: I verified in the other player log that the…*

***Admin**: Can you hang on please!*

Admin put ***Colle*** on hold [2:10:15]:

***Admin**: I can't, I can't think because I've got too many <expletive> people annoying me… There's too many people. I hate when there's too many people involved, and everyone's telling me to do something different and it's like you can only do one thing at a time, you know.*

After following ***Colle***'s instructions, ***Admin*** attempted to explain the process to ***Archi*** by phone [2:20:15]:

***Admin**: All right I think we got it. What we did was, uh what did we do? The, uh, rather than specifying the 7137 port, that, cause...What happened was we had opened a port going to... We were under the impression for some reason that the port that player talks to maestro over is 7137 and then maestro returns on 7236, or 7135 and 7234, whatever. That was the impression we were under, so we opened the firewall ports with, um, and we opened it for 7137 to go from player to maestro and then 7236 to go maestro to player, so we only needed to open one port because, uh, and the port we needed to open was the one that maestro goes back to player on, so we already had that open, but it was the 7236 port so we just, I created the new instance specifying that as the port, so in the -m option I specified 7236 and I created all the junctions and everything looks cool at this point.*

Analysis. ***Colle*** coached ***Admin*** through the process of fixing the port settings. But in the end, ***Admin***'s explanation was confused, suggesting he actually had little understanding of the details. Again, the movement of representational state was from ***Colle***'s screen to ***Colle*** to ***Admin***. For his part, ***Admin*** put ***Colle*** on hold to execute the plan undisturbed. ***Admin*** relied on memory of what ***Colle*** had said, commands ***Colle*** had sent via instant messages, and the manual to execute the command to create a player instance with the correct port number.

Results and Discussion

Our administrator (***Admin***) spent nearly three hours coordinating information from various sources to transform the initial configuration (Figure 1a) into the final configuration (Figure 1c). He coordinated information from many other people, from many configuration files and log files, from the output of many commands typed on the

command line, and from many online documents including web pages and email. We have sampled only a few of these interactions. Nevertheless, the story that emerges is one of how constraints on movement of representation and how attributes of communication influence what information is attended to, transmitted, and used.

Consider first the interactions with technical support (***Tech***). As described, the support person was in fact on the right path to the solution when he asked our administrator to verify the listen ports. For his part, our administrator executed the commands to verify the ports, but in examining the propagation of representational state (Hutchins, 1995), we find that he did not faithfully transmit all state information back. He focused on 7234, though he saw and mentioned 7137 as well. It seems that he filtered what he transmitted according to his incorrect understanding of the direction in which data flowed through the ports.

According to the theory of language use as joint activity (Clark, 1996), we can suppose that at the highest level the administrator and technical support were engaged in a joint project to find and fix the problem with the new player instance. Subordinate to this was the project to establish common understanding of which ports were listening on the maestro and player servers. Note that only the administrator could determine which ports were listening because only he had access to the actual computer systems. Technical support attempted to draw out the relevant information by asking about the ports. However, when technical support initiated the project to obtain information from the manual, the administrator did not take up the project. Almost all useful communication between them ended at that point, as it seems the administrator did not see this as worthwhile.

The administrator's interactions with technical support and the architect (***Arch***) involved joint projects to determine, understand, and fix the problem; yet the administrator performed all diagnostic and repair operations. This contrasts with the administrator's interaction with his colleague (***Colle***), in which the colleague could access the system independently. As shown, the colleague was confident of his understanding of the problem and of the path to solution, but his repeated pleas for the administrator to simply perform the operations were ineffective. In this case, it seems as if the administrator understood the joint project with his colleague to be similar to those with technical support and the architect: the establishment of mutual understanding so as to develop a solution together. It seems the colleague, however, understood the joint project to be the solution of the problem itself. Sensing this mismatch, the colleague resorted to explanations in the form of commands to be run, his increasing agitation expressed in capital letters and exclamation points in instant messages. Once the conversation switched to the phone, further explanation attempts were made. Here is where the colleague seems to have realized a further mismatch: rather than a mismatch in knowledge of the various ports settings, he realized that the administrator did not have a correct mental model of the system with which to understand the details of the ports. To debug the administrator's model of how the system was put together, the colleague merely stated that their initial understanding had been wrong. Only at this point did the administrator begin to engage in the project the colleague had been proposing all along, changing the port settings.

The joint project of fixing a problem was accomplished without establishing common understanding about many technical aspects of the situation. Movement of representational state about computer system parameters, whether correctly or incorrectly expressed, flowed among participants but did not affect the actual computer system until representations of the entire configuration itself were conveyed. Solving the problem required participants to coordinate activity around *system model* rather than around *system parameters*. The telephone (as medium) enabled this change in coordination whereas text-based messaging did not. Discourse by telephone had a different character than discourse by text messaging: telephone resulted in give and take and shifting of projects, whereas messaging resulted mainly in opposing positions. That is, the rich interaction afforded by the telephone enabled participants to coordinate information not only about the business at hand (setting the parameters properly) but also about deciding what to do (debugging the system model).

Conclusion

Support and maintenance of large-scale computer systems is rarely done by one person working alone. Given the size and complexity of systems, many people with many different expertise and skills are required to work together to keep systems running. Yet the establishment of common ground among participants in these tasks requires not only transmission of technical information but also establishment appropriate coordinated activity (joint projects) and management of attention. Our analysis suggests that information flow is moderated by whom or what people pay attention to, which in turn is moderated by discourse attributes influencing project initiation and uptake.

References

Anderson, E. (2002). *Researching system administration*. Unpublished doctoral dissertation. University of California, Berkeley.

Clark, H. H. (1996). *Using language*. Cambridge, England: Cambridge University Press.

Fairburn C., Wright P. & Fields R., (1999). Air traffic control as distributed joint activity: Using Clark's theory of language to understand collaborative working in ATC. In *Proceedings of the European Conference on Cognitive Science*.

Hutchins E. (1995). *Cognition in the Wild*. Cambridge, MA: MIT Press.

Woods, D. D. (1988). Coping with complexity: The psychology of human behavior in complex systems. In H. B. Goodstein & S. E. Olsen (Eds.) *Tasks, errors, and mental models*. London: Taylor & Francis, 128 – 148.