Telling Stories - What Homer, Dickens, and Comic Books can teach your (social) software

Tom Armitage | tom@infovore.org | http://infovore.org

Reboot 8, Copenhagen, 2nd June

Preface

Hello. My name's Tom Armitage. To quickly get the biography out of the way: I currently work as a client-side developer at a publishing company. In my spare time, I do all manner of hacking and bodging on the net, which I'll happily chat about after this talk.

Prior to my current job, I spent two years doing all manner of internet things - design, code, editorial, ideas - for a weekly British political magazine. And prior to that, I went and got myself a degree in English Literature.

That last fact is probably the most relevant fact to the talk I'm here to give you today, which is called *Telling Stories*.

There are a number of subtitles that I could give this talk; one is on the screen behind me, but I think the easiest way to explain it (for now) is "social software design through the lens of serial fiction".

Actually, I'm not sure how well that explains it, but it's a start.

Let's do a quick survey. So: raise a hand if you've got a weblog. OK? Now, raise another hand if you've got a Flickr account. Next up: another hand if you've got a Livejournal - start waving one hand if you've run out of limbs. Next, a Myspace account. Does anybody podcast?

Congratulations. Everyone with one or more hands raised is a serial publisher.

Hands down. Now, one last pair of questions: raise one hand if you have ever written anything, said anything, or published a picture, to any of these services, which was not strictly the whole truth. You don't have to own up to lieing outright - anything you've dramatised, conflated, enhanced, to make it read better - if it's not exactly what happened, up goes the hand.

Right. If you raised a hand to that one, then congratulations, you're not just serial publishers; you're involved in serial fiction.

Let's go back to the first set of questions, and take a trip back to the nineteenth century, where we'll look at the serial publishing of novels.

1. Serial Publishing

"Make 'em cry, make 'em laugh, Make 'em wait – exactly in that order." – Wilkie Collins

Nowadays, our experience of Dickens' novels is as single-volume books, a couple of hundred pages long. But that's not how they were originally published. Instead, they were published in monthly installments, much like comics today. The format emerged through a tax loophole in the 17th century; newspapers, having to pay tax on the paper used in their printing, started printing on larger sheets of paper so that they could be classified as "pamphlets", and thus fall into a lower tax bracket. Of course, once you've moved to bigger paper, you need more content to fill it, so the papers would look to writers to serialise works of fiction, which would fill space. The tax law was eventually repealed, but the readers still wanted their serials, and so the papers continued to deliver them. A good serial could boost circulation - they were the DVD giveaways of their day.

Dickens' first novel, *The Pickwick Papers,* published in 1836, was a smash hit, and re-invigorated the serial format. The novel was a stand-alone publication, not inserted into anything else, consisting of 32 pages of type, loosely stitched into a flimsy cover, and with a pair of illustrations. It cost a shilling. 18 issues contained the full novel - 17 single issues, and a double issue to end on, with 64 pages, four illustrations, and the necessary table of contents and title page so that you could get the whole lot bound as a "real book".

There were many advantages to the format. For the publisher, it meant that he would recoup costs faster - each issue would practically pay for itself, as opposed to waiting for a book to be written, then printing large copies at great expense, and then selling them at a fairly high price (31s 6d, a guinea and a half) and hoping to make some money. The format lent itself obviously to advertising - again, look at comics today - which pleased the publishers no end, and because, spread out over eighteen issues, a single novel might garner multiple reviews, there were more opportunities for the book to market itself.

For authors, the advantages were being able to deliver content more steadily over time, being able to react to criticism and audience reaction, and alter the book (or not) accordingly, and being paid on a regular basis - rather than waiting for a book to be printed, published, sold, and its royalties divided.

It's important to note how manageable the serial format makes large, sprawling works; suddenly, an obvious structure is created, and Dickens took great advantage of this, building climaxes in to his works at the halfway mark, and in later books, the quarter-marks; his final book, *Our Mutual Friend*, was divided into four "books" before the finer-grained division of issues kicked in.

So what's this got to do with the web? And social software?

Let's think about placing data on display. When I place data on public display, that's publishing, right? Well, I'm going to assume that. Any time I display content publicly on the web - and by "publicly", I mean so that Google could see it, always a good yardstick - I'm publishing. For now, a good model to think of is a blogging tool.

So I have this tool, and I'm publishing to the web. By and large, blogs are fragments. Whilst each fragment stands alone, there is a "whole" being built at the same time, whether I like it or not. There's an ongoing story. If you look at a personal blog - a journal, effectively - those individual entries add up to a perspective on a life. Not a complete life, not a complete human being; just the story of a life.

The "modularity" of these individual fragments - be they text or otherwise - is very similar to writing a novel. In a recent interview, the author David Mitchell said

"I think all novels are actually compounded short stories. It's just the borders get so porous and so squished up that you no longer see them, but I think they are there. And I do structure my novels in that way. One of the commandments of Black Swan Green *[Mitchell's most recent novel] was to write a novel made of chapters that are theoretically extractable short stories."*

And before there's the sense of a completed work, these fragments start to hint at their desire to be interconnected. Mitchell again:

"Short stories have a background white noise that creates the illusion that the world is much bigger than the mere 10 or 15 pages, and I wanted to see if I could sync up the white noise of the background of short stories."

