The Sun Never Sets

on Distributed Development

People around the world can work around the clock on a distributed project, but the real challenge lies in taming the social dynamics.

More and more software development is being distributed across greater and greater distances. The motives are varied, but one of the most predominant is the effort to keep costs down. As talent is where you find it, why not use it where you find it, rather than spending the money to relocate it to some ostensibly more "central" location? The increasing ubiquity of the Internet is making far-flung talent ever-more accessible.

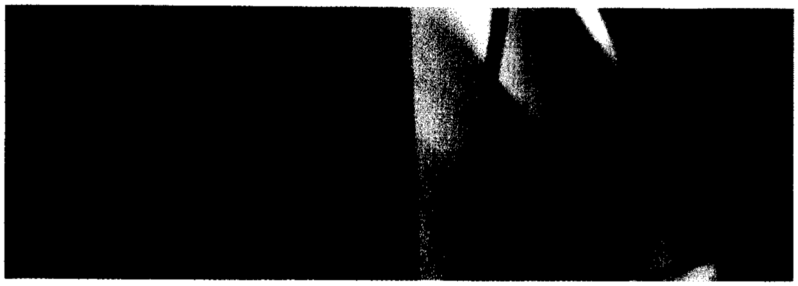
That makes sense as far as it goes—but it's an approach that potentially has a number of side effects even less tangible than the monetary savings. Large projects with widely dispersed participants have problems that are unknown in projects in which all the people involved work in the same building. If project planning is forward-looking enough to be concerned about potential savings, in a perfect world it certainly should also be aware of the other less tangible costs that will be incurred. Alas, it's not a perfect world, and so the same issues are encountered over and over again as organizations dip their toes in the distributed pool.

There is always the issue of coordinating all the developers, so as to avoid having people who are working on the same part of a project inadvertently interfering with each other. The challenge of coordinating the project can be at least partially addressed with technical solutions. Much less of a science is understanding and influencing the social dynamics of a highly diverse and distributed group of people, some of whom may have never even met, but all of whom must nevertheless work together. That is the more difficult issue to understand—and the most difficult issue to solve.

There are two main causes of the problems peculiar to distributed development, and they're a rock and a hard place:

- Geographical distribution across space includes *temporal* distribution across time zones.
- People in different regions often have different cultures, languages, or both.

The latter is worthy of an in-depth study in its own right, so I'm going to focus primarily on the former.



The Sun Never Sets

on Distributed Development

A lot of the problems of distributed development are direct outgrowths of the fact that humans are social animals. When we are alone in a room, we have a tendency to be self-centered—even if we're communicating online with a dozen other people at the moment—so seeing things from the other person's point of view is not automatic. If we're actually face-to-face with someone, our upbringing triggers a particular set of social behavior patterns. Generally, no such habits are formed when the presence of others is virtual rather than physical; it usually requires a conscious effort for us to behave as though people are present in the flesh. Since there's no one else in the room whose behavior we can imitate and learn from, the burden of teaching ourselves new habits is ours alone.

Over the past few years, as a participant in distributed development processes and providing voluntary realtime support through IRC (Internet Relay Chat), I have personally encountered all of these issues—and fallen into their traps more than once. Presumably, not all people are as fallible (though I've yet to encounter any that aren't), so to avoid absolutisms I've peppered this article with a sort of "in many cases" leavening. That said, in my experience the better wording would be "in most cases."

THE ISSUES

Distribution across time zones virtually forces a non-realtime means of communication, such as e-mail or text-entry conferencing tools. These place their own burdens on communication even when all the participants are in the same time zone and culture and speaking the same language. The remainder of this article focuses specifically on the issues of working almost exclusively with such impersonal media—e-mail in particular.

Serial and Synchronous Communication. Heated discussions using e-mail can often be extraordinarily stressful because of the very nature of the medium. Communication is by atomic content-packets, so you can't receive partial e-mail messages. You can't respond until you have something to which you can respond. This "taking turns" is the synchronous nature of the medium, and it's sufficiently different from our normal mode of discussion to derail a lot of our usual behaviors and habits.

