

Prepared by: Clay Shirky

We've all gone to school on Slashdot and eBay. In those cases, their growing popularity in the period after their respective launches led to a tragedy of the commons, where open access plus incentives led to nearly constant attack by people wanting to game the system, whether to gain attention for themselves or their point of view in the case of Slashdot, or to defraud other users, as with eBay.

The traditional response to these problems would have been to hire editors or other functionaries to police the system for abuse, in order to stem the damage and to assure ordinary users you were working on their behalf. That strategy, however, would fail at the scale and degree of openness at which those services function. The Slashdot FAQ tells the story of trying to police the comments with moderators chosen from among the userbase, first 25 of them and later 400. Like the *Charge of the Light Brigade*, however, even 400 committed individuals were not up to the task. The very presence of effective moderators made the problem worse over time; when the improved moderation saved the site from drowning in noise, more users joined, but this increase made policing the site harder, eventually breaking the very system that made the growth possible.

EBay faced similar, ugly feedback loops; any linear expenditure of energy required for policing, however small the increment, would ultimately make the service unsustainable. As a result, the only opportunity for low-cost policing of such systems is to make them largely self-policing. From these examples and others we can surmise that large social systems will need ways to highlight good behavior or suppress negative behavior or both. If the guardians are to guard themselves, oversight must be largely replaced by intrasight, designed in such a way that imbalances become self-correcting.

The obvious conclusion to draw is that, when contemplating the a new service with these characteristics, the need for some user-harnessed reputation or ranking system can be regarded as a foregone conclusion, and that these systems should be carefully planned so that tragedy of the commons problems can be avoided from launch. I believe that this

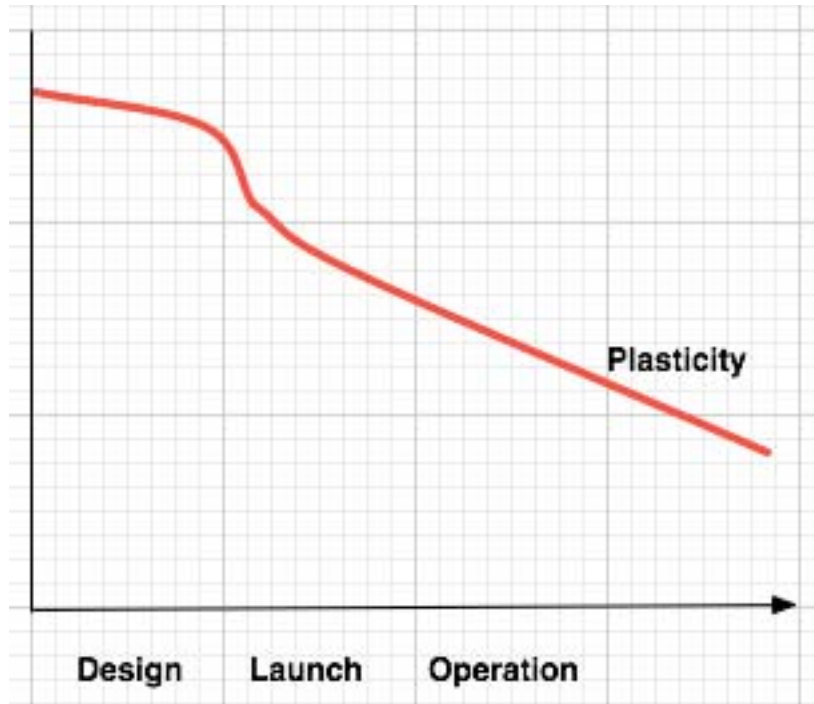
conclusion is wrong, and that where it is acted on, its effects are likely to be at least harmful, if not fatal, to the service adopting them.

There is an alternate reading of the Slashdot and eBay stories, one that I believe better describes those successes, and better places Community Patent to take advantage of similar processes. That reading concentrates not on outcome but process; the unfolding of the history of the Slashdot reputation system should teach us not "End as they began -- build your reputation system in advance" but rather "Begin as they began -- ship with a simple set of features, watch and learn, and build reputation and ranking only after you understand the problems." In this telling, constituting users' relations as a set of bargains developed incrementally and post hoc is more predictive of eventual success than simply adopting any residue from previous successes.

