

Clay Shirky

A GROUP IS ITS OWN WORST ENEMY¹

In the late 1980s, software went through a major transition.

*Before about 1985, the primary goal of software was making it possible to solve a problem, by any means necessary. Do you need to punch cards with your input data? No big deal. A typo on one card means you have to throw it away and start over? No problem. Humans will bend to the machines, like Charlie Chaplin in *Modern Times*.*

Suddenly with personal computers the bar was raised. It wasn't enough just to solve the problem: you had to solve it easily, in a way that takes into account typical human frailties. The backspace key, for example, to compensate for human frailty, not to mention menus, icons, windows, and unlimited Undo. And we called this usability, and it was good.

Lo and behold, when the software industry tried to hire experts in usability, they found that it was a new field, so nobody was doing this. There was this niche field in psychiatry called ergonomics, but it was mostly focused on things from the physical world, like finding the optimal height for a desk chair.

Eventually usability came into its own as a first-class field of study, with self-trained practitioners and university courses, and no software project could be considered complete without at least a cursory glance at usability.

1. This is a lightly edited version of the keynote Clay Shirky gave on social software at the O'Reilly Emerging Technology conference in Santa Clara on April 24, 2003. See http://www.shirky.com/writings/group_enemy.html.

We're about to undergo a similar transition.

As soon as the Internet happened, software stopped being solely about computer-to-human interaction and started being about human-to-human interaction. We had new applications like the Web, email, instant messaging, and bulletin boards, all of which were about humans communicating with one another through software.

Now, suddenly, when you create software, it isn't sufficient to think about making it possible to communicate; you have to think about making communication socially successful. In the age of usability, technical design decisions had to be taken to make software easier for a mass audience to use; in the age of social software, design decisions must be taken to make social groups survive and thrive and meet the goals of the group even when they contradict the goals of the individual. A discussion group designed by a usability expert might be optimized to make it easy to post spam about Viagra. But in social software design it's pretty obvious that the goal is to make certain things harder, not easier, and if you can make it downright impossible to post spam, you've done your job. Features need to be designed to make the group successful, not the individual.

Today, hardly anybody really studies how to design software for human-to-human interaction. The field of social software design is in its infancy. In fact, we're not even at the point yet where the software developers developing social software realize that they need to think about the sociology and the anthropology of the group that will be using their software, so many of them just throw things together and allow themselves to be surprised by the social interactions that develop around their software.

Clay Shirky has been a pioneer in this field, and his talk A Group Is Its Own Worst Enemy will be remembered as a watershed in the widespread realization that in this new era, sociology and anthropology are just as crucial to software design as usability was in the last. – Ed.



Good morning, everybody. I want to talk this morning about social software, and about a pattern I've seen over and over again in social software that supports large and long-lived groups. In particular, I want to talk about what I now think is one of the core challenges for designing large-scale social software, the pattern described in the title of this talk: "A Group Is Its Own Worst Enemy."

Let me offer a definition of social software, because it's a term that's still fairly amorphous. My definition is quite simple: it's software that supports group interaction. I also want to emphasize, though that's a fairly simple definition, how radical social software is. The Internet supports lots of communications patterns, principally point-to-point and two-way, one-to-many outbound, and many-to-many two-way.

Prior to the Internet, we had lots of patterns that supported point-to-point two-way. We had telephones; we had the telegraph. We were familiar with technological mediation of those kinds of conversations. Prior to the Internet, we had lots of patterns that supported one-way broadcast of information. I could put something on television or the radio; I could publish a newspaper. We had the printing press. So although the Internet does good things for those ways of communicating, technological support for point-to-point and broadcast well predate the Internet.

Software for groups is different. Prior to the Internet, the last technology that had any real effect on the way people sat down and talked together was the table. There was no technological mediation for group conversations. The closest we got was the conference call, which never really worked right—"Hello? Do I push this button now? Oh, shoot, I just hung up." It's not easy to set up a conference call, but it's very easy to email five of your friends and say, "Hey, where are we going for pizza?"—so ridiculously easy group forming is quite a new pattern, something technology has never made easy before.

We've had social software for 40 years at most, dated from the Plato BBS system, and we've only had a decade or so of widespread availability, so we're just finding out what works. We're still learning how to make these kinds of things.

Now, software that supports group interaction is a fundamentally unsatisfying definition in many ways, because it doesn't point to a specific class of technology. If you look at email, it obviously supports social patterns, but it can also support a broadcast pattern. If I'm a spammer, I'm going to mail things out to a million people, but they're not going to be talking to one another, and I'm not going to be talking to them—spam is email, but it isn't social. If I'm mailing you, and you're mailing me back, we're having a point-to-point conversation, but not one that creates group dynamics.

So sometimes email supports social patterns, and sometimes it doesn't. Ditto weblogs. If I'm Glenn Reynolds of Instapundit.com,² and I'm publishing something to a million users a month, with comments turned off on my blog, that's really broadcast—Glenn's users aren't talking back to him, and they aren't talking to each other. It's obviously interesting that Glenn can reach that many people as a single individual, but the pattern is closer to MSNBC than it is to a conversation. If it's a cluster of half a dozen LiveJournal³ users, on the other hand, talking about their lives with one another, that's social. So weblogs are not necessarily social, although they can support social patterns.

While that definition—software for group interaction—cuts across existing categories, I think it is the right one, because it recognizes the fundamentally social nature of the problem. Groups are a runtime effect. You cannot specify in advance what any given group will do, and so you can't instantiate in software everything you expect to have happen.

Now, there's a large body of literature saying, "We built this software, a group came and used it, and they began to exhibit behaviors that surprised us enormously, so we've gone and documented these behaviors." Over and over and over again this pattern comes up. (I hear Stewart⁴ laughing. The WELL is one of those places where this pattern came up over and over again.)

