THE SIGNIFICANCE OF SOCIAL SOFTWARE

danah boyd

UC Berkeley School of Information, Berkeley, CA, USA
dmb@sims.berkeley.edu
http://www.zephoria.org/

Social software… It's a term that is thrown around frequently, rarely defined, completely elusive, and yet totally significant. What on earth does it mean? And should you care?
(IMHO) Of course you should care! Social software is all around us now. It is altering the organization of social life. How do you communicate with loved ones? How do you find information? A lot of this hinges on social software.
In this opening essay, I want to address the significance of „social software." I will look at this from a few different angles, building on the attitudes and practices of creators, users, and researchers of social software. My goal is to lay out some possible approaches for thinking about the types of technologies and behaviors that emerge. What I'm offering is not some truth serum (nor even a proper analysis) – it can and should be contested, questioned, and wrestled with.

Let's begin.

Social Software

First… what constitutes „social software"? Where does this term come from?
In 2002, Clay Shirky (re)claimed the term „social software" to encompass „all uses of software that supported interacting groups, even if the interaction was offline."[1] His choice was intentional. Although other terms existed – groupware, social computing, computer-mediated communication, etc. – he felt that these older terms

were either polluted or a bad fit to address certain new technologies. He introduced the term in conjunction with an event – the „Social Software Summit" – that was organized to gather like minds to talk about this kind of technology.
Of course, the event was invite-only and the participants were primarily invested in the development of *new* genres of social technologies. While research and development in social technology extends back decades, none of the old guard was present. Not surprisingly, while Shirky wanted the term to be all encompassing, it took on the values and scope of the early community that circled his event, referring only to the kinds of technologies emerging from the Summit attendees, their friends, and those that they identify as part of their tech community.
Still, the term proliferated. Many who participated were prominent bloggers or organizers of other large tech industry events where the term starting floating in the air as the solution to the tech crash. The O'Reilly Emerging Technologies Conference (Etcon/Etech) devoted an entire track to „social software." Because this conference is all about the *new*, the conference reinforced the idea that social software is all about the *new*.
While social software became a moniker to denote contemporary technology that supports social interaction, the community continued to define the term broadly. Shirky often uses the succinct „stuff worth spamming"[2] while Tom Coates notes that „Social Software can be loosely defined as software which supports, extends, or derives added value from, human social behaviour - message-boards, musical taste-sharing, photo-sharing, instant messaging, mailing lists, social networking."[3]
Of course, not everyone was so happy to see the term appear, let alone take hold. Researchers and developers of earlier technologies that supported communication and collaboration took offense at the industry's apparent myopia. While Shirky certainly understood the historical foundation on which social technologies were being built, venture capitalists and entrepreneurs were running around screaming *new new new*, generating as much hype as possible.

Critics saw the hype and challenged bloggers who used the term, arguing that social software is simply a hyped term used by the blogosphere in order to make a phenomenon out of something that always was; there are no technological advances in social software – it's just another term that encompasses „groupware,“ „computer-mediated communication,“ „social computing“ and „sociable media.“ Embedded in this complaint is an argument that social software is simply a political move to separate the nouveau technologists from the old skool developers, practitioners, and researchers. This outrage stems particularly from a perceived distance between the overarching portrait that Shirky's definition seems to portray and the limited scope in which the term is typically employed. Because of this contradiction, critics gaffed at those involved in social software and dismissed social software itself as ahistorical.

I want to acknowledge this bickering and now throw it away. Regardless of what Shirky, Coates, and others might have intended with the term... Regardless of how pissed off academics and earlier practitioners are... Regardless of what you think the term should mean... The fact is that social software has come to reference a particular set of technologies developed in the post-web-bust era. In other words, in practice, „social software" is about a movement, not simply a category of technologies. It's about recognizing that the era of e-commerce centered business models is over; we've moved on to web software that is all about letting people interact with people and data in a fluid way. It's about recognizing that the web can be more than a broadcast channel; collections of user-generated content can have value. No matter what, it is indeed about the *new* but the new has nothing to do with technology; it has to do with attitude.