As David Weinberger noted in *The Unspoken of Groups*, in social settings clarity is violence. You don't get 1789 without living through 1788; successful constitutions, which necessarily create clarity, are typically ratified *after* a group has come to a degree of informal cohesion, and are thus able to absorb some of the violence of clarity, in order to get its benefits. The desire to participate in a system that constrains freedom of action in support of group goals typically requires that the participants have at least seen, and possibly lived through, the difficulties of unfettered systems, while at the same time building up their sense of membership or shared goals in the group as a whole. Working reputation systems have historically been fit to their situation only after that situation has moved from theoretical to actual; both eBay and Slashdot moved from a high degree of uncertainty to largely stable systems after a period of early experimentation. However, this has not committed them to continual redesign. In those cases, systems designed after launch but early in the process of users adoption that have survived to this day with only relatively minor adjustments. Even Digg, arguably the most successful service to design a reputation system in advance, had an enormous amount of prior art directly in its domain (/., k5, mefi, et al), and still ended up with serious re-design issues that came from discovering that its advertised virtues of democratic editing were unsustainable.

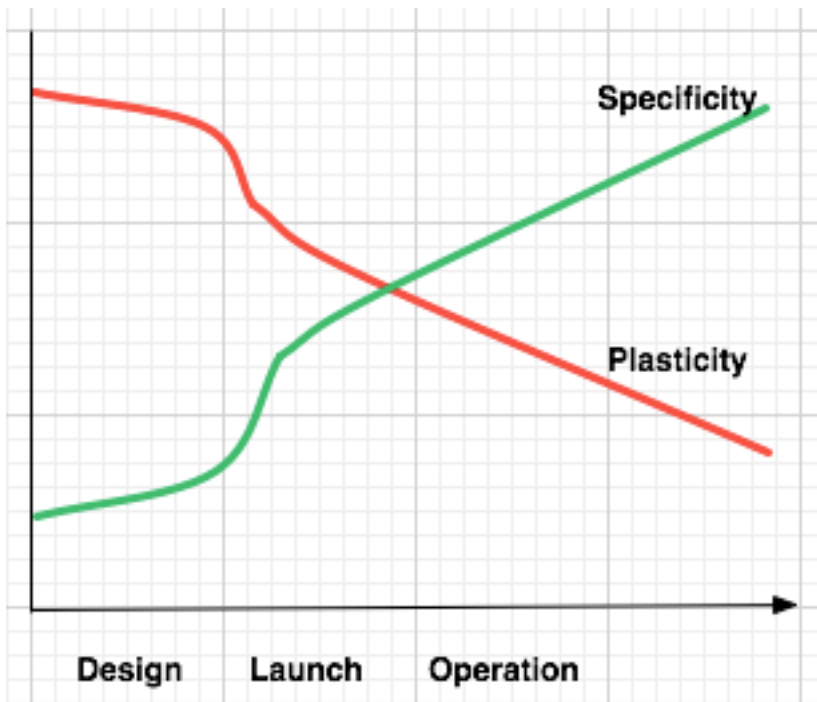
## The Argument in Two Pictures

The argument I am advancing can be made in two notional graphs.



The first graph concerns plasticity, the ease with which any piece of software can be modified. Plasticity decays with time. It is highest at the outset, when a project is in its most formative stages. It is easier to change a list of potential features than a set of partially implemented features, and it is easier to change partially implemented features than fully implemented features. Especially significant is the drop in plasticity at launch; even for web-based services, which exist only in a single instantiation and can be updated frequently and for all users at once, the addition of users creates both inertia, in the direction of not breaking their mental model of the service, and caution in upgrading, so as not to introduce bugs or create downtime in a working service. As the userbase grows, the expectations of the early adopters harden still further, while the expectations of new users follows the norms set up by those adopters; this is particularly true of any service with a social component.

An obvious concern with reputation systems is that, as with any feature, they are easier to implement when plasticity is high than when it is low. Other things being equal, one would prefer to design the system as early as possible, and certainly before launch. In the current case, however, other things are not equal. In particular, the specificity of information the designers have about the service and how it behaves in the hands of real users moves counter to plasticity over time.



When you are working to understand the ideal design for a particular piece of software, the specificity of your knowledge increases with time. During the design phase, the increasing concreteness of the work provides concomitant gains in specificity, but nothing like launch. No software, however perfect, survives first contact with the users unscathed, and given the unparalleled opportunities with web-based services to observe user behavior individually and in bulk, in the moment and over time, the period after launch increases specificity enormously, after which it continues to rise, albeit at a less torrid pace.

There is a tension between knowing and doing; in the absence of the ideal scenario where you know just what needs to be done while enjoying complete freedom to do it (and a pony), the essential tradeoff is in understanding which features benefit most from increased specificity of knowledge. The two characteristics that will tend to push the ideal implementation window to post-launch are when a set of possible features is very large, but the set of those features that will ultimately be required is small; and when culling the small number of required features from the set of all possible features can only be done by observing actual users. I believe that both conditions apply a fortiori to reputation and ranking.