With that background out of the way, the rest of this talk is in three parts. The best explanation I have found for the kinds of things that happen when groups of humans interact is psychological research that

2. A popular political weblog – *Ed.*

3. An online journal service, similar to blog software with more of an emphasis on community – *Ed.*

4. Stewart Brand, of the WELL, a very early online community that predates the Internet, now a part of Salon.com. – *Ed.*

predates the Internet, so the first part is going to be about W. R. Bion's research, which I will talk about in a moment, research that I believe helps explain how and why a group is its own worst enemy.

The second part is: why now? What's going on now that makes this worth thinking about? I think we're seeing a revolution in social software in the current environment that's really interesting.

And third, I want to identify some things, about half a dozen things, in fact, that I think are core to any software that supports large, long-lived groups.



Part One: How Is a Group Its Own Worst Enemy?

So, Part One. The best explanation I have found for the ways in which this pattern establishes itself, the group is its own worst enemy, comes from a book by W. R. Bion called *Experiences in Groups*, written in the middle of the last century.

Bion was a psychologist who was doing group therapy with groups of neurotics. (Drawing parallels between that and the Internet is left as an exercise for the reader.) And while he was trying to treat these patients, he realized that they were, as a group, conspiring to defeat therapy.

There was no overt communication or coordination. But he could see that whenever he would try to do anything that was meant to have an effect, the group would somehow quash it. And he was driving himself crazy, in the colloquial sense of the term, trying to figure out whether or not he should be looking at the situation as “Are these individuals taking action on their own, or is this a coordinated group?”

He could never resolve the question, and so he decided that the unresolvability of the question was the answer. To the question “Do groups of people behave as aggregations of individuals or as a cohesive group?” Bion's answer was that human groups are “hopelessly committed to both,” which is to say hopelessly committed to individual identity *and* to group membership.

He said that humans are fundamentally individual, and also fundamentally social. Every one of us has a kind of rational decision-making

mind that allows us to assess what's going on and make decisions and act on them. And we are all also able to enter viscerally into emotional bonds with other groups of people who transcend the intellectual aspects of the individual.

In fact, Bion was so convinced that this was the right answer that the image he put on the front cover of his book was a Necker cube, one of those cubes that you can look at and resolve in one of two ways, but you can never see both views of the cube at the same time. So groups can be analyzed both as collections of individuals and as having this kind of emotive group experience.

Now, it's pretty easy to see how with groups of people who have formal memberships—groups that have been labeled and named like “I am a member of such-and-such a guild in a massively multiplayer online role-playing game”—you would have some kind of group cohesion there. But Bion's thesis is that this effect is much, much deeper, and kicks in much, much sooner than many of us expect. So I want to illustrate this with a story, and to illustrate the illustration, I'll use a story from your life. Because even if I don't know you, I know that what I'm about to describe has happened to you.

You are at a party, and you get bored. You say “This isn't doing it for me anymore. I'd rather be someplace else. I'd rather be home asleep. The people I wanted to talk to aren't here.” For whatever reason, the party fails to meet some threshold of interest. And then a really remarkable thing happens: you don't leave. You make a decision: “I don't like this.” If you were in a bookstore and you said, “I'm done,” you'd walk out. If you were in a coffee shop and said, “This is boring,” you'd walk out.

You're sitting at a party, and you decide, “I don't like this; I don't want to be here.” And then you don't leave. That kind of social stickiness is what Bion is talking about.

And then, another really remarkable thing happens. Twenty minutes later, one person stands up and gets their coat, and what happens? Suddenly everyone is getting their coats on, all at the same time. Which means that everyone had decided that the party was not for them, and no one had done anything about it, until finally this triggering event let the air out of the group, and everyone kind of felt okay about leaving.

This effect is so common that it's sometimes called the “paradox of groups.” It's obvious that there are no groups without members. But

what's less obvious is that there are no members without a group—because what would you be a member of?

So there's this very complicated moment of a group coming together, where enough individuals, for whatever reason, sort of agree that something worthwhile is happening, and the decision they make at that moment is "This is good and must be protected." And at that moment, even if it's subconscious, you start getting group effects. And the effects that we've seen come up over and over and over again in online communities.

Now, Bion decided that what he was watching with the neurotics was the group defending itself against his attempts to make the group do what they said they were supposed to do. This group of people was in therapy to get better, but they were during therapy defeating the very things that might help them get better. And after years of these observations, Bion said there are some very specific patterns that they're entering into in order to defeat the ostensible purpose of the group meeting together. And he detailed three patterns.

The first is sex talk, what he called, in his mid-century prose, "A group met for pairing off." And what that means is, the group conceives of its purpose as the hosting of flirtatious or salacious talk or emotions passing between pairs of members.

Imagine going on IRC (internet relay chat, a global set of chat rooms)—you scan the list of channel names, and you say, "I know what they are talking about on the #hamradio channel, because I can see the channel name." But when you go into the group, you will also almost invariably find that it's about sex talk as well, usually expressed as double entendres. The topic of sex is always in scope in live human conversations, according to Bion. (Interestingly, it is a much less frequent pattern in asynchronous communication, like mailing lists, than in synchronous ones, like IRC.) That is one basic pattern that groups can always devolve into, away from the sophisticated purpose and toward one of these basic purposes.

The second basic pattern that Bion detailed is the identification and vilification of external enemies. This is a very common pattern. Anyone who was around the open source movement in the mid-1990s could see this all the time. If you cared about Linux on the desktop, there was a big list of jobs to do. But you could always instead get a conversation

going about Microsoft and Bill Gates. And people would start bleeding from their ears, they would get so mad.

The open source movement at the time seemed pretty enemy-free, because of their mode of working: if you want to make it better, there's a list of things to do. Just fix it. But you could always get people wound up on the subject of Microsoft and Bill Gates, and the foam would start coming out of their mouths.