So let's roll with this framing of „social software“ as the social technologies that fit under the venture capital-backed nuveau tech boom dreadfully called „Web 2.0." It's certainly not complete and as a category, it's difficult to make sense of its boundaries. It's unclear who's in and who's out when it comes to community membership. And it is certainly not unproblematic. But y'know what? It's good enough for talking about the wave of new technologies that have emerged post-bust as well as the behaviors that have captured the imagination of millions of users worldwide. So, regardless of how high your blood pressure is right now, let’s move forward.

What’s it matter?

Of course, if we accept social software as a shift, how is it different? And why does it matter? If we acknowledge that the technologies aren’t that fundamentally different from technologies with other (dismissed) labels, why should we focus solely on the newfangled things?

I would like to argue that „social software“ has brought about three dramatic changes - one in the way that technologies are designed, one in the way that participation spreads, and one in the way that people behave. Let's start with the first one.

How Technologies are designed

Given that technology development involves software *engineering*, it shouldn’t be all that surprising that, historically, an engineering mindset shaped the development of technology. Systems were designed, tested, and then deployed. This certainly made sense when software was shipped on disks to stores to be purchased, but the same patterns applied to early web development as well. Sure, bug fixes were shipped and new versions came out more quickly online, but the terms „beta“ and „version 2.1“ actually meant something in the early days.

Friendster killed the beta. Even two years after launch, with millions of users, Friendster still labeled itself beta. Friendster’s beta status was a source of immense laughter at geek events. By that point, Friendster had gone through so many iterations that people were speculating that Friendster would never leave beta. (Of course, most of their „developments” involved removing features instead of adding them.) Given this, frustrated onlookers questioned what beta could possibly mean.

The only answer that the snarky amongst us could come up with was that beta meant not-yet-profitable.
What is confusing about Friendster and many other projects under the social software umbrella is that the people behind these technologies are approaching design and deployment in a fundamentally different way. Rather than building locked-down versions, the designers of these systems are „shipping" hacked up systems and constantly morphing them depending on how people used them.

Let's return for a moment to thinking about the traditional software design process. Anyone who has ever worked in a large technology company knows the pain of this process. An idea emerges, dramatic process ensues (usually involving Powerpoint decks and monetization justification and market speculation and blah blah blah), the product people design a spec[4], the engineers build it to that spec, the quality assurance team tests the system, the usability people make certain that people can use it, the legal team makes certain that it can be shipped (and adds unreadable mumbo jumbo and a button to sign away your firstborn), the marketing team builds out a deployment plan, and on and on and on... until you die of bureaucratic overload before the product ever ships. No wonder things take years to gestate and so many people quit burnt out from meetings!
Let's now look at a piece of social software that bucks this traditional process in nearly every way: MySpace. The folks behind MySpace thought that Friendster was lame and making foolish decisions; they thought they could do better (just like every major corporation out there). They hacked together some code based on Cold Fusion (which, for the non-technical readers, makes most engineers shudder) and put it on the web. They invited their friends, those who didn't want to be on a dating site, and those who were getting kicked off of Friendster to join. There was no spec, no quality assurance, no usability... (OMG there was no usability...) There was no legal, no marketing. They just deployed. From idea to deployment, it was a matter of months. No traditional company could even begin to compete in that time frame.

It's easy to shrug our shoulders and say, well, they just got lucky... but that's not the full story. After the system deployed, they started talking to their users. They reached out to musicians and asked how they could support them better. For example, they created the simple URL (y'know - myspace.com/tom) to help bands advertise their profiles. They watched what weird things people did and the figured out how to support that.