How does an e-mail exchange differ from one held face to face? For one thing, the in-person participants can interrupt each other to correct mistaken impressions or to short-circuit confusion and long-winded digressions. It's one of the basic characteristics of in-person exchanges; it doesn't matter if it's in a team meeting, on a bus, or during a luncheon.

That doesn't happen in e-mail; by the time you read someone's message, the sender has already finished composing it; you can't interrupt before the message is completed and dispatched. And if you happen to disagree, or see that your correspondent is wandering into strange roads under some misapprehension, it's easy to start fuming and mentally drafting a scathing response as you read through the monolithic message you received. The other person, however, isn't going to "hear" you until long after making the original remarks. If the remarks were made with no thought of being controversial or offensive, an acerbic response from you could likely lead to an escalation of emotions.

Related to this is the effect of the impersonal nature of the medium. Even with the use of emoticons,[1] the inability to detect each other's body language tends to sensitize us and put us on our guard, making it easy to take offense. Adding "Just kidding!" may or may not have the same disarming effect in e-mail as a sincere grin in person; it depends on the people involved and their history with each other. Maybe one paragraph was just in fun; but if the others irritated you, which ones are you likely to respond to?

Internet Time. As people devote more time to being online, they tend to become more accustomed to quick response time. Transmission speed has come a long way in a very short time. Consider that in 1989 I was speaking on the telephone to someone a thousand miles away and sent him an e-mail message. When we heard the beep on his end about 15 minutes later, we were both astonished that it had arrived so quickly. What was remarkable then is taken for granted now. It's not at all unusual to be stopped by a problem, send a message asking for help, and have an answer by return e-mail before getting back with your next cup of coffee.

This is convenient for the actual collaborative development effort, but problems arise when it spills over into the decision-making process. There is a distinct tendency to expect everyone to be as responsive—or at least to overlook the possibility that they aren't—and to assign deadlines and decision points accordingly.

This is an example of the error of *ad nuntium* (responding directly to the content of a message, discussed in the next section on Flame Wars); the effort to make things happen quickly often fails to consider that not all participants are in the same time zone, or may be on vacation, or may be enjoying a regional holiday. And if things proceed, when those who were unable to participate before the deadline return, the subsequent meta-discussions and *flamage* (emotional e-mail outbursts) can be more damaging to the effort than extending the deadline in the first place would have been.

The expectation of this near-instant gratification, and the lack of patience it fosters, can lead directly to friction and bad feelings. If you send a message with a question to someone and don't receive a response within an hour or so, you might (and some people do) give up and send it to someone else. When the first person gets back from lunch, or the bank, or (in the case of differing time zones) comes in to work, and finds out that you didn't allow enough time to respond, it's quite possible your original correspondent will look upon you with something other than pure benevolence.

Heated discussions held in e-mail often are extraordinarily stressful.

There's yet another aspect of the conflict between speeded-up expectations and the demands of the real world. E-mail messages are often very much to the point of a particular issue; an individual may have several distinct "conversations" in progress at once. When we timeshare from one topic to the next, it gives us an opportunity to take a mental breath and refocus our attention—so it can be especially jarring to come back to a conversation and find the tone has become less than convivial. This sudden unexpected plunge into an emotional whirlpool can make our reactions stronger than they might otherwise be, just as a shower of room-temperature water feels hot at first when we've just come in from a snowstorm.

Flame Wars. These jarring exchanges of acrimony are an almost inevitable consequence of the previous two issues. Participants feel attacked—either personally or ideologically—by sequences of messages made uninterruptible and monolithic by the serial nature of the medium, or perhaps they feel slighted or disenfranchised because things progressed quickly when they weren't able to keep up and make their views known.

A well-known form (and cause) of flamage is the *ad hominem* attack, mounted against one personally rather than against one's position or arguments. It almost always results in responses in kind, and the tensions start rapidly spiraling upward. First the issues become thinly veiled smokescreens as the participants dissect each other; then, some particularly virulent flame wars can end up dispensing with any pretense of being anything else.

