Virtual Organizing: Using Threads to Coordinate Distributed Work

JoAnne Yates
MIT Sloan School of Management
jyates@mit.edu

Wanda J. Orlikowski
MIT Sloan School of Management
wanda@mit.edu

Stephanie L. Woerner
MIT Sloan School of Management
woerner@mit.edu

Abstract

This paper explores the critical role of conversational threads in facilitating the ongoing, distributed work of one virtual organization. In studying the electronic mail exchanges of organizational members during one year, we found that they engaged in a range of threading activity to establish and maintain continuity, coherence, and coordination in their collaborative work over time. In particular, we found that organizational members relied on simple threads to focus their attention and action on a particular topic over a short period of time, concurrent threads to enable their participation in multiple topics at the same time, and compound threads to allow provisional settlement of key issues that were subsequently revisited over extended periods of time. We conclude by discussing the implications of conversational threads for research and practice of virtual organizing.

1. Introduction

In recent years, we have seen considerable interest (both in research and practice) in virtual ways of organizing work within and across firm boundaries, for example global virtual teams [4, 7] and geographically-distributed projects, communities and organizations [5]. Critical questions raised by these virtual forms of organizing include the following: What are effective means of coordinating work over time and across space? How do organizational members establish and maintain coherence and continuity in such distributed conditions? We explore these questions in this paper through an empirical study of a small software start-up company whose members were geographically dispersed in four different cities and three different time zones, and who relied primarily (though not exclusively) on electronic communication to conduct the work of their organization.

Our research uses the notion of conversational threads as an analytic lens to examine the electronic mail exchanges among the organizational members as they engaged in their distributed software development activities. In particular, we are interested in understanding how these members used conversational threads as a way of coordinating their ongoing work over time and across distance, and as a means of establishing and maintaining virtual coherence and continuity.

The use of conversational threads to obtain topical coherence is common in both on-line and off-line communications [8, 1]. Conversational threads have been defined in multiple ways, but a useful definition is provided by McDaniel, Olson and Magee [8, p. 41]: "We define a thread as 'a stream of conversation in which successive contributions continue a topic, following an initial contribution which introduces a new topic.'" In comparing the threading activity of face-to-face and computer-mediated communications, they find more threading by those participants engaged in computer-mediated interactions. In addition, they find that the threading activity of computer-mediated communications includes more concurrency than that of face-to-face interactions. The researchers define concurrency as the interleaving of contributions of different threads, and their finding indicates that participants in electronic communications tended to take part in multiple conversational threads at the same time. This notion of concurrency resembles "the intertwining threads of activity" evident in team interaction processes [6, p. 2], and the "crisscrossing of communicative chains" identified by Reder and Schwab [10, p.12], where they find that "the many switches and intertwinings of task and channel … are fundamental features of workplace communication."

In the research reported here, we studied the nature and use of conversational threads by the members of one organization. Before discussing our findings, we will provide background on the organization we studied, and describe the methods we used to analyze threading activity in its electronic communications. We next discuss the role of conversational threads in coordinating distributed activities over time and across distance, and conclude with implications for research and practice.

2. Site and Methods

LittleCompany (LC) is a small, primarily self-funded, software start-up company building a programming

language product, the LC system (names of the company, product, technology, and members have been disguised for confidentiality reasons). At the time of our study, three of the members (Dan, Keith, and Robert) worked full-time and two (Martin and Fred) worked part-time (the two part-time members eventually dropped out, some time after the period covered in this analysis). Members of the company were geographically dispersed; the five original members worked in four different cities across the U.S.A. and in three different time zones. This virtual company had no central physical location, as the members worked out of their homes or private offices. Although they were geographically dispersed, there were multiple existing interpersonal ties among the members of this company: three of the LC members – Dan, Keith and Fred – had gone to college together; Dan and Keith were friends and had written papers together; Keith and Robert were friends and had worked together; and Fred and Martin, who had worked together many years ago, were friends who lived in the same city and met regularly to play racquetball. The work of this company was done in collaboration among all five members. The primary form of communication was electronic mail, supported by phone calls and a few face-to-face meetings. During the period of our study, face-to-face meetings occurred only between pairs of LC members. Indeed, Keith did not meet Martin until well after the period of our study, and to this day Dan and Robert have never met Martin.

In this study, we analyzed the electronic mail archives belonging to one of the full-time members of LC who had saved all of his email dating from the inception of the company. While some members of LC began to organize the company several months earlier, our analysis of email starts in early 1997, when all members of the organization were in place and work on the LC system product began. We chose to end the analysis at the time of the alpha release of the product, approximately one year later.