Compared to MySpace, Friendster is a traditional dinosaur. With MySpace, things were so rapid that you'd miss something if you didn't log in hourly. They coded straight to live servers and when bugs brought down the system, they apologized. When users clamored for a feature, they tried to provide for them (even if this provisioning was held off to make them more money). When people starting hacking the system, they allowed it and even made it easier (except for those who took this power too far). Even as they grew, rather than hiring quality assurance or usability teams, their founder preferred to hang out on the system and respond to people's reactions. If people complained, they changed it... if people reported bugs, they fixed it. Why go through the layers of time-consuming bureaucracy to launch something when your users could tell if you it was working or not? Furthermore, why create specs when you can just launch features piecemeal and see if they take off? Of course, tell this to any traditional exec and you might be accused of involuntary manslaughter for the heart attack you spark.

Of course, MySpace is extreme in its practices, and the costs are phenomenal. You will be hard-pressed to find a time when Tom Anderson (one of the site's founders) is not on the site unless he is sleeping. And he doesn't really do much of that. But this ethos reflects some of the key design values of the social software movement:

1) Hack it up, get it out there.
2) Learn from your users and evolve the system with them.
3) Make your presence known to your users and invite them to provide feedback.
4) When you make mistakes, grovel for forgiveness; you're human too!

(One might also include: Monetization? What's monetization? Oh, right, remember to add ads somewhere to keep investors happy.)

Needless to say, there are pros and cons to each of these common practices. Let me just highlight one of each.
First, a CON. This approach produces terribly unstable code that is poorly documented, fails any extensibility test, and is often held together by magic that not even the engineers understand. As a result, once these systems are out and rolling, plumbers are constantly needed to plug the leaks. Of course, by apologizing profusely, this can be considered a feature, not a bug.
More importantly, a PRO. Usability is based on a human interaction paradigm. Bring a potential user into the lab, give them a set of tasks to do and see how well they understand the system. Sure, this process has allowed people to figure out how many pixels wide things have to be and how to label a button, but it's nearly useless for social software. As any pop sociologist knows, when groups gather, the behavior of the crowd is always different than the behavior of the individuals. This is even more true online where the mischievous side of so many people's imagination takes hold. And it is magnified by the size of the crowd. In this way, Shirky's later definition of social software is extremely appropriate - social software is indeed stuff worth spamming because you know you'll have an audience. No lab study can properly capture the dynamics that engender such peculiarities.

(I'll let the MBAs figure out the pros and cos of monetization... To many designers, this is like asking a hippie about capitalism. Push him hard enough and he'll acknowledge that having money to buy luxuries like clothes and medicine is important, but he still likes to spout ideals of living off the land.)

The spread of social software

Once social software is out in the wild, the next question emerges: who is going to use it? How?
Big companies have big marketing budgets. Super bowl ads, subway banners, messages to current users, etc. Yet, social software has been mostly about startups and entrepreneurs reaching millions of people. How on earth does that happen?
Social network sites are not the only social software to rely on social networks to spread the word. Most social software has started out being used by the friends of the developers and spreading to different populations based on what takes hold. Organic growth is at the heart of social software.
All marketers know that the stickiness of a product is greatly increased by learning about the product through friends instead of through advertisements. This is why there is so much scheming around viral marketing. But organic growth is not just a means of advertising - it is the primary means in which the culture of a site is formed.
Consider Flickr. When Flickr launched, Caterina and Stewart welcomed all users, told them about what was available, and asked them how they could improve the system. They framed the site as a space for amateur photography, not just as a place to store photos. As users invited their friends, they helped pass along the message, both explicitly and implicitly through the ways that they used the system. When the site was sold to Yahoo!, it received some powerful press and many of the people who joined at that point were not socialized into the community; their practices were fundamentally different than the practices of the „community," forcing Flickr to do some damage control. Thankfully, Yahoo! has never done mass flood advertising for Flickr, allowing the site to grow organically.
Values are built into social software and spread through the networks of people who join. This is why most of these sites are so tech-centric. Because social software is all about the collective, the early values often get reinforced on the entrance page. Consider del.icio.us or Digg - both of their front pages make visible how dominated

these sites are by geeks (who else cares about the details of AJAX?). This can be - and often is - alienating for non-geek newcomers, but it is extremely tricky to shift a culture once it becomes deeply rooted in the site. It is not impossible, but it is definitely an uphill battle.