Nothing causes a group to galvanize like an external enemy. So even if someone isn't really your enemy, identifying them as an enemy can cause a pleasant sense of group cohesion. And groups often gravitate toward members who are the most paranoid and make them leaders, because those are the people who are best at identifying external enemies.

The third pattern Bion identified is religious veneration—the nomination and worship of a religious icon or a set of religious tenets. The religious pattern is, essentially, we have nominated something that's beyond critique. You can see this pattern on the Internet any day you like. Go onto a Tolkien newsgroup or discussion forum, and try saying, “You know, *The Two Towers* is a little dull. I mean loooong. We didn't need that much description about crossing the forest, because it's pretty much the same forest all the way.”

Try having that discussion. On the door of the group it will say, “This is for discussing the works of Tolkien.” Go in and try and have that discussion.

Now, in some places people say, “Yes, but it needed to, because it had to convey the sense of lassitude,” or whatever. But in most places you'll simply be flamed to high heaven, because you're interfering with the religious text. Groups often have some small set of core tenets, beliefs, or interests that are beyond criticism, because they are the things that hold the group together. Even in groups founded for fairly intellectual discussion, the emotional piece comes out whenever you threaten one of these core beliefs, because when you take on those beliefs, you are not just offering an opinion, you are threatening group cohesion.

Bion's patterns have shown up on the Internet, not because of the software, but because it's being used by humans. Bion has identified this possibility of groups sandbagging their sophisticated goals with these basic urges. And what he finally came to, in analyzing this tension, is that group structure is necessary. Robert's Rules of Order are necessary.

Constitutions are necessary. Norms, rituals, laws, the whole list of ways that we say, out of the universe of possible behaviors, we're going to draw a relatively small circle around the acceptable ones, all of those are ways to keep groups from just wallowing in these patterns and never actually getting anything done. Anyone who has been in a competitive industry knows you can kill a two-hour meeting by mentioning what the competition is up to, at which point everyone will stop thinking about the hard work of actually getting anything done, and will switch to alternately vilifying the competition and assuring themselves that there is no threat.

Most importantly, Bion said the various forms of group structure we have created over the centuries are necessary to defend the group from itself. Group structure exists to keep a group on target, on track, on message, on charter, to keep a group focused on its own sophisticated goals and away from sliding into these basic patterns. Group structure defends the group from the action of its own members.

This is a pattern that's shown up on the network over and over again. In the 1970s, a BBS called Communitree launched, one of the very early dial-up BBSs. This was launched when people didn't own computers—institutions owned computers.

Communitree was founded on the principles of open access and free dialogue. (“Communitree”—doesn't that say “California in the '70s”?) And the notion was, effectively, throw off structure and new and beautiful new social patterns will arise.

And, indeed, as anyone who has put discussion software into groups that were previously disconnected has seen, that does happen. Incredible things happen. The early days of Echo,⁵ the early days of Usenet, the early days of Lucasfilms' Habitat (one of the original multiplayer games), over and over again, you see all this incredible upwelling of people who suddenly are connected in ways they weren't before.

But it's not all beautiful; as time sets in, difficulties emerge. In this case, one of the difficulties was occasioned by the fact that one of the institutions that joined Communitree was a high school. And who, in 1978, was hanging out in the room with the computer and the modems in it but the boys of that high school. And the boys weren't terribly

5. A small but prestigious online community in New York City, which also predates the Internet, created by Stacy Horn at NYU. — *Ed.*

interested in sophisticated adult conversation. They were interested in fart jokes. They were interested in salacious talk. They were interested in running amok and posting four-letter words and nyah-nyah-nyah all over the bulletin board.

And the adults who had set up Communitree were horrified, because they were being overrun by these students. The place that was founded on open access had too much open access, too much openness. They couldn't defend themselves against their own users. The place that was founded on free speech had too much freedom. They had no way of saying, "No, that's not the kind of free speech we meant."

But that was a requirement. In order to defend themselves against being overrun, that was something that they needed to have that they didn't have, and in the end, they simply shut the site down.

Now you could ask whether or not the founders' inability to defend themselves from this onslaught, from being overrun, was a technical or a social problem. Did the software not allow the problem to be solved? Or was it the social configuration of the group that founded it, where they simply couldn't stomach the idea of adding censorship to protect their system. But in a way, it doesn't matter, because technical and social issues are deeply intertwined. There's no way to completely separate them.

What matters is, a group designed this and then was unable, in the context they'd set up, to save it from this attack from within, and that context was partly technical and partly social. The lesson of Communitree is that attack from within is what matters. Communitree wasn't shut down by people trying to crash the server or flood it from the outside. It was shut down by people logging in and posting, which is what the system was designed to allow. The technological patterns of normal use and attack were so similar at the machine level, there was no way to specify technologically what should and shouldn't happen. Some of the users wanted the system to continue to exist and to provide a forum for discussion. And other of the users, the high school boys, either didn't care or were actively inimical. And the system provided no way for the former group to defend itself from the latter.

This pattern has happened over and over and over again. Someone built the system; they assumed certain user behaviors. The users came on and exhibited different behaviors. And the people running the system discovered to their horror that the technological and social issues could

not in fact be decoupled. This story has been written many times. It's actually frustrating to see how many times it's been written, because although there's a wealth of documentation from the field, people starting similar projects often haven't read these accounts.

The most charitable description of this repeated pattern is "learning from experience," but learning from experience is the worst possible way to learn something. Learning from experience is one up from remembering—that's not great. The best way to learn something is when someone else figures it out and tells you: "Don't go in that swamp. There are alligators in there."

Learning from experience about the alligators is lousy, compared to learning from reading, say. There hasn't been, unfortunately, in this arena, a lot of learning from reading. And so, the essay "Lessons from Lucasfilms' Habitat,"⁶ written in 1990, reads a lot like Rose Stone's description of Communitree from 1978.