For the analysis reported in this paper, we began by sorting all of the messages in the email archives into broad subject categories, and selecting all those messages related to the company, its members, and the product—a category consisting of 4,402 messages. In order to identify threads, we first cleaned the date and time data. As the email messages covered three time zones, we found that we could not use the local time stamped on each message to order the sets of messages into coherent threads. We thus converted the local time stamps to Greenwich Mean Time (GMT), which standardized the time each message was sent or received (an inspection of messages showed that the sent and the received times were generally quite close to each other). Having organized the email messages into chronological order, we then began grouping the email messages into provisional thread clusters, primarily by examining and matching subject lines. We occasionally grouped together messages that were chronologically close to each other

with subjects that didn't match exactly but which seemed related.

This process produced an initial set of 626 threads consisting of 2,215 email messages. We next read through the subject lines of the initial threads, flagging messages, which had been provisionally but questionably included in a thread, indicating places where an initial or interim message was missing in an exchange, and noting possible relationships among separate threads. We then went back to the email messages themselves, using close reading as the basis for adding messages to, subtracting messages from, and combining threads. In cases where the technical content made it difficult to judge whether a message belonged to a thread, we held iterative consultations with one of the LC members who served as a key informant. At the end of this stage, we had identified 560 threads composed of 2,431 messages, indicating that just over one half of the email messages we had analyzed could be clustered into distinguishable threads.

It is important to note that we are probably understating the total number of LC email threads as we are using the email files of only one LC member for this particular analysis. However, 77% of the messages in this thread data were sent to everyone in LC, leaving only 23% as exchanges with only one or two members. According to our key informant, these figures probably hold for the other members. As most of the email in LC originates from the three primary full-time members, we are missing primarily those exchanges that may have taken place between the other two full-time members. Our discussions with our key informant suggest, however, that we have captured the majority of LC email threads during the company's first year.

3. Threads in LC Communication

In our analysis of the LC electronic archive, we identified three distinct types of threading activity, all of which were used by the LC members to structure their ongoing work and interaction over time and space. The first, which we refer to as simple threading, resembles that identified in prior studies of electronic interaction [8, 1], and may be defined as temporally-bounded sequences of contributions on a single topic. The other two types of threading activity we identified represent different forms of interaction across threads. Concurrent threading refers to a practice of interleaving contributions from different threads at the same time, while compound threading refers to the practice of linking a series of simple threads on the same topic over long stretches of time. Where concurrent threads allowed the LC members to conduct work on multiple fronts at the same time, compound threads allowed them to pick up and continue work on a particular front after some pause or provisional settlement had ended

activity for a time. We now discuss each of these three forms of threading activity below.

Table 1: Threading Activity in LC Electronic Mail Archive

	Simple Threads	Compound Threads
Number of Threads	560	5
Number of Messages	2,431	650
Average Number of Messages per Thread	4.3	6.5
Median Number of Messages per Thread	3	4
Number of Simple Threads in Compound Threads	—	104
Shortest Thread (hh:mm:ss)	00:00:50	00:04:37
Longest Thread (dd:hh:mm:ss)	113:02:48:54	36:15:11:00
Average Elapsed Time (hh:mm:ss)	33:02:51	30:53:09
Median Elapsed Time (hh:mm:ss)	07:13:35	08:42:37
2-message threads	226	30
3-message threads	131	16
Longest Thread (number of messages)	32	28

Figure 1: Distribution of Simple Threads in LC Electronic Mail Archive
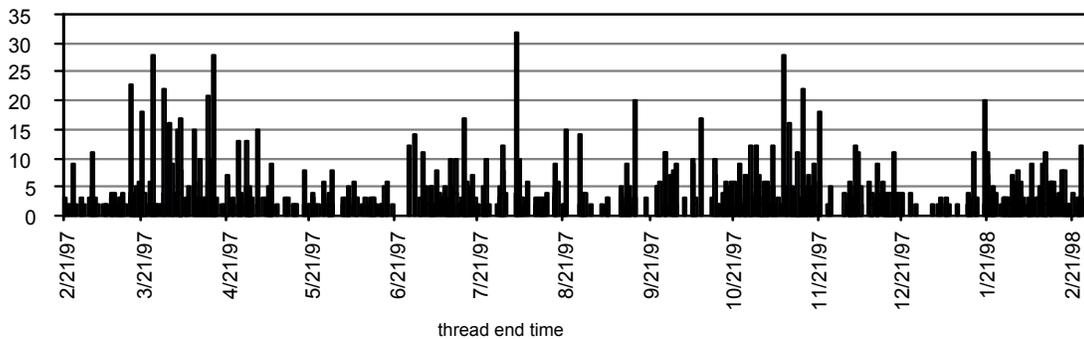
3.1 Simple Threads

We identified 560 simple threads in the data (see Table 1). These threads averaged 4.3 messages per thread, with a median of 3 messages per thread. Figure 1 shows the length and distribution of simple threads over time. The largest number of threads identified (40% of the total) consisted of only 2 messages, with another 23% consisting of 3 messages. These typical, brief threads included an inquiry and a response, with sometimes an additional comment from the originator of the thread or occasionally from a second party. The subject line in the second and third messages would normally be "Re:" plus the subject line of the original message, and the initial

message would be reproduced, either as a whole, with the response following it, or in pieces, with specific responses to it. In the example below, the 2-message thread is used to clear up confusion about source code control that has evidently arisen in an earlier conversation:

Date: Fri, 14 Mar 1997 15:55
From: Keith
To: all
Subject: backups

I would like to correct a fallacy that martin seems to have about cvs.