Consider Orkut. Orkut is an absolute phenomenon in Brazil but it is often mocked in the American press as worthless since its American audience is abysmally small. Part of what makes Orkut very interesting is the „Brazilian takeover." Let me explain because this highlights ways in which unexpected outcomes can emerge through design decisions and, when unchecked, things can spiral things a bit out of control.

Orkut has a demographics page; in its early inception, this page was quite prominent. On one side, Orkut lists nation-state flags and the percentage of Orkut that is comprised of people from that nation-state. This flag-ordered listing is quite similar to something familiar to all sports fans and any non-American would recognize such a display from the World Cup. *(I'm sure that it doesn't shock anyone to realize that most Americans assume their country will be on top and pretty much ignores any sport where this is not true.)* A handful of Brazilian ex-pats living in the States joined. They received invitations because of their links to the Silicon Valley tech world and they invited their friends back home to join. By this point, Americans were pretty tired of yet-another-social-network-site and so most of the Unite States-based early adopters on Orkut were pretty inactive. At some point, folks in Brazil started realizing that they could beat the US in the listings. Messages spread in Portuguese, encouraging people to join Orkut and overtake the Americans. At some juncture, there was a tipping point after which it was seen as a Brazilian site and Brazilians joined without such nationalist encouragement.

Interestingly, Orkut is starting to take hold in a new location: India. Given some structural similarities between Brazil and India, this makes sense, but it has also taken a lot of work on the part of Google. What's most interesting is that, as it spreads through India, caste structures are being translated to and reinforced by the site. I suspect that this will be a very fascinating thing for some researcher to follow…

Let me shift focus for a moment. Sure, it's fascinating that social software sites tend to take hold in specific cultures, but so what? If it's such a good piece of software, shouldn't it work in any cultural context? As these sites expand, won't they just eventually grow to support all cultures? The answer is flatly no.

Cultural cues provide context. When a culture forms on a site, it provides meaningful contextual information to visitors. Context allows people to understand how to behave and what to expect from others' behavior. If the contextual cues are culturally illegibile to the visitor, they walk away. Let's go back to Friendster for a moment.

There were three dominant early adopter groups - gay men living in New York, attendees of Burning Man[5] („Burners”), and bloggers. Of course there was overlap, but for the most part, each group thought that the site was about them. Because of the structure of the site, when they logged in, they only saw people like them. Thus, when they crafted their Profile, they did so in a way that reflected that facet of their identity. For example, when Burners logged in, they filled their Profiles with half-naked pictures involving lots of fur set in a desert backdrop. They used their „Playa names”[6] and made innumerable references to music and partying, even though many of them were teachers, doctors, engineers, etc. in their professional life and would never discuss such things in a public setting that their bosses might attend.

We always use contextual cues to figure out what's appropriate to say. I'm assuming that you're interested in technology so I'm making passing references to MySpace and Friendster without explaining what these sites are. Yet, I took the time to explain Burning Man because you might not be a part of that community. Having a sense of one's audience allows you to make assumptions and take shortcuts in discussing matters. If I had to stop and explain everything, this essay would be dry and painful. Of course, I also take the risk that I'm speaking over your head or boring you with things you already know because I have no idea who you, dear reader, are. I'm imagining who I think might be my audience as I write this. I'm