There's a great document called "LambdaMOO Takes a New Direction," which is about the wizards of LambdaMOO, Pavel Curtis's Xerox PARC experiment in building a MUD⁷ world. And one day the wizards of LambdaMOO announced, "We've gotten this system up and running, and all these interesting social effects are happening. Henceforth we wizards will only be involved in technological issues. We're not going to get involved in any of that social stuff."

And then, I think about 18 months later, the wizards come back, extremely cranky. And they say, "What we have learned from you whining users is that we can't do what we said we would do. We cannot separate the technological aspects from the social aspects of running a virtual world.

"So we're back, and we're taking wizardly fiat back, and we're going to do things to run the system. We are effectively setting ourselves up as a government, because this place needs a government, because without us, everything was falling apart."

People who work on social software are closer in spirit to economists and political scientists than they are to people making compilers. They both look like programming, but when you're dealing with groups of people as one of your runtime phenomena, you have an incredibly

6. See <http://www.fudco.com/chip/lessons.html>.

7. Multiuser Dungeon, a textual online multiplayer adventure game – *Ed*.

different practice. In the political realm, we would call these kinds of crises a constitutional crisis. It's what happens when the tension between the individual and the group, and the rights and responsibilities of individuals and groups, gets so serious that something has to be done.

And the worst crisis is the first crisis, because it's not just "We need to have some rules." It's also "We need to have some rules for making some rules." And this is what we see over and over again in large and long-lived social software systems. Constitutions are a necessary component of large, long-lived, heterogeneous groups.

Geoff Cohen has a great observation about this. He said, "The likelihood that any unmoderated group will eventually get into a flame-war about whether or not to have a moderator approaches one as time increases." As a group commits to its existence as a group, and begins to think that the group is good or important, the chance that they will begin to call for additional structure, in order to defend themselves from themselves, gets very, very high.



Part Two: Why Now?

If these things I'm saying have happened so often before, have been happening and been documented and we've got psychological literature that predates the Internet, what's going on now that makes this important?

I can't tell you precisely why, but observationally there is a revolution in social software going on. The number of people writing tools to support or enhance group collaboration or communication is astonishing.

The Web turned us all into size queens for six or eight years there. It was loosely coupled, it was stateless, it scaled like crazy, and everything became about how big you could get. "How many users does Yahoo have? How many customers does Amazon have? How many readers does MSNBC have?" And the answer could be "A lot!" But MSNBC, say, could only get a lot if they didn't have to be talking with their users, just talking to them, and they didn't have to figure out a way to let those readers talk to each other.

The downside of going for size and scale above all else is that the dense, interconnected pattern that drives group conversation and collaboration isn't supportable at any large scale. Less is different—small groups of people can engage in kinds of interaction that large groups can't, and during the Web years, we blew past that interesting scale of small groups. In groups of larger than a dozen but smaller than a few hundred, there are conversational forms that can't be supported when you're talking about thousands or millions of users a single group.

We've had things like mailing lists and BBSs for a long time. More recently we've had IM, and we've had these various tools for a while. But now, all of a sudden, a bunch of new forms are spreading. We've gotten weblogs and wikis, and I think, even more importantly, we're getting platform stuff. We're getting RSS. We're getting shared Flash objects. We're getting ways to quickly build on top of some infrastructure we can take for granted, that lets us try new things very rapidly.

I was talking to Stewart Butterfield about Flickr, the application they're launching here. I said, "Hey, how's that going?" He said, "Well, we only had the idea for it two weeks ago. So this is the launch." When you can go from "Hey, I've got an idea" to "Let's launch this in front of a few hundred serious geeks and see how it works," that suggests that there's a platform there that is letting people do some really interesting things really quickly. It's not that you couldn't have built a similar application a couple of years ago, but the cost would have been much higher. And when you lower costs, interesting new kinds of things happen.

So the first answer to Why Now? is simply, "Because it's time." I can't tell you why it took as long for weblogs to happen as it did, except to say it had absolutely nothing to do with technology. We had every bit of technology we needed to do weblogs in 1994, the day Mosaic launched the first forms-capable browser. Every single piece of it was right there. Instead, we got Geocities. Why did we get Geocities and not weblogs? We didn't know what we were doing.

One was a bad idea; the other turns out to be a really good idea. It took a long time to figure out that people talking to one another, instead of simply uploading badly scanned photos of their cats, would be the real source of value.

We got the weblog pattern in around '96 with Drudge. We got weblog platforms starting in '98. The thing really was taking off in

2000. By last year, it was “Omigod, this thing is going mainstream, and it’s going to change everything.”

The vertigo moment for me was when Phil Gyford launched the Pepys weblog, Samuel Pepys’ diaries of the 1660s turned into a weblog form, with a new post every day from Pepys’ diary. What that said to me was that Phil was asserting, and I now believe, that weblogs will be around for at least 10 years, because that’s how long Pepys kept a diary. And that was this moment of projecting into the future: this is now infrastructure we can take for granted.

Why was there an eight-year gap between a forms-capable browser and the Pepys diaries? It just takes a while for people to get used to these ideas, to understand the technical form well enough to put it to socially novel uses.

The other big change is that the social software people are building now is web-native, built on the Web from the ground up. When you got social software on the Web in the mid-1990s, a lot of it was enterprise software with a web front-end slapped on: “This is the Giant Lotus Dreadnought, now with New Lightweight Web Interface!” It never felt like the Web. It felt like this hulking thing tarted up with some clickable icons.