Each person's machine has only a snapshot of our systems. By snapshot I mean a copy of the latest work in progress files. No history is kept on any client.

Homer (and my machine because I back up the server) is the only place that has the archival versions of files (stored in rcs form).

The combination of my isp's backups and my weekly backup could be considered enough protection (except in the unlikely circumstance of a regional disaster like a hurricane or the local reactor going non linear.)

…

keith

From: Martin
To: all
Subject: RE: backups
Date: Fri, 14 Mar 1997 23:35

> I would like to correct a fallacy that Martin seems to have about cvs.

Keith, I'm aware that we do not bring RCS repositories with each CVS operation down to our machines. Each of us has only "work in progress." I was only pointing out that in case of emergency we also ought to consider it when restoring the LC assets.

I agree with you that we need somebody else to backup the server on a weekly basis.

The exchange helps provide coherence by getting all LC members "on the same page." In the next example, a 3-message thread is used to coordinate a change in time for a regular phone conference:

Date: Sun, 14 Sep 97 15:22
From: Fred
To: all
Subject: next week's conference call

Gentlemen:

Since Martin is unavailable on Wednesday at the usual time I propose that we move the weekly conference call to Tuesday at (1500/1300/1200) hours instead. This does not conflict with any of the constraints that people have noted in email and in telephone conversations with Keith, Martin, and Dan they have said that this was an acceptable time for them.

Please send email around indicating your agreement or veto.

I have two agenda items for the meeting:
 * patent issues: choice and procedure

 * foreign-function strategy

regards..fred

Date: Mon, 15 Sep 1997 13:42
From: Robert
To: Fred
Cc: all
Subject: Re: next week's conference call

Fred wrote:
>
> Gentlemen:
>
> Since Martin is unavailable on Wednesday at the usual time I propose that we move the weekly conference call to Tuesday at (1500/1300/1200) hours instead.

This is okay with me.
--
Robert

To: all
From: Dan
Subject: Re: next week's conference call

At 01:42 PM 9/15/97, Robert wrote:
>Fred wrote:
>>
>> Gentlemen:
>>
>> Since Martin is unavailable on Wednesday at the usual time I propose that we move the weekly conference call to Tuesday at (1500/1300/1200) hours instead.
>
>This is okay with me.

Works for me too.

Dan

These 2- and 3-message threads were normally concluded within a single day, with an average elapsed time of 11 hours and 23 minutes, and a median elapsed time of 2 hours and 50 minutes. The shortest such interaction took less than one minute, indicating that the recipient of the original message was on line and replied immediately and briefly to it. Such cases indicate almost synchronous interactions. Typically, however, the short, 2-3 message interactions were clearly asynchronous, taking several hours.

Threads sometimes centered around personal, rather than technical or business, issues. The following thread was organized around a personal subject, but with references to some technical matters mixed in:

To: all
From: Dan
Subject: fyi

In case anyone is wondering, we just had a shitpile of snow. Keith's power is out, and my ISP is unusually hard to reach.
Dan

From: Robert
To: all
Subject: Re: fyi

Dan wrote:
> In case anyone is wondering, we just had a shitpile of
snow. Keith's power is out, and my ISP is unusually
hard to reach.

Does LittleCompany have snow days?
--
robert

To: all
From: Dan
Subject: Re: fyi

At 10:26 AM 4/1/97, Robert wrote:
>Does LittleCompany have snow days?

Don't think so. It picked a good day to snow; I made a
couple of small fixes to instruction-form (form = reg/reg
vs reg/spill vs reg/mem) selection, and have been
testing them while I shoveled snow.

For reference, a "shitpile" is somewhere between 18
and 24 inches, packed. A pine tree in the yard next
door broke off about 12 feet off the ground, an
arborvitae in another neighbor's yard removed itself,
roots and all, from the ground. Down the street there's
a maple tree with two big branches down, across the
front walk of a guy who's about 80. But, lucky us, all our
services come in below ground.
Tonight it is supposed to get to 20, but tomorrow and
the next day it is supposed to be warm.

        Dan

The joke about whether LC has snow days refers to the
business side of the tiny company, while the definition of
"shitpile" is a humorous reference to programmers'
tendency to specify terms.

    While the majority of threads were short, a few were
quite significant in length, with 50 of the 560 threads
including 10 or more messages, and 10 threads including
20 or more messages. The longer threads tended to be
extended discussions attempting to thrash out company
strategy, identify and fix bugs in the LC system, or
resolve disagreements about aspects of the LC system.
They could include as many as 32 messages and, at the
extreme, last more than 100 days. The average elapsed
time, however, was less than two days, indicating that
even with the longer threads, the interactions typically
were completed rapidly.   One 28-message thread
concerning a system crash, for example, took place over a
period of two and a half days and involved only the three
primary individuals. During this time they ran various
traces to diagnose the problem, identified a "smoking
gun," and finally fixed the bug, tested the fix, and
committed new code to the LC system site.