And, again, it's a stage many have been through with weblogs, and that other tools have also started to hint at: when we realise we're not just throwing data into a box which happens to have a window, but putting on an ongoing show. It's when you start to raise your game.

When I post a blog, a podcast, a photo; when I bookmark a link or tag a photo; there's a date and time attached. It might be public, it might be hidden in the system, but it's there. And if not, *it should be*; I'm sure you're all aware of this.

Whenever I publish anything with a date attached - there's a framework for ongoing narrative. The item published is our narrative, but the date gives it ongoing-ness. It takes time for the pattern to emerge; initially, throwing data at that black box, it seems random. For instance: I upload photos to Flickr at arbitrary intervals. I go silent on my blog without explanation. It may seem, in the short-term, like a blip, but in the long-term, it's an important part of my story. My blog is full of delicious bookmarks right now because I've been busy at work, and writing this talk. That'll be reflected in the longer game, when I write my post-Reboot blog entry, and suddenly the pattern becomes clear.

I used to have a blog called *tajmahal*, from Feb 2001 to Jun 2003. Then I got my current domain/blog, *infovore* (http://infovore.org). And I never imported my old content. It was what I wrote at University, and I felt slightly embarrassed by it, especially the early, awkward years. My new blog, right now, is mainly full of stuff from del.icio.us and serious, technical postings - it feels different. Until you see the join. The edges are different, but the end of the first and the beginning of the second are a perfect join. It flows. I realised my mistake - it was all interconnected after all! And now I can rectify that; I can import my student years to my current blog, maybe tag them differently, and flesh out my story.

It's important to be able to collect data across boundaries - not just chronological, but also boundaries of my software and the software of other people. So allow users to import and export to standardised formats, and give them APIs to connect their stories. The stories we tell are part of a whole, spread across multiple platforms - my photos, my journal, my blog, my links, the books I buy, the films I see - this is all one collected story - my story. Social software should support that interconnectedness.

Don't just consider collecting data across digital boundaries; collect it across a physical one, too. Nothing beats hard copy. It's like part-work - you know, 60 issues to build a model boat; I've thrown all this data at a black box - now I want it out as a perfectly formed product. Hence Lulu, Cafepress. Flickr books. Do Odeo do CDs, MP3 or audio? They should. If there's a logical hard-copy format, investigate it; it may be expensive, but you should consider it nonetheless.

Whilst we experience the present serially, we experience the past as condensed, refined volumes. And we'll always want to return to that past. In 1766, Horace Walpole wrote:

"I almost think there is no wisdom comparable to that of exchanging what is called the realities of life for dreams. Old castles, old pictures, old histories, and the babble of old people, make one live back into centuries that cannot disappoint one. One holds fast and surely what is past. The dead have exhausted their power of deceiving..."

Never underestimate the value of what you're doing now. Or what users do now. Because one day, it will be these old histories, old pictures, old castles, the babble of old people; hindsight brings value, as does nostalgia, not to mention the factual record. So our software needs to be built not so much for a long 'now' but for a long lifetime of use - during which it may be considered "present", it may be "past". And don't define that term - let your users define it. One person's past is everything before they left their husband; another's is everything before they had a child; another's is everything before today. It's a subjective thing, the past; so what you can do with "old data" - export, publish, print - should be able to be done to any data. That may sound obvious, but consider the many ways the term "Archive" is used in web software; I hope you see my point.

Even though things may be 'past', the constant ongoing discussion around our stories brings them back into the present. These stories don't have endings, by and large. They're persistently ongoing. I found a lovely quotation when researching this, in an essay, referring to the advantage of ongoing discussion:

"...having the novel discussed in the periodical press monthly further blurred the distinction between reading a book, a closed container whose contents are enjoyed in a private space and time separated from diurnal activities, and reading an installment that is simultaneously circulating, through reviews and conversation, in the public world of daily events."

In social software, and online publishing, we attach the ongoing discussion to the original source: comments. But also blog posts and trackback, forum discussions around other items, Livejournal threads on syndicated blogposts. Our stories are ongoing, consistently circulating; months after the fact, someone makes our recipe, recognises themselves in our picture, begs to differ with our blogpost, and the story is enlivened again. What Dickens could have done with Technorati!

Finally, to wrap up this section, consider the importance of feedback loops. I mentioned the advantage that writers could anticipate reader response, and react accordingly. You should let your users do this; they need a feedback loop. This is why Google bought MeasureMap. They already had Blogger, but when you have a publication tool but no statistics, you have a megaphone and a set of ear defenders.