trying to speak in a way that's relevant for you, regardless of who you are. That's not saying that I'm succeeding (you are the judge of that). But, ::shrug:: I'm trying.
This is quite different from speaking in unmediated physical spaces where I can see who can hear me and I can watch their body language for feedback. I can also get a sense of what's appropriate from spatial cues. For example, if I'm in a bar, I'm probably going to be far more crass than when I'm at an opera house.
Now, imagine a physical setting where you're supposed to share part of your life simultaneously to your boss, your children, your spouse, your drinking buddy, your mother, your neighbors, etc. The only way we handle this is through social scripts (this is what weddings are all about). Most public situations aren't like that. Most public situations provide contextual cues so you know how to behave. It's not a matter of deceptions; it's because you talk to your daughter differently than you talk to your colleagues.
Online, things aren't that simple. While Usenet groups might be split by topic, a little search will collapse that right down. All of a sudden, you can see that someone is present in both comp.lang.perl and alt.sex.bondage. My guess is that when that person was writing to alt.sex.bondage, they weren't writing for the comp.lang.perl or the search engine audience. While we know how to behave in public, we don't know how to behave in a front of a potential, unknown audience of all people across all space and all time. Even celebrities and politicians are terrible about this even though they negotiate such audiences through mediated technologies like TV all the time.
In Joshua Meyrowitz's *No Sense of Place*, he tells the story of Stokely Carmichael's dilemma. Carmichael, an American civil rights activist in the 1960s, regularly spoke in front of a wide array of different audiences. When he talked to white politicos, he used a very posh, „educated" vernacular. When he spoke in front of southern black church audiences, he used a very different speech pattern that terrified white society. As he gained notoriety, he was encouraged to speak on radio and television. The problem with that is that there would be both white and black audiences. There was

no neutral way of speaking; he had to choose. Carmichael decided not to turn his back on his people and, to this day, Black Power is seen as anti-white.
Guess what? Online, folks are facing the same problem. The early adopters of Friendster developed social norms based on an assumption of a limited audience; when the audience expanded, they were ill-prepared to be exposed to a much broader public that wasn't present when they initially crafted their expressions. Teenagers on MySpace aren't prepared for their parents – how can one be simultaneously cool to parents and peers when the norms of each are quite different?

Behavior on social software

Now we're at the heart of the third issue: how social software creates shifts in behavior. Behavior is about context and context begins with the designers, morphs with the early adopters, and continues on diversifying as a technology spreads. What is most interesting about context in social software is that it starts with people. Let me back up for a moment.
When researchers took over Arpanet, they were interested in ways of sharing research amongst scholars. Usenet is a good example of software that came out of this era as it was designed to support groups of people gathering around specific topics. This continued on beyond the academic roots. Context was conveyed topically - all people on the web interested in cats, join here. This made sense when everyone on the Internet was relatively homogenous but it didn't take long until pranksters decided to join the cats group and post messages about how to cook cats, outraging many of the active members who had built a community based on very different suppositions. More problematically, this did not scale. Imagine if everyone on the net who loved cats tried to have a conversation now. Eeek. Instead, we're resigned to watching CuteOverload[7] every day for our broadcast cat fix.
Most social applications pre-boom were all about connecting people around topics - Usenet, bulletin boards, mailing lists, etc. Even MUDs and MOOs, like today's

MMORPGs, connected people based on a topical task even if the primary motivation to participate was random chatter. Sure, there were one-to-one systems that supported known people (email, instant messaging) but these weren't really built as public spaces for a variety of people to participate.

Then the boom hit and everything became commercial commercial commercial. Social uses of technology barely matured during this period although their usage continued to grow. And then Crash. Bang. Bust. The commercial ventures collapsed but social interaction online continued to flourish.

When the social software movement emerged, along with it came a new way of building context. Sure, there were plenty of services built to connect people in new ways around topics - Meetup being a good example. But, really, most of what emerged was people first, topic second. People connected to people that they knew or people they respected and communities were built around social ties, around the same social networks that provided the organic growth of the network. While early social technologies were about finding people with similar interests, the latest round is far more about connecting to people and watching shared interests emerge through that.

This is also why context is so radically different in social software. Rather than being defined by the topic, it's defined by the egocentric collections of people. In other words, those who joined Friendster and assumed that everyone was like them did so because of the way the site was designed – the structure is inherently egocentric. This is also where things get tricky because egocentric communities cannot support that many different contexts. And thus, what you see, is people using multiple sites to keep contexts separate. This is also why proposals like FOAF are nervewracking - folks don't necessarily want to deal with multiple contexts in the same place... In fact, they can't.