    Another long thread, this time involving a conflict,
began with a request for clarification of two words in a
specification for part of the system laid out by one
participant. From this point it ultimately escalated into
exchanges with as many as 7 levels of indented embedded

messages (i.e., an initial statement or question embedded
within a response, embedded within a counter response,
etc.), all within the current message. Both parties in the
conflict (Robert was in conflict with Keith, with Dan
occasionally supporting the latter) claimed not to
understand what the other meant. The conflict grew
sufficiently serious that Keith resorted to back channel
phone calls to defuse the problem somewhat. The next
message in the thread, from Keith to Robert, was lower in
key, though it continued to respond to the earlier
discussion and still exhibited occasional testiness, as in
the following response to one point made by Robert:

    This is pretty obscure, and if I felt like causing a fight, I
    would stand firm on my reading. But it is easy to fix so I
    will.

    The message ends with a combination of an apology
and a plea for a changed approach to documentation of
the system:

    I can see that you may think that I spent a lot of time
    trying to mess with you. I really did not. The problem was
    that it never occurred to me that all names had the section
    in them since I have never seen a system where this was
    done….
    …
    I would like to say that there are three problems with
    language lawyer documents. They are time consuming to
    produce, they are time consuming to read, when there is a
    misunderstanding there is always an argument. LC really
    does not have the time to spend on any of this.

    Please, for sanity sake, try to make your stuff more
    accessible. The company really depends on this.

    The thread as a whole ended with several more
messages clearing up the final open issues, and in a more
neutral tone. Such conflicts were rare in the archive as a
whole, and were primarily restricted to the initial months
of the company's life when the participants were learning
how to work together and in a distributed fashion.

3.2. Concurrent Threads

    Given the complexity of the work engaged in by the
LC members, it is not surprising that we found evidence
that they dealt concurrently with multiple subjects. Some
concurrency happened *within* messages and threads, as in
cases where a message discussing a technical problem
ended or began with comments about something personal,
for example:

    From: Robert
    To: Dan
    Cc: LC all
    Subject: Re: ssh problems?
    Dan wrote:
    >
    > At 07:29 AM 7/29/97 -0600, Robert wrote:
    > >the server is not asking for the rsa password, only
    the one for homer. Anyone know what is up?
    >
    > No idea. It just worked fine for me, both update and
    commit.

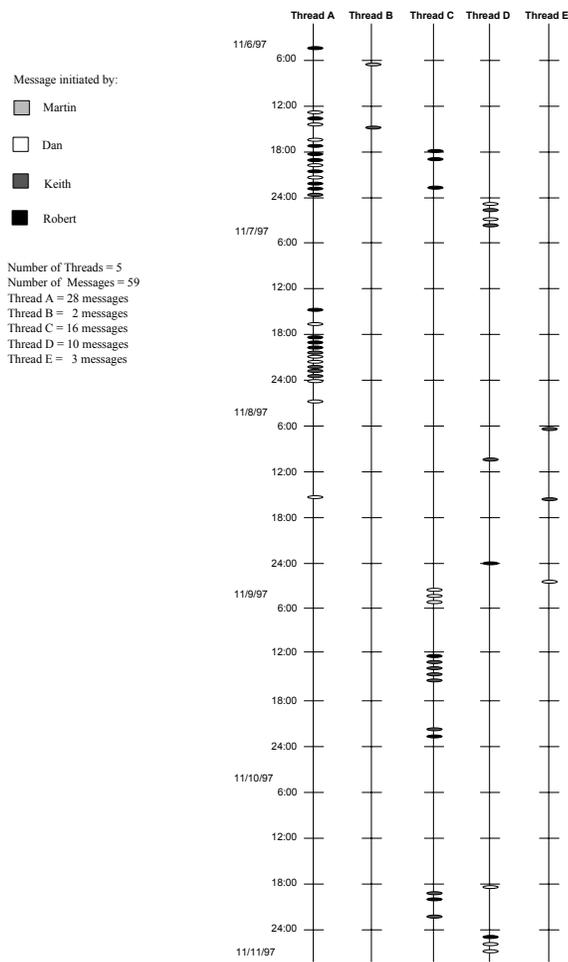I just did it again and it worked as normal.

We are in the middle of moving and flooding. We were up until 1am last night playing hydraulic engineers. There was waist deep water across the street and it was running by our house at about basement window level.

More rain forecast for tonight, it could be real bad.
--
robert

This message is part of a thread dealing with a technical issue, but at the end of this message Robert concurrently informed the others about his personal situation. As the thread on the technical issue continued, the personal problem with flooding was brought up in two more messages, but then dropped. While a certain amount of concurrency of personal with technical issues appeared within specific messages and threads, in general, LC members kept to one primary work-related topic per message, and thus dealt with multiple issues by carrying on multiple conversational threads concurrently.