### **Costs of Acting In Advance of Knowing**

Consider the costs of designing a reputation system in advance. In addition to the well-known problems of feature-creep ("Let's make it possible to rank reputation rankings!") and Theory of Everything technologies ("Let's use RDF!"), reputation systems create an astonishing perimeter defense problem. Social networks are degenerate, which is to say that there are multiple alternate paths to similar goals -- someone who wants to act out and is thwarted along one path can readily find others. As a result, the number of possible ills you can imagine in advance is typically much larger than the number of ills that manifest themselves in functioning communities. As you will not know which of these ills you will face, the perimeter you will end up defending will be very large and, critically, hard to maintain. The likeliest outcome from such an a priori design effort is inertness; a system designed in advance to prevent all negative behavior will typically have as a side effect deflecting all behavior, period, as users simply turn away from adoption. Should the system achieve escape velocity, however, a second difficulty awaits: every Deathstar ships with an exhaust port.

Working social systems are both complex and homeostatic; as a result, any given strategy for mediating social relations can only be analyzed in the context of the other strategies in use, including strategies adopted by the users themselves. Since the user strategies cannot, by definition, be perfectly predicted in advance, and since the only ungameable

social system is the one that doesn't ship, every social system will have some weakness. A system designed in advance is likely to be overdefended while still having a serious weaknesses unknown the designer, because the discovery and exploitation of that class of weakness can only occur in working, which is to say user-populated, systems. (As with many observations about the design of social systems, these are precedents first illustrated in *Lessons from Lucasfilm's Habitat*, in the sections "Don't Trust Anybody" and "Detailed Central Planning Is Impossible, Don't Even Try".) The worst outcome of such a system is collapse (the Communitree scenario), but even the *best* outcome would still require post hoc design to fix the system with regard to observed user behavior. You could save effort while improving the possibility of success by letting yourself not know what you don't know, and then learning as you go.

### **In Favor of Instrumentation Plus Attention**

The N-squared problem is only a problem when N is large; in most social systems N does not grow large for some time. (Indeed, this scaling up only over time typically provides the ability for a core group to inculcate new users a bit at a time, using moral suasion as their main tool.) As a result, in the early days of a system, the designers occupy a valuable point of transition, after user behavior is observable, but before scale and culture defeat ready intervention. To take advantage of this designable moment, I believe that what Community Patent needs, at launch, is only this: metadata, instrumentation, and attention.

**Metadata:** There are, I believe, three primitive types of metadata required for Community Patent -- people, patents, and interjections. Each of these will need some namespace to exist in -- identity for the people, and named data for the patents themselves and for various forms of interjection, from simple annotation to complex conversation. In addition, two abstract types are needed -- links and labels. A link is any unique pair of primitives -- this user made that comment, this comment is attached to that conversation, this conversation is about those patents. All links should be readily observable and extractable from the system, even if

they are not exposed in the interface the user sees. Finally, following Schachter's intuition from del.icio.us, all links should be labelable. (Another way to view the same problem is to see labels as another type of interjection, attached to links.) I believe that this will be enough, at launch, to maximize the specificity of observation while minimizing the loss of plasticity.

**Instrumentation:** As we know from collaborative filtering algorithms from Ringo to PageRank, it is not necessary to ask users to rank things in order to derive their rankings. The second necessary element will be the automated delivery of as many possible reports as can be productively imagined, and, at least as essential, a good system for quickly running ad hoc queries, and automating their production should they prove fruitful.

**Attention:** This is the key -- it will be far better to invest in smart people than smart algorithms at launch. If we imagine an average of 10 unique examiners per patent and 10 comments per user, then a system with even a thousand patents will be relatively observable without complex ranking or reputation systems, as both the users and the comments will almost certainly exhibit power-law distributions. In a system with as few as ten thousand users and a hundred thousand comments, it will still be fairly apparent where the action is, allowing you the time between Patent #1 and Patent #1,000 to work out what sorts of reputation and ranking systems need to be put in place.

This is a simplification, of course, as each of the categories listed above presents its own challenges -- how should people record their identity? What's the right balance between closed and open lists of labels? And so on. I do not mean to minimize those challenges. I do however mean to say that the central design challenge of user governance -- self-correcting systems that do not raise crushing participation burdens on the users or crushing policing barriers on the hosts -- are so hard to design in advance that, provided you have the system primitives right, the Pentagon strategy of OODA -- Orient, Observe, Decide, Act -- will be superior to any amount of advance design work.