Curiously enough, directing your comments specifically at the content of a message (ad nuntium), rather than the sender, can be almost as bad. It's very common to be impersonally abrupt or even vicious, and forget that you're not addressing a computer—there's a person on the other end, after all. So even if you meant no slight or offense to the author of the message to which you're replying, it may not appear that way on the other side of the screen.

One reason flame wars can be particularly divisive and disruptive is the possibility of "spawning a fork"—that is, one side or the other starting an independent effort. That's an occasional occurrence in the distributed open-software development projects, but it probably isn't a major concern if the work is being done entirely in-house.

Staying Abreast. Environments that permit in-person meetings

The Sun Never Sets on Distributed Development

have a built-in way of keeping everyone abreast of the current status of various aspects of the project. A widely distributed collection of individuals working separately, however, can easily lose track of each others' work and progress. The two pathological end-points of the spectrum of consequences are:

- People duplicating work because they don't know someone else is already doing it (or has done it).
- Some tasks failing to get done because everyone assumes someone else is taking care of them.

If the project work truly needs to be coordinated, then the managers must find some means of resynchronizing everyone to the Big Picture and the various endeavors that are under way.

Because electronic discussions are continuous streams of focused to-the-point messages, being out of touch for any length of time can be disastrous. There are no meetings with minutes that can be reviewed; there probably aren't any detailed progress reports. To *really* get back in the swing of things, absentee participants need to read all the relevant discussion traffic that transpired while they were away. In active discussion forums, that can be a truly daunting prospect. If the volume is too great, the tendency is to ignore, or only skim, the earlier part of the content that was missed, and pay close attention only to the more recent posts. Depending upon what happened during the absence, this may leave the returnee floundering unexpectedly at some point in the future.

High traffic volume is a trap for the *readers*, enticing them to skim or skip. Low volume, on the other hand, is a trap for the *writers*. If there are no questions, if there is no burning need for discussion, work may proceed normally in the hands of individuals. But the very fact of its normal progress can easily lull developers into forgetting that their tasks are not only to develop, but also to communicate. Software developers are often suboptimal when it comes to documentation, so this is potentially another area that requires an effort of conscious will—to keep the other participants apprised of progress, even if it is going swimmingly.

Of course, the quite human fear of appearing foolish might also contribute to a reluctance to describe issues through online means. Mail messages are forever, so if someone makes some horribly silly mistake, it could conceivably provide amusement to others for years to come. Some people have no problem with posting whenever something comes up, whether because they've grown up in the medium, have cast-iron egos, or are so excruciatingly competent. Those who feel less comfortable, though, may have an additional hurdle to jump: the recognition and acceptance that it's OK to make a mistake, that ignorance is curable, and that keeping silent may be bad for the entire project.

Reaching Consensus. A third natural consequence of the e-mail communication medium is the difficulty of coming to conclusions or reaching consensus. Since all participants are out of realtime touch with each other, it's difficult for anyone to do the equivalent of shouting, "OK, that's enough!" If attempted in e-mail, there are almost always some people who are either in the middle of writing messages continuing the discussion, or who haven't caught up with events and are replying to older messages they're just now reading.

The need for consensus building is another illustration of the apparently paradoxical coupling of rapid ("Internet time") discussion and development with the need for more meet time to arrive at a decision. If the participants were all face-to-face in a meeting, the discussion would very likely move quite rapidly, and decisions could be made before adjournment. The online discussion medium, however, allows messages and arguments to be crafted, research to be done, and experiments performed in ways not typically possible in a meeting. Development can move quickly because all of the people involved can proceed largely at their own paces in their own time zones. Introduce some sort of *rendezvous* point for a decision to be made, however, and you have to slow the process down sufficiently to allow for the varying schedules.

As an analogy, consider several people who agree to meet somewhere before separating for lunch. The group can't proceed until the slowest member arrives at the meeting point. It gets even worse if the last person is particularly late, since there's a tendency for the earlier ones to "temporarily" pursue other activities while waiting (going to the bank, picking up a newspaper, getting a soft drink, etc.). Then even after the last person arrives, there's a need to re-coordinate before taking the next step.