Figure 2: Thread Concurrency (times in GMT)

Thread A   Thread B   Thread C   Thread D   Thread E

11/6/97
6:00

Message initiated by:

☐ Martin

☐ Dan

■ Keith

■ Robert

Number of Threads = 5
Number of Messages = 59
Thread A = 28 messages
Thread B = 2 messages
Thread C = 16 messages
Thread D = 10 messages
Thread E = 3 messages

12:00

18:00

24:00

11/7/97
6:00

12:00

18:00

24:00

11/8/97
6:00

12:00

18:00

24:00

11/9/97
6:00

12:00

18:00

24:00

11/10/97
6:00

12:00

18:00

24:00

11/11/97

Such concurrent threads, involving the temporal interleaving of one thread by messages from another thread, were common. Indeed, 93% of all the threads identified above overlapped with at least one other thread, and at some points up to seven threads ran concurrently. These concurrent threads allowed the LC members to carry on parallel discussions of a variety of issues.

Figure 2 shows how five simple threads interlinked during one particularly busy 5-day period. Thread A, the 28-message thread about the system crash mentioned above, involved intensive work and interaction among the LC members. Shortly after that thread began, a brief, 2-message thread (B) dealt with system benchmarking, an unrelated issue. As the investigation of the crash proceeded, the participants discovered that they were dealing with multiple bugs. Two other threads, C and D, broke off to deal with these bugs, both continuing after Thread A ended. Yet another brief thread, Thread E, about an unrelated technical issue, began just at the end of Thread A, and before the end of Threads C and D. In this particular case, the placement of a new version of the LC system on the server, revealed three different bugs, in turn triggering three of the five concurrent threads. Getting the new version of the system up and running required solving all three bugs, and the concurrent threads helped the key LC members coordinate their efforts towards solving all of the bugs at the same time, shortening the total elapsed time required to get the system running. At the same time, the two additional (and shorter) threads dealt with unrelated technical issues.

One final phenomenon related to these concurrent threads is worth noting. In the crash that precipitated the discovery of three bugs, one thread diverged into three concurrent threads, with two breaking off from the initial thread as the nature of the three bugs became clearer. Because of the technical nature of the subject matter, we came to understand this divergence only through iterative discussions with our key informant over this flurry of messages. Nevertheless, this divergence of a single thread into multiple concurrent threads seems likely to occur in other discussions of a complex issue (whether technical or business related), as different discussions emerge out of different aspects of the issue. While a single thread might mix personal and technical topics (as we saw earlier in the cases of concurrency within a message), the LC members were careful to separate out technical issues into their components as clearly as possible. Thus one thread quickly diverged into three in dealing with the crash.

3.3. Compound Threads

Compound threads, an unexpected phenomenon emerging from the data, connected related work over long periods of time. Such compound threads were made up of multiple, sequential (rather than concurrent) simple threads, loosely connected by a common subject and

occasional references to previous discussions. As shown in Table 1, we identified 5 different compound threads in our data, with each compound thread including from 5 to 37 threads and from 20 to 294 total messages (these figures understate the total because some compound threads extended beyond the first year). Compound threads organized discussion about key technical aspects of the LC system over extended periods of time. When asked to reflect about what characterized the use of compound threads, our key informant explained:

> There are four types or qualities of the things that we came back to again and again:
>
> 1. We didn't understand it in the beginning but we had to start. We would come back when we hit a problem. ....
>
> 2. The problem is so hard that it isn't the most important thing to implement it correctly. You do a good enough job and then come back. You don't want to prematurely optimize because that pessimizes the use of your time. You want to see if it is good enough. ....
>
> 3. The problems change, the specifications change, or the requirements change. ....
>
> 4. Getting something right or changing it requires cooperation and coordination among everybody. ...

That is, compound threads dealt with issues that were central to their work of building the LC system and that recurred or evolved as the system development continued.

The longest of these compound threads, which concerned building the system's linker, extended over almost the entire period studied (see Figure 3), and, in fact, into subsequent years. The subject of this compound thread had several of the above qualities. The linker had to be started and provisionally stabilized in order to make progress on other parts of the system. The provisional nature of the early stage linker is made clear in the first thread, in which the initial specifications, proposed by Robert, were discussed among the three key LC members. The following exchange between Robert and Keith appears in the first half of the thread:

> > - Does this really have enough power to express the things we want?
>
> I think this is ok for now.

A few messages later in the thread, Robert describes a case the system might need to deal with in the future, then explains:

> While I don't want to do this right now, I'd like to avoid introducing stuff into the linker spec that makes this impossible.

At this stage, then, we can see that the linker issue reflects the first two of our key informant's criteria (the need to start somewhere and the need to keep options open). Moreover, in the last messages of this first thread, Robert refers directly to the provisional nature of their work on the linker at this time:

> I have hopes that this is the last linker spec for a while. Changes are pretty minor, keith is getting tired of arguing.