With MeasureMap, budding publishers now get feedback, in the form of statistics. And suddenly you can see what people think, and react (or not) accordingly. It's not just about ego - either removing it or spurring you to write more posts about Sudoku - it's about taking advantage of the serial medium. Novelists today trust their editor and the publishing house, and have to publish and be damned. Serial publishing isn't like that, and so the software we do it with should be similarly well informed. So collect stats everywhere - item views, item downloads, clicks, links, proportions of contacts to non-contacts. It doesn't have to be visible - the user has less need of most of this than you, and you want to keep things understandable - but if you give them the things you want them to harness - your eBay rating, the number of times a photo has been viewed - they'll do what they can to improve it. If you tell users something is important, they will modify behaviour to improve it. And, sometimes, you have to buck the stats. Dickens had to kill Little Nell. He knew it would be a huge tear-jerker, he knew the impact it'd have - which is why he did it. It was a necessary action. The trick was keeping the audience despite tricks like that.

That's serial narrative out of the way. Now, I'm going to go back a bit further in history.

2. Epic

Telling the same story again and again.

Let's turn to the Greek epic poet Homer. His two main works are the Iliad, an epic narrative poem about the Trojan War, and what could be described (very loosely) as its sequel, the Odyssey, which is about Odysseus' journey home from Troy.

So: whilst we experience these two epic stories now as single books, they were originally told as oral performances. The poet would tell these stories, in person, to an audience. These stories are huge, though; how could anyone remember them? The answer is that they're built out upon a structure of convention, epithet, and metre.

There are stock epithets that can be called upon - not just short turns of phrase, but whole paragraphs; there are constructions both of language and of plot that can be called upon. The metre of the Iliad and Odyssey is a regular dactylic hexameter. As a result, it's possible to fit language around the metre. This is best explained thus: Odysseus is referred to by a variety of epithetical names. His name, "Odysseus", is three syllables; however, he's often also called "wily Odysseus", "polumetis Odysseus", which is seven syllables, and (conveniently) the second half of a line of dactylic hexameter. Then you've got the nine-syllable "Odysseus Laertiades" - Odysseus, son of Laertes. So whenever the poet needs to describe something being done by Odys-

seus, he doesn't just have this three syllable name - it can be expanded to fill practically any space in the line.

This poem is entirely built on convention, on formula, but because it's being told live, the poet can ad-lib according to his audience; you can adapt depending upon context, making reference to local things, altering pace according to whether the audience is bored or not.

So whilst you can tell this same story again and again, two tellings are never the same. It's only until much, much later, that these texts get written down and they become fixed.

Anyhow. What's this got to do with software? The whole 'same story again told again and again' concept is the key. And here's one story I've told, about myself, god knows how many times.

Sign-up forms [left]. Bane of my life. I remember this point in 2004 when I'd done this so many times - for Friendster, Orkut, Flickr, filling out who I am, my name and email address, my favourite films, my favourite books, etcetera, etcetera, and getting thoroughly fed up.

You've probably experienced this problem, I'm guessing, signing up for many services with a metric ton of the same data. And you've probably thought that there's got to be a better way.

In my opinion, that better way is not single-sign-on, though. Whilst there's a mass of data that a user could provide about themselves, only some of it is relevant to your product. Some things - name, email address, IM account - are suitable for most software. But some stuff isn't. So don't ever copy somebody else's signup form - think about what you honestly, genuinely need. When we tag data objects, we think up tags on the spur of the moment, and it's the immediacy that gives it value. Encourage users to tell a fragment of their own epic story when they sign up; find one single, epithetical piece of data they can give you, and see what use you can put it to. On a recipe-sharing site, get them to tell you their favourite cookbook, or recipe, or experience; in a scientific community, find out what they're currently working on. The storytelling begins on the user's profile page, and instead of making them trot out something by rote, get them to perform for you; make them call up on their reserves of convention and genre, and come up with something unique to your site.

We should actively expect people's profiles to differ between sites, and encourage this uniqueness. The stories we tell with publishing software are all similar in concept, if not execution. I have a LiveJournal, and I have a weblog, and I use both - but there's no duplicate content. I use both in different ways.

Also: unless absolutely impossible, allow for the possibility of users having multiple accounts. My parallel stories don't just exist in different applications; they might exist in the same application. I contain multitudes, and some of my stories need to be told in different contexts to others. So let the same email address have multiple accounts - don't limit me by that - so I can

tell all these stories in parallel. Some quick examples: a personal flickr profile, and a professional one; a profile for me, and a profile for my company. Just as "bands" friend "bands on Myspace, so my company should be allowed its own "friends", and these are distinct from its members friends; you can use an institution as a persona. Allow me these different contexts - don't assume anything about the story your users wish to tell.

And don't worry too much about continuity, either. In Book 5 of the Iliad, Menelaus kills Pylaimenes, and in book 13 Meriones kills Pylaimenes' son Harpalion - and Homer describes Harpalion's father being in the crowd of mourners. It's not a strict formulaic construction, but the concept is a formulaic one - fathers mourning untimely deaths of sons - and so it's hardly what you call a true "continuity error". Continuity is rarely as important as people make out; the on-going nature of linear tales is, in many ways, their strongest form of continuity.

If continuity is a problem, though, we can always change it.

3. Retcon

We can do that with Retroactive