FIG 1

```
Subject: cvs commit: httpd-2.0/modules/mappers mod_rewrite.c
From:  nd () apache ! org
Date:  2003-08-05 21:18:47

nd    2003/08/05 14:18:47

        Modified:  modules/mappers mod_rewrite.c
        Log:
        add a comment for future editors
        no code change.

        Revision Changes  Path
        1.222   +1 -0   httpd-2.0/modules/mappers/mod_rewrite.c

        Index: mod_rewrite.c
        ===============================================================
        RCS file: /home/cvs/httpd-2.0/modules/mappers/mod_rewrite.c,v
        retrieving revision 1.221
        retrieving revision 1.222
        diff -u -r1.221 -r1.222
        --- mod_rewrite.c  5 Aug 2003 18:45:53 -0000   1.221
        +++ mod_rewrite.c          5 Aug 2003 21:18:47 -0000   1.222
        @@ -1416,6 +1416,7 @@
                        }
                }

        +       /* buf is not zero terminated, so be careful! */
                if (i == 4 && strncasecmp(buf, "NULL", 4) == 0) {
                        return NULL;
                }
```

Summary message from Apache HTTP server project development mailing list indicating change in source code

THE SOLUTIONS

Most of these problems can be handled through the application of *netiquette*, which is essentially politeness and civility applied to the electronic communication media. Even good *netizens* can become passionate and excited, however, and lose track of how their electronic presence will be perceived.

Automation. First and foremost, remember that computers and networks are your servants. Automate as many tasks as possible and let the machines do them. Not only are they less likely to forget than a fallible developer, but their very impersonality can introduce a retardant against some forms of flame war.

For example, a large number of distributed open development projects have computers automatically send out a summary message whenever the source code is changed. For example, figure 1 displays the text of such a message from the Apache HTTP server project development mailing list: The message identifies who made the change (nd), when it was made, what file(s) were changed and why, and what the actual changes were. Lines that have been added are prefixed with a "+"; lines that have been deleted are marked with a "-". Replaced lines are shown with a "-" line marking the old text followed by a "+" line showing the new one.

The Apache environment takes it a small step further: If the change is quite large, a Web link to a page that shows the changes is provided rather than the changes themselves; this avoids taxing the network, the disks, and people's patience with megabytes of mail. Figure 2 displays an example of such a message:

Note the number of lines modified: 1,618 added, 1,543 removed. Since each of those lines is shown, plus some context on either side, the raw report would be several thousand lines long—hence, the link to a single Web-based version rather than a multitude of copies, one in each mailbox.

It may take a while to become accustomed to this sort of report, but once you have gotten used to it, you can

grasp changes quickly. And since it's a side effect of the development—requiring no additional conscious effort on the developer's part—it makes progress self-documenting after a fashion.

All of the records and archives, both of mailing lists and any sort of automated accounting such as that just described, should be [illegible] searchable. This helps identify prior art and previous discussions and decisions, thus streamlining work by providing a means of avoiding rehashing old topics [illegible] approaches.

Any other way in which automation can relieve participants of "administrivia" is probably worthy of consideration. Examples include tools for recording and collating votes or opinions; regular automated mailing of status documents or bug-tracking reports, and timed jobs that update Web pages.

Awareness and Understanding. Netiquette is a combination of common sense and the Golden Rule. Keep in mind that at the other end of your e-mail is an individual whose opinions, beliefs, culture, language, and time zone may differ from your own. Treat your correspondents not only as you would if they were in the same room with you, but also as you would want them to treat you.

Remembering to be a good netizen at all times is a difficult task, particularly since it does require conscious thought in the physical absence of others.

Tolerance and Patience. In addition to remembering that there are people on the other end of your message, you must keep in mind differences in language, culture, and/or time zone. One of the common causes of minor flame wars is simply misunderstanding: two people using the same words or expressions to mean different things, or just plain not understanding the meaning the author meant to invest in the words. I've seen this many, many times in online realtime support forums, particularly when each party is essentially unknown to the other.

Patience for others is one side of the coin; being patient with yourself is the other. For some, the challenge of speaking up online, in a forum that will be archived for an indefinite period, can seem insurmountable. One thing that can help overcome this obstacle is the comforting realization that no one was born with the knowledge you lack; everyone had to learn it at some point.