Thus, a specification for the linker is "stabilized-for-now" [12], with the clear understanding that it will be revisited and changed as the system develops.

After a period of stability, then, we would expect further changes. Indeed, in this case the stability of the linker did not last long. The next several threads of the compound thread occurred within the next few weeks, and started with questions about what the provisional linker specification implied for other parts of the system or for specific situations that might arise in the future. One of these, for example, started with a message with subject line "latest linker spec" and a statement that the constraints in the linker created problems in making the LC system work on different types of computer hardware. Several of these threads escalated into sharp disagreements in these early months of the company's existence.

Figure 3. Distribution of Threads in *Linker* Compound Thread
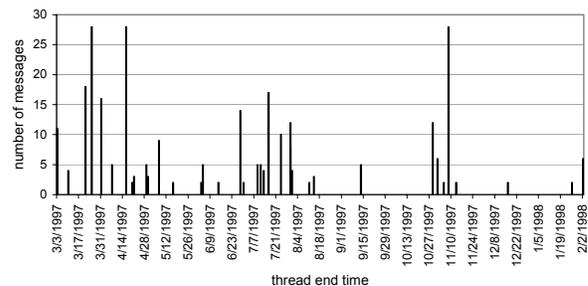
As is clear from Figure 3, linker was the subject of extensive discussion throughout March and April of 1997, continuing into May at a lower level. Even the subject lines on some of the threads revealed the provisional nature of the linker:

> Re: linker versions
> Re: getting things started
> Re: one last time
> Re: again

By the end of this intense period of discussion, however, the linker issue was adequately resolved for the moment, allowing other work to continue.

Still, questions and requests for change in the linker would continue to be triggered by clarifications of constraints imposed by the current form, other changes in the system, performance problems as the rest of the system got larger, and bugs revealed by all of the above. The performance issues that accompanied system growth were first hinted at in Dan's reference to the "obesity of our current system" in a message speculating about which

direction LC should take in developing one part of the system. In two subsequent messages Dan and Martin had the following exchange, triggered by Martin's question:

> Mon Apr 28 23:56 1997
> To: Martin
> From: Dan
> Subject: RE: types in the linker.
>
> At 12:42 PM 4/28/97 you wrote:
> >Dan, the subject may be discussed in detail in the later e-mails, but I wonder why do you think that our system is obese?
>
> Because I run out of memory compiling some of our files, and we aren't even doing all of code generation, let along optimization.
> Remember that I'm running on a very big machine -- 64Mb of memory, with about 40Mb allowed for use by the interpreter.

This theme reemerged in subsequent months, and performance problems generated by system growth frequently triggered new threads within this compound thread. For example, one thread contributing to the compound thread began with the following performance complaint from Martin:

> I have 120 Mhz pentium with 40Mb RAM (and a reasonable swap space). When I run anything related to TEST (......) the system seems to be performing at acceptable speed. However, when I run LINKTEST (....) the system starts swapping like crazy. …
>
> Is there a way to reduce the memory requirements of the system, so it could run more reasonably on my machine? Or alternatively, could we introduce a subset of tests that would perform reasonably on my machine?

Subsequently, speed of testing and linking became a problem for everyone. At one point, Keith made the following plea to Robert, who was the primary person responsible for the linker:

> Robert, you have to understand that dan and I are dying here. We cannot make small changes and test things in a reasonable way. I spent almost all day yesterday waiting on links. This is a serious problem and we really need some relief.

An update to the linker seemed to improve matters, but soon the performance problems reemerged, initiating another thread including the following exchange:

> > The big black hole is the size of the inhale. Why are we reading so much more stuff?
> > --
> >robert
>
> Three reasons:
> 1) Now that dan is generating code for exceptions, all of the types mentioned in the handler clauses are now inhaled.
> 2) …
> 3) …

These developments were natural ones in the growth of the system, but they triggered new discussions about and changes in the linker, followed by tests to measure gains in speed. LC members revisited the linker issue to deal with performance problems related to system growth during the summer, then again in late October and early November. Indeed, one discussion in June stated the need to be explicitly provisional at this point, to allow for later changes to reflect customer, rather than developer, needs:

> In the long run, this may require more recompilation than the customer is willing to deal with. If we try to back off to the point where we follow the … rules at least to link time, we will give up some run time performance….
>
> Dan and I have decided to know everything because this is the way to get some good performance in the short term. However, we are very cognizant of identifying these points so we will later be able to turn the knob back if the market dictates it.
>
> If (When) we decide to turn that knob back, the linker will have to be enhanced because it is there where we will need to determine the object layout.

It clearly was not a surprise to LC members that the linker issue would keep coming up as the system evolved.