A weblog is web-native. It’s the Web all the way in. A wiki is a web-native way of hosting collaboration. It’s lightweight, it’s loosely coupled, it’s easy to extend, it’s easy to break down. And it’s not just the surface, like “Oh, you can just do things in a form.” It assumes HTTP is transport. It assumes markup in the coding. RSS is a web-native way of doing syndication. So we’re taking all of these tools and we’re extending them in a way that lets us build new things really quickly.

The third thing that’s happening now to accelerate social software is that, in David Weinberger’s felicitous phrase, we have a “Small Pieces Loosely Joined” way of making software. It’s really worth looking into what Joi Ito is doing with the Emergent Democracy movement, even if you’re not interested in the themes of emerging democracy. This started because a conversation was going on, and Ito said, “I am frustrated. I’m sitting here in Japan, and I know all of these people are having these conversations in real time with one another. I want to have a group conversation, too. I’ll start a conference call.

“But since conference calls are so lousy on their own, I’m going to bring up a chat window at the same time.” And then, in the first

meeting, I think it was Pete Kaminski who said, “Well, I’ve also opened up a wiki, and here’s the URL.” And he posts the URL of the wiki in the chat window. And people on the call also start annotating things in the wiki, adding bookmarks in the chat channel, and so on. The meeting is going on in three separate modes at the same time, two in real time (the phone and the chat) and one annotated (the wiki).

You know how conference calls usually are: either one or two people dominate it, or everyone’s walking over each other, interrupting and cutting each other off. It’s very difficult to coordinate speakers in a conference call because people can’t see one another, which makes it hard to manage the interrupt logic. In Joi’s conference call, the interrupt logic got moved to the chat room. People would type “Hand,” and the moderator of the conference call will then type, “You’re speaking next,” in the chat. So the conference call flowed incredibly smoothly, because the chat provided a kind of control channel for the speaking.

Meanwhile, in the chat, people are annotating what people are saying. “Oh, that reminds me of So-and-so’s work.” Or “You should look at this URL. . . you should look at that ISBN number.” In a conference call, to read out a URL, you have to spell it out—“No, no, no, it’s w w w dot net dash. . .” In a chat window, you get it and you can click on it right there. You can say, in the conference call or the chat: “Go over to the wiki and look at this.”

This is a broadband, multimedia conference call, but it isn’t implemented as a single giant thing. It’s just three little pieces of software, laid next to each other and held together with a little bit of social glue. This is an incredibly powerful pattern. It’s different from “Let’s take the Lotus juggernaut and add a web front-end.”

And the fourth and final driving the current revolution in social software is ubiquity. The Web has been growing for a long, long time. In the beginning, just a few people had web access, and then lots of people had web access, and then most people had web access. But something different is happening now. In many situations, *all* people have access to the network. And “all” is a different kind of amount than “most.” “All” lets you start taking things for granted.

Now, the Internet isn’t everywhere in the world. It isn’t even everywhere in the developed world. But for some groups of people—students, people in high-tech offices, knowledge workers—everyone they work with is online. Everyone they’re friends with is online. Everyone in their family is online.

This pattern of ubiquity lets you start taking this for granted. Bill Joy once said, “My method is to look at something that seems like a good idea and assume it’s true.” We’re starting to see software that simply assumes that all offline groups will have an online component, no matter what. It is now possible for every grouping, from a Girl Scout troop on up, to have an online component, and for it to be lightweight and easy to manage. And that’s a different kind of thing than the old pattern of “online community.” I have this Venn diagram image of two hula hoops, where my real life is off to the left, and my online life is off to the right, and I’m the only thing in common between the two; people in my offline world are different than people in my online world. And for most of the last 30 years, the Net has been like that—you had different friends online than offline. If the hula hoops are swung together, though, so that everyone who’s offline is also online, that’s a different kind of pattern. In a world of ubiquitous Net access, the split between offline and online is not between different groups, but between different modes of interacting in one group.

There’s a second kind of ubiquity, which is the kind we’re enjoying here at the conference, thanks to the Wifi network at the conference. If you assume whenever a group of people are gathered together that they can be both face to face and online at the same time, you can start to do different kinds of things than if real versus virtual communications are treated as separate cases. I don’t run a real-world meeting now without either having a chat room or a wiki up and running. Three weeks ago I ran a meeting for the Library of Congress. We had a wiki, set up by Socialtext, and used it during the meeting to capture a large and very dense amount of technical information on long-term digital preservation.

It really quickly becomes an assumption that a group can do things like “Oh, I took my PowerPoint slides, I showed them, and then I dumped them into the wiki. So now you can get at them.” It becomes a sort of shared repository for group memory. This is new. These kinds of ubiquity, both “everyone is online,” and “everyone who’s in a room can be online together at the same time,” are leading to new patterns.



Part Three: What Can We Take for Granted?

If these assumptions are right—first that a group is its own worst enemy, and second, we’re seeing this explosion of social software—what should we do? Can we say anything with any certainty about building social software, at least for large and long-lived groups?

I think we can. A little over 10 years ago, I quit my day job, because Usenet was so interesting. I thought at the time, “This is really going to be big.” And I actually wrote a book called *Voices from the Net*, about Net culture at the time, Usenet, the Well, Echo, IRC, and so forth. It was published in April of ’95, just as that world was being washed away by the Web. But it was my original interest, so I’ve been looking at this problem in one way or another for 10 years, and I’ve been looking at it pretty hard for a year and a half or so.

So there’s this question: “What is required to make a large, long-lived online group successful?” I think I can now answer with some confidence: “It depends.” (I’m hoping to flesh that answer out a little bit in the next 10 years.)

But I can at least say some of the things it depends on. The Calvinists had a doctrine of natural grace and supernatural grace. Natural grace was, “You have to do all the right things in the world to get to heaven. . .” and supernatural grace was, “. . .and God has to anoint you.” And you never knew if you had supernatural grace or not. This was their way of getting around the fact that the book of Revelation put an upper limit on the number of people who were going to heaven.