The problem is that monetization is hanging on the tip of everyone's tongues again. To make money, sites have to grow. To grow, they have to expand beyond comfortable context borders. There is no better example of that than when Facebook announced that it was opening doors to everyone. Part of what made

Facebook so desired was that the context really was college, for better or worse... It got complicated when high schoolers joined and then when companies began doing their recruiting there. But your mom? Who wants your mom to join you at college? Regardless of whether or not these fears are rational, their decision to be more open eliminated the rite of passage that many college students felt when they could first join the site because they were newly minted freshmen. Of course, Facebook needs to grow.

This is where we reach an interesting challenge of social software: to what degree can it scale? And when will growth kill the system? As the community-oriented sites grow and gain profitability, investors and stockholders push for more growth, more profitibility... even if it will kill the system in the process. Is growth worth destroying a system that could be a stable source of income?

Friendster could not sustain its growth and people flipped when they were faced with multiple contexts. MySpace is definitely on sketchy ground as parents, law enforcement, and commercial entities join to befriend teens. Flickr is faced with increasing siloing of people's photos and is in constant battle with those who want to use it as porn distribution. Even on del.icio.us and Digg, many early adopters are pissed off that there are so many „uninteresting" articles being posted... whereby uninteresting means that they are more interesting to the masses than to the niche group who were the early adopters.

Ironically, the only piece of social software that is really scaling well is blogs. Of course, the reason that's true is because the blogosphere is mostly distributed and so the political bloggers seem completely unaware of the food bloggers and the mommy bloggers are completely disconnected from the emo kids. Only on Technorati does it all come together and create a massive headache because how do you really rank apples and oranges? Luckily, the vast majority of bloggers aren't concerned about rank; they're just blogging to hang out with their friends.

Conclusion

Let me begin by raising questions for a few relevant social groups.

- Designers: Social software has introduced new ways of building and deploying systems. Yet, there are costs to these chaotic processes. What innovations in design process are needed to move forward and better support the emergence of these systems so that they support users in a meaningful way?
- Researchers: Social software is introducing new types of social organization, but it's also complicating the way people navigate sociality. What are the implications of this for society and culture?
- Businessfolk: Monetization and growth are central to your view of the world but efforts to push this in social software have destabilized it. This has great potential costs for you. Are there ways to rethink the scaling process to make social software more economically viable without killing the communities in the process?

When I think of the term „social software," I still want to roll my eyes but when I think of the radical shifts that have happened in design process, flow of information, and interaction paradigms under the movement connected with the term, I can't help but smile. These shifts are quite significant both for the tech sector and for the millions of people who are engaging with these technologies. Of course, in gushing over the new, I don't want to forget the old. There are innumerable lessons to learn from earlier experiments and it would benefit everyone if we took a moment to learn from what came before. Of course, that might be wishful thinking because I'm always amazed at how people are unable to learn from the failures that are happening right now. Still, while we celebrate all of what is new, let us not forget the significance of what is old. In this way, we can build on the shoulders of giants rather than reinventing the wheel.

1 Allen, C. 2004. "Tracing the Evolution of Social Software." *Life With Alacrity* (blog), October 13. http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html.

2 Shirky, C. 2004. "Blog Explosion and Insider's Club: Brothers in cluelessness." *Many 2 Many* (blog), October 6. http://many.corante.com/archives/2004/10/06/blog_explosion_and_insiders_club_brothers_in_cluelessness.php.

3 Coates, T. 2005. An addendum to a definition of Social Software. *Plastigbag.org* (blog), January 5. http://www.plasticbag.org/archives/2005/01/an_addendum_to_a_definition_of_social_software/

4 Spec is short for "specification." It's a formal articulation of what features the product will contain.

5 Burning Man is an annual arts festival held in the desert of Nevada (United States). It has a reputation for fire, nudity, drugs, music, art, and community bonding. For more information, see http://www.burningman.com/.

6 "Playa names" are nicknames that people use when they are at the festival and when they socialize with others who regularly attend the festival.

7 http://www.cuteoverload.com/