Another frequent trigger to new threads of the linker compound thread were bugs, most of which arose in trying to discover the reason for system errors or crashes when new elements were loaded onto the server. Such threads would start by someone reporting an error message or crash and asking for help in pinpointing its cause. In one case, for example, Dan described the function he had tried to execute in the subject line, then wrote as follows:

> It does not work. After 14 minutes (elapsed) the interpreter running the linker falls over with an NT stack trace. I am going to retrofit some options for during on/off checking code, and I'm going to look into some ways that I might generate fewer relocations.
>
> But, we've got a big problem.

As it turns out, this one was solved relatively easily, but some of them were not. Thread A of the five concurrent threads discussed above is a linker thread, for example, triggered by a system crash during linking and subsequently discovered to involve at least three bugs.

An examination of the linker compound thread, then, supports our key informant's analysis of the common triggers for such compounding of threads at LC, as well as revealing some of the factors that prompted their return to the issue. The linker posed a difficult and large piece of the whole LC system they were building, and problems with it could be only provisionally settled, to be revisited as the system developed. Moreover, while one person was in charge of coding the linker, this piece of the system was so interrelated with other pieces that changes to it inevitably involved at least the three full-time LC members, and sometimes the part-time ones, as well. As LC members worked towards alpha release of their product, they periodically revisited the linker issue,

triggered by consideration of constraints imposed on other parts of the system, performance problems created by system growth, and crashes with the subsequent search for bugs. This and other compound threads, linked primarily by a common topic, loosely connected the ongoing work discussions in this virtual organization over time.

4. Discussion and Implications

Our study of the LC electronic mail archive reveals that the members of a small, geographically distributed virtual organization/team, collaborating to build both a software system and a firm, engaged in a variety of threading activity to structure their ongoing electronic interactions. While prior studies of electronic communication have identified the tendency for interactions to coalesce around conversational threads, and for such interactions to become interleaved into concurrent conversations, the critical role of threading in facilitating the continuity, coherence, and coordination of the distributed work of virtual teams and organizations has not been highlighted. While drawing attention to this coordinating role of threading, our study has also identified the interesting new phenomenon of compound threading—the practice of using a loosely linked series of conversational threads on the same topic to pursue an ongoing discussion. Furthermore, we found that a range of threading activity—with simple threads often serving as components of concurrent and compound threads—was used by members of this virtual organization to accomplish their distributed, collaborative work over time.

Simple threads are tightly coupled sequences of interactions that focus participants' attention and action on a particular issue. The interdependence among the messages making up the thread (typically signaled via a repeated subject line and/or an embedded passage from a previous message) establishes and maintains coherence in members' communication and facilitates coordination of their activities around common issues.

Concurrent threads are composed of multiple simple threads whose constituent messages are interleaved over time, thus reflecting members' engagement in multiple conversations at the same time. The types of interleaving in the LC communication varied: some represented parallel discussions on unrelated technical topics; some represented interactions on related but separable technical topics; and others reflected the intersection of personal matters with work-related topics. The use of concurrency in electronic interactions gives members the opportunity to raise and work on multiple different issues and topics at the same time. As noted by McDaniel, Olson and Magee [8], increased concurrency is particularly pronounced when members interact via asynchronous electronic tools, as the delay between contributions provides time for additional conversations. Increased

concurrency may decrease the amount of elapsed time required to complete a set of tasks, and it may allow members to make connections across different conversations. We might speculate that such connections would be both valuable (e.g., when serendipitous coupling produces a more integrative view of an issue) and distracting (e.g., when concurrent examination of multiple issues causes confusion or unnecessarily increases complexity).

Concurrent conversations on related technical topics may arise out of what we term thread *divergence*, when an initial thread breaks into multiple concurrent threads as participants realize that the issues require different conversational streams. We are intrigued by the possibility that an analogous phenomenon of thread *convergence* may exist in such task-oriented communication, though we did not identify it in this data. Clearly, synchronous off-line conversations may both diverge and converge, and we would also expect to find both forms in asynchronous on-line communication.

Compound threads are simple threads linked together over time by members' recurrent attention to and discussion of the same topic. In the LC archive, the compound threads tended to represent subjects of central importance to the work of the virtual organization, in this case, the development and testing of the LC system. Compound threads reveal the importance—in the ongoing work of any project—of being able to halt a particular interaction for a period of time, settling on some provisionally stabilized agreement, thus making possible the pursuit of other work-related interactions. Provisional settlements [3] allow the (temporary) reconciliation of uncertain, incomplete, or incommensurable requirements to facilitate work to move forward on other fronts, while acknowledging that further work on the provisionally settled area remains to be done in the future. The use of compound threads by organizational members enables them to coordinate their interactions and maintain continuity in their work as it is distributed over time and across dispersed locales.

The exploratory research reported here has revealed the role of a range of threading activity in facilitating the coordination, coherence, and continuity of distributed electronic collaborations. Our study is based primarily on a close reading of the messages in the electronic mail archive, complemented by a statistical analysis of the identified threads. More insights into the distributed electronic collaborations of the LC members will be obtained by pursuing additional research methods, particularly in-depth interviews with the LC members, and content analysis of messages in the electronic archive.