Social software is like that. You can find the same piece of code running in many, many environments. And sometimes it works and sometimes it doesn’t. So there is something supernatural about groups, where having good software alone isn’t enough, because the social behavior of groups is a runtime experience.

The normal experience of social software is failure. If you go into Yahoo groups and you map out the subscriptions, it is, unsurprisingly, a power law. There's a small number of highly populated groups, a moderate number of moderately populated groups, and this long, flat tail of failure. And the failure is inevitably more than 50% of the total mailing lists in any category. So it's not like a cake recipe. There's nothing you can do to make it come out right every time.

There are, however, I think, about half a dozen things that are broadly true of all the groups I've looked at and all the online constitutions I've read for software that supports large and long-lived groups. And I'd break that list in half. I'd say, if you are going to create a piece of social software designed to support large groups, you have to accept three things, and design for four things.

Three Things to Accept

1.) Of the things you have to accept, the first is that you cannot completely separate technical and social issues. There are two attractive patterns for thinking about the intersection of social and technological issues. One says, "We'll handle technology over here, we'll do social issues there. We'll have separate mailing lists with separate discussion groups, or we'll have one track here and one track there." This doesn't work; you can't separate the two. It's never been stated more clearly than in the pair of documents called "LambdaMOO Takes a New Direction" that I referred to earlier. I can do no better than to point you to those documents.

This may seem obvious, but it's one of those patterns that gets endlessly repeated. I recently was on a social software discussion list, and someone said, "Hey everybody, I know! Let's set up a second mailing list for just discussing the technical issues." The LambdaMOO docs were written in the early '90s, here it is 2003, and people still want to believe that the technology has some kind of clean edge that separates from the behavior of the mere users. And of course what happened when the second technical mailing list was created? Nothing. Nothing happened. No one moved the conversations away from the first list; no one could fork the conversation between social and technical issues, because the conversation can't be forked.

There's another way of thinking about tech and social dynamics that's very, very attractive—anybody who looks at this stuff has the same epiphany: “Omigod, the software is determining what people do!” And that is true, up to a point. But you cannot completely program social issues either—different mailing lists run on the same software but have different cultures; both Slashdot and Plastic.com run on the same software platform, but they have very different cultures too.

You can't separate technological effects from social ones, and you can't specify all social issues in technology. The group is going to assert its existence independently of the software somehow, and you're going to get a mix of social and technological effects.

The group is real. It will exhibit emergent effects. It can't be ignored, and it can't be programmed, which means you have an ongoing issue. And the best pattern, or at least the pattern that's worked the most often, is to put into the hands of the group itself the responsibility for defining what value is, and defending that value, rather than trying to describe everything in the software up front.

2.) The second thing you have to accept: members are different from users. A pattern will arise in which there is some group of users that cares more than average about the integrity and success of the group as a whole. And that becomes your core group, Art Kleiner's phrase for “the group within the group that matters most.”

The core group on Communitree was undifferentiated from the group of random users that came in. They were separate in their own minds, because they knew what they wanted to do, but they couldn't defend themselves against the other users. But in all successful online communities that I've looked at, a core group arises that cares about the community as a whole—not just their part of it—and that gardens effectively and takes care of the social environment by encouraging good behavior and discouraging bad behavior.

Now, if the software does not always allow the core group to express itself, it will invent new ways of doing so. On `alt.folklore.urban`, the Usenet discussion group about urban folklore, a group of people hung out together and, over time, got to be friends. Enough of these AFU regulars were also Silicon Valley dwellers that they decided to get together for a real-world barbecue, and to coordinate that, they set up a separate mailing list, which they called the Old Hats list.

After the barbecue, though, the mailing list stayed up, and membership was extended to other AFU readers, but only selectively, only to those members who'd been around AFU long enough to get to know everyone—the average reader of AFU didn't even know the mailing list existed. Old Hats became a place for meta-discussion, discussion about AFU, and the members of Old Hats began to coordinate efforts formally if they were going to troll someone or flame someone or ignore someone in `alt.folklore.urban` itself.

Then, as Usenet kept growing, many newcomers arrived and seemed to like the environment, because it was well run. In order to defend themselves from the scaling issues that come from adding a lot of new members to the Old Hats list, they said, “We're starting a second list, called the Young Hats.”

So AFU ended up with this three-tier system, not dissimilar to the tiers of anonymous cowards, logged-in users, and people with high karma on Slashdot. But because Usenet didn't let the AFU core group do it in the software, they brought in other pieces of software, these mailing lists, that they needed to build the structure. So you don't get the program users—in any healthy group, the members in good standing will find one another and be recognized by one another.

3.) The third thing you need to accept: the core group has rights that trump individual rights in some situations. This pulls against the libertarian view that's quite common on the network, and it absolutely pulls against the one-person/one-vote notion. But you can see examples of how bad an idea voting is when citizenship is the same as ability to log in.

In the early '90s, a proposal went out to create a Usenet newsgroup for discussing Tibetan culture, to be called `soc.culture.tibet`. And it was voted down, in large part because a number of Chinese students who had Internet access voted it down, on the logic that Tibet wasn't a country; it was a region of China. And in their view, since Tibet wasn't a country, there oughtn't be any place to discuss its culture, because that was oxymoronic.

Now, everyone could see that this was the wrong answer. The people who wanted a place to discuss Tibetan culture should have it. That was the core group. But because the one-person/one-vote model on Usenet said, “Anyone who's on Usenet gets to vote on any group,” sufficiently contentious groups could simply be voted away.

Imagine today if, in the United States, Internet users had to be polled before any discussion group opposed to the war in Iraq could be created, or French users had to be polled before any pro-war group could be created. The people who want to have those discussions are the people who matter, and absolute citizenship, with the idea that if you can log in, you are a citizen, can actually be a harmful pattern, because it allows the tyranny of the majority. The core group needs ways to defend itself so that it can keep the larger group concentrated on its sophisticated goals and away from its basic instincts.