Records. One of the causes of excessive (and often unproductive) discussion, and sometimes flame wars, is a sort of "he said/she said" dispute about who said what, and when. This can be particularly disruptive if one of the participants indulges in, uh, "situational responses,"

```
Subject: cvs commit: httpd-2.0/modules/filters mod_include.Ca
From:  nd () apache ! org
Date:  2003-08-22 0:15:28
nd    2003/08/21 17:15:28
          Modified:  modules/filters mod_include.c
          Log:
          before working further, bring some kind of system into the stuff
          and (re-)order the code. That should finally improve readability...

          Revision Changes  Path
          1.239   +1618 -1543httpd-2.0/modules/filters/mod_include.c

          http://cvs.apache.org/viewcvs/httpd2.0/modules/filters/mod_include.c.diff?r1=1.238&2=1.239
```

FIG 2

Summary message from Apache HTTP server project development mailing list indicating major change in source code evading questions, changing the tune from message to message, and in general being difficult to pin down.

One of the common causes of minor flame wars is simply misunderstanding.

One way to deal with this pathological situation, as well as any number of normal confusions about how things stand, is to have the communications recorded somewhere accessible, such as an archive for the mailing list. How accessible the archives are depends entirely on your environment, but they should at least be available to the participants themselves.

Another useful record to keep is a "current status" document that details what the various current activities are, who's involved, what decisions are pending, and which decisions have been made. Most projects have something like this as a matter of course, but in a distributed community it can be immeasurably useful to have the status document sent to the mailing list or forum on a regular basis. This helps everyone keep on the same page, as well as helping avoid the possibility of "I didn't know" excuses for incomplete tasks.

Leadership. The best way to keep an electronic discussion on track is to assign someone—or more than one someone—to monitor the discussion and try to keep it flowing in productive ways. Occasionally reminding all the participants of the pitfalls waiting for the unwary is a good start, but the more challenging task is spotting incipient flame wars and heading them off before they can really get started. Sometimes they're unavoidable, so the challenge becomes one of being a champion firefighter and getting the flames dampened as quickly as possible.

Not everyone has the disposition to fill this role; sometimes leaders surface on their own, and sometimes they get drafted and surprise everyone with their skill. One of the prerequisites for an online leader is garnering respect from the other participants. As such, observation usually reveals people with the appropriate qualities—if there are any. Obviously, the larger the community is, the more likely that one or more potential leaders will be present.

Even if a discussion leader isn't in a position of authority, the moderating effect of comments and responses should be evident. Whether such a person should have any recognized authority is a highly situational matter.

In the role of peacekeeper, a community leader is well served by being facile with searching the discussion archives in order to clear up disputes before they advance very far.

CONCLUSION

If you've spent any time participating in mailing lists that have participants from all over the globe, the chances are pretty good that most of the points I've raised either resonate with you or seem painfully obvious. Unfortunately, obvious as they may appear in hindsight, in practice a lot of people overlook them.

The best tool we have for keeping our distributed discussions civil and productive is vigilance—watching for the warning signs of developing flame wars, ensuring that everyone is given a chance to participate, and, most important of all, scrutinizing our own behaviors to ensure that we're being good netizens. Q

REFERENCES

1. For more on emoticons, which were created by Scott Fahlman in 1982, visit the Web site of Wikipedia: http://wikipedia.org/wiki/Emoticon.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

KEN COAR is a director and vice president of the Apache Software Foundation and a senior software engineer with IBM. He has more than two decades of experience with network software and applications, system administration, system programming, process analysis, and computer security. Coar has worked with the World Wide Web since 1992, is a member of ACM, and is involved in the project to develop Internet RFCs for the CGI (Common Gateway Interface). He is the coauthor of *Apache Server for Dummies* (Dummies Press, 1998), coauthor of *Apache Server Unleashed* (Pearson Education, 2000), and has just completed a new book, *Apache Cookbook* (O'Reilly, 2003). He has been involved in distributed development and realtime technical support for a number of years.