Additional research is also needed to examine this threading activity in other empirical settings. While we would expect such concurrency and compounding to occur in other types of virtual environments, the

frequency and nature of concurrent and compound threads may look somewhat different depending on differences in circumstances and goals.

This study and the related speculations have implications for both theory and practice. Theoretically, we suggest the potential usefulness of the concept of *communicative ecology.[1]* The communicative ecology of a particular virtual team, organization, or community might be identified by the types and frequencies of its communicative practices, such as threading activities. Such an ecology would reflect the influence of factors such as: 1) whether members are engaged in a common task, the components of which need to be coordinated (a group with minimal coordination demands would be less likely to have a use for concurrent threads); 2) whether the group or community is interacting over an extended period of time (compounding of threads is less likely in discrete interactions such as those through instant messaging than in ongoing interaction via email); 3) whether the media in use support synchronous or asynchronous communication (asynchronous media such as email are currently more capable of supporting multiple concurrent conversations than such synchronous media as telephone conferencing); and 4) whether members share linguistic and cultural backgrounds (groups that must contend with multiple languages and/or cultures may find it more difficult to carry on concurrent threads without confusion).

In terms of practice, we believe that the success of this group in negotiating the first year (and subsequent years) of virtual organizing provides some lessons for those working this way. The individuals making up LC had a complex web of personal, educational, and work experience connections that made it easier for them to engage in various types of communicative activities relatively implicitly. The group explicitly agreed on weekly phone conferences, but the varieties of threading activity were conducted quite implicitly. In cases where less commonality (linguistic, cultural, etc,) exists, however, explicit agreements about elements of the communication ecology may be useful in facilitating virtual work [9]. Raffoni [11], for example, suggests that creating an explicit communication plan will aid in managing a virtual organization. Such a plan may need to consider threading activity and expectations for how quickly members will respond to asynchronous messages. By considering such decisions explicitly, virtual teams and organizations may improve their ability to coordinate their work, especially task-oriented activities, over time.

In closing, we believe it is only by understanding the whole communicative ecology enacted by members of a community or organization that researchers and practitioners can come to understand how the conversational threads created in electronic communications weave "the fabric of activity" [10, p.11] that constitutes virtual organizing.

5. References

[1] Donath, J., Karahalios, K. and Viegas, F., "Visualizing Conversation." In *Proceedings of the 32nd Hawaii International Conference on System Sciences,* IEEE Press, Hawaii, January 1999, pp.1-9.

[2] Erickson, T., "Making Sense of Computer-Mediated Communication (CMC): Conversations as Genres, CMC systems as Genre Ecologies," In *Proceedings of the 33rd Hawaii International Conference on System Sciences,* IEEE Press, Hawaii, January 2000, pp. 1-10.

[3] Girard, M. and Stark, D., "Distributing Intelligence and Organizing Diversity in New Media Projects. *Environment and Planning A*., Forthcoming.

[4] Lipnack, J. and Stamps, J., *Virtual Teams--Reaching Across Space, Time, and Organizations with Technology,* New York: John Wiley & Sons, 1997.

[5] Markus, M.L., Manville, B., and Agres, C.E., "What Makes a Virtual Organization Work?" *Sloan Management Review*, 42, 1, 2000, pp. 13-26.

[6] Massey, A.P., Montoya-Weiss, M. and Hung, Yu-Ting C., "Synchronizing Pace in Asynchronous Global Virtual Project Teams," In *Proceedings of the 35th Hawaii International Conference on System Sciences,* IEEE Press, Hawaii, January, 2002, pp. 154-163.

[7] Maznevski, M.L. and Chudoba, K.M., "Bridging Space Over Time: Global Virtual Team Dynamics and Effectiveness." *Organization Science,* 11, 5, 2000, pp. 473-492.

[8] McDaniel, S., Olson, G., and Magee, J., "Identifying and Analyzing Multiple Threads in Computer-Mediated and Face-to-Face Conversations," In *Proceedings of the Conference on Computer Supported Cooperative Work,* ACM Press, Cambridge, MA, 1996, pp. 39-47.

[9] Orlikowski, W. J., Yates, J., & Fonstad, N., "Sloan 2001: A Virtual Odyssey," In I. Zigurs, & L. Chidambaram (Eds.), *Our Virtual World: The Transformation of Work, Play, and Life Via Technology*, Hershey, PA: Idea Group Publishing, 2001,pp. 191-218.

[10] Reder, S., and Schwab, R.G., "The Temporal Structure of Cooperative Activity." In *Proceedings of the Conference on Computer Supported Cooperative Work,* ACM Press, Los Angeles, CA, 1990, pp. 303-316.

[11] Raffoni, M., "Managing Your Virtual Company: Create a Communication Plan," *Harvard Management Communication Letter*, April 2000, pp. 7-8.

[12] Schryer, C.F., "Records as Genres," *Written Communication,* 10, 1993, pp. 200-234.

---

[1] This notion draws on Erickson's [2] idea of genre ecologies.