The Wikipedia (the group-created online encyclopedia) has a similar system today, with a “volunteer fire department,” a group of people who care to an unusual degree about the success of the Wikipedia. And since they have enough leverage (because of the way wikis work, they can always roll back graffiti and so forth), that thing has stayed up despite repeated attacks. So leveraging the core group is a really powerful system.

And because of the difficulty in maintaining a focus on sophisticated goals, all groups of any integrity have a constitution. There is always an informal piece of the Constitution, and there is sometimes a formal piece as well, an explicit and publicly examinable piece. At the very least, the formal part is what’s instantiated in code—“the software works this way.” The informal part is the sense of “how we do it around here.” And no matter how it is substantiated in code or written in charter, whatever, there will always be an informal part as well. You can’t separate the two.

Now, when I say these are three things you have to accept, I mean you *have* to accept them, because if you don’t accept them up front, they’ll happen to you anyway. And then you’ll end up writing one of those documents that says, “Oh, we launched this and we tried it, and then the users came along and did all these weird things. And now we’re documenting it so future ages won’t make this mistake”—even though you didn’t read the thing that was written in 1978.

Four Things to Design For

In addition to the things you have to accept, the forced moves, I also believe there are a handful of things designers of group software need to design for:

1.) The first thing you would design for is handles the user can invest in. Now, I say “handles” because I don’t want to say “identity”; identity has recently become one of those ideas where, when you pull on the little thread you want, this big bag of stuff comes along with it. Identity is such a hot-button issue now, but for the lightweight stuff required for social software, it’s really just a handle that matters.

It’s pretty widely understood that anonymity doesn’t work well in group settings, because “who said what when” is the minimum requirement for having a conversation. What’s less well understood is that weak pseudonymity doesn’t work well, either, because I need to associate who’s saying something to me now with previous conversations.

The world’s best reputation management system is right here, in the brain. And actually, it’s right here, in the back, in the emotional part of the brain. Almost all the work being done on reputation systems today is either trivial or useless or both, because in most human situations, reputations aren’t easy to make explicit. eBay has done us all an enormous disservice, because eBay works in noniterated atomic transactions, which are the opposite of social situations. eBay’s reputation system works incredibly well, because it starts with a simple transaction (“How much money for how many Smurfs?”) and turns that into a metric that’s equally linear. That doesn’t work well in social situations, where karma, a.k.a. nonreciprocal altruism, is a much subtler and more diffuse thing than eBay’s reputation is.

Reputation is also not generalizable or portable. There are people who will cheat on their spouse but not at cards, and vice versa, and both, and neither. Reputation in one situation is not necessarily directly portable to another.

If you want a good reputation system, just let me remember who you are. And if you do me a favor, I’ll remember it. And I won’t store it in the front of my brain; I’ll store it here, in the back. I’ll just get a good feeling next time I get email from you; I won’t even remember why. And if you do me a disservice and I get email from you, my temples will start to throb, and I won’t even remember why. If you give users a way of remembering one another, reputation will happen, and that requires nothing more than simple and somewhat persistent handles.

Users have to be able to identify themselves and there has to be a penalty for switching handles. The penalty for switching doesn’t have to be total. But if I change my handle on the system, I have to lose some

kind of reputation or some kind of context. This keeps the system functioning.

Of course, this pulls against the sense that we've had since the early psychological writings about the Internet. "Oh, on the Internet we're all going to be changing identities and genders like we change our socks." But this sense of completely fluid identity is disrupted by things like the Kaycee Nicole story.

The story is baroque, but the basic outline is simple: a woman in Kansas was living online in an alternate persona, a high school student named Kaycee Nicole, and then, because the invented high school student's friends got so emotionally involved, the woman decided to kill her persona off, and so she began reporting, in the persona of Kaycee Nicole, that she had contracted a fatal disease.

So here's this attractive young woman everyone has befriended and now she's dying, and what happens? Everyone wants to fly to meet her before she goes. So then woman sort of panicked and Kaycee Nicole vanished. And a bunch of places on the Internet, particularly the MetaFilter community, started to smell a rat. And dozens of those people spent *hundreds* of hours trying to find out what was going on—it was sort of a distributed detective movement—and they eventually uncovered the hoax by putting all the various pieces together from Nicole's various posts.

Now a number of people point to this and say, "See, I told you about how fluid identity is online!" But that's not the lesson of the Kaycee Nicole story; the important lesson is this: changing your identity is really weird. And when the community understands that you've been doing it and you're faking, that is seen as a huge and violent transgression. And they will expend an astonishing amount of energy to find you and punish you. So identity is much less slippery than the early literature would lead us to believe, because although the technology makes fluid identity easy, social life demands some degree of fixity. And all you need is a system with some sort of persistent handle, and users will invest them with all the trappings of identity and even the layers above that like reputation.

2.) Second, you have to design a way for there to be members in good standing, some way in which good works get recognized. The minimal way is, posts appear with identity. You can do more sophisticated things like having formal karma or listing "member since" dates or noting who is a Pro user who helps fund the system.

I'm on the fence about whether or not this is a design worth accepting, because in a way I think members in good standing will rise. But more and more of the systems I'm seeing launching these days are having some kind of additional accretion so you can tell how much involvement members have with the system.

There's an interesting pattern I'm seeing among the music-sharing group that operates between Tokyo and Hong Kong. They operate on a mailing list, which they set up for themselves. But when they're trading music, what they're doing is, they're FedExing one another 180-gig hard-drives. So they're sending each other .wav files and not MP3s, and they're sending them in bulk.

Now, you can imagine that such a system might be a target for organizations that would frown on this activity. So when you join that group, your username is appended with the username of the person who is your sponsor. You can't get in without your name being linked to someone else. You can see immediately the reputational effects going on there, just from linking two handles.

So in that system, you become a member in good standing when your sponsor link goes away and you're there on your own report. If, on the other hand, you defect, not only are you booted, but your sponsor is booted. There are lots and lots of lightweight ways to accept and work with the idea of member in good standing.

3.) Three, you need some barriers to participation, however small. This is one of the things that killed Usenet, because there was almost no barrier to posting, leading to both generic system failures like spam, and also specific failures, like constant misogynist attacks in any group related to feminism, or racist attacks in any group related to African-Americans. You have to have some cost to either join or participate, if not at the lowest level, then at higher levels. There needs to be some kind of segmentation of capabilities.

Now, the segmentation can be total—you're in or you're out, as with the music group I just listed. Or it can be partial—anyone can read Slashdot, anonymous cowards can post, non-anonymous cowards can post with a higher rating. But to moderate, you really have to have been around for a while. It has to be hard to do at least some things on the system for some users, or the core group will not have the tools that they need to defend themselves.

Now, this pulls against the cardinal programming virtue of ease of use, but ease of use is the wrong goal for social software. Ease of use is the wrong way to look at the situation, because you've got the Necker cube flipped in the wrong direction, toward the individual, when in fact, the user of a piece of social software is the group.

The groups' goals sometimes differ from those of the individual members, and the user of social software is the group, so ease of use should be for the group. If the ease of use is only calculated from the user's point of view, it will be difficult to defend the group from the "group is its own worst enemy" style attacks from within.

I think we've all been to meetings where everyone had a really good time, everyone was telling jokes and laughing, and it was a great meeting, except we got nothing done. Everyone was amusing themselves so much that the group's goal was defeated by the individual interventions.

4.) Finally, you have to find a way to spare the group from scale. Scale alone kills conversations, because conversations require dense two-way conversations. In conversational contexts, Metcalfe's Law—the number of connections grows with the square of the number of nodes—is a drag. Since the number of potential two-way conversations in a group grows so much faster than the size of the group itself, the density of conversation falls off very fast as the system scales up even a little bit. You have to have some way to let users hang onto the "less is more" pattern, in order to keep associated with one another.

This is an "inverse value to scale" issue. Think about your Rolodex: a thousand contacts, maybe 150 people you can call friends, 30 people you can call close friends, two or three people you'd donate a kidney to. The value is inverse to the size of the group. And you have to find some way to protect the group within the context of those effects.

Sometimes you can do soft forking. LiveJournal does the best soft forking of any software I've ever seen, where the concepts of "you" and "your group" are pretty much intertwined. The average size of a LiveJournal group is about a dozen people. And the median size is around five.

But each user is a little bit connected to other such clusters, through their friends, and so while the clusters are real, they're not completely bounded—there's a soft overlap, which means that although most users participate in small groups, most of the half-million LiveJournal users are connected to one another through some short chain.

Some pieces of social software, like IRC channels and mailing lists, are self-moderating with scale, because as the signal-to-noise ratio gets worse, people start to drop off, until it gets better, so people join, and so it gets worse. You get these sorts of oscillating patterns, but the overall system is self-correcting.

And then my favorite pattern is from MetaFilter, which is: when we start seeing effects of scale, we shut off the new user page. “Someone mentions us in the press and how great we are? ’Bye!” That’s a way of raising the bar; that’s creating a threshold of participation. And anyone who bookmarks that page and says, “You know, I really want to be in there; maybe I’ll go back later,” that’s the kind of user MeFi wants to have.

You have to find some way to protect your own users from scale. This doesn’t mean the scale of the whole system can’t grow. But you can’t try to make the system large by taking individual conversations and blowing them up like a balloon; human interaction, many-to-many interaction, doesn’t blow up like a balloon. It either dissipates, or turns into broadcast, or collapses. So plan for dealing with scale in advance, because it’s going to happen anyway.



Conclusion

Now, those four things are of course necessary but not sufficient conditions. I propose them more as a platform for building the interesting differences off. There are lots and lots and lots of other effects that make different bits of software interesting enough that you would want to keep more than one kind of pattern around. But those are commonalities I’m seeing across a range of social software for large and long-lived groups.

In addition, you can do all sorts of things with explicit clustering, whether it’s guilds in massively multiplayer games, or communities on LiveJournal or what have you. You can do things with conversational artifacts, where the group participation leaves behind some record. Right now, the Wikipedia is the most interesting conversational artifact

I know of, where product is a result of process. Rather than “We’re specifically going to get together and create this presentation,” it’s just “What’s left is a record of what we said.”

There are all these things, and of course they differ platform to platform. But there is, I believe, this common core of things that will happen whether you plan for them or not, and things you should plan for, that I think are invariant across large communal software.

Writing social software is hard. And, as I said, the act of writing social software is more like the work of an economist or a political scientist. And the act of hosting social software, the relationship of someone who hosts it is more like a relationship of landlords to tenants than owners to boxes in a warehouse.

The people using your software, even if you own it and pay for it, have rights and will behave as if they have rights. And if you abrogate those rights, you’ll hear about it very quickly.

That’s part of the problem that the John Hegel theory of community—“community leads to content, which leads to commerce”—never worked. Because lo and behold, no matter who came onto the Clairol chat boards, they sometimes wanted to talk about things that weren’t Clairol products.

“But we paid for this! This is the Clairol site!” say the sponsors. Doesn’t matter. The users are there for one another. They may be there on hardware and software paid for by you, but the users are there for one another.

The patterns here, I am suggesting, both the things to accept and the things to design for, are givens. Assume that addressing these issues is a forced move in the social platform, and then you can start going out and building on top of that the interesting stuff that I think is going to be the real result of this period of experimentation with social software.

Thank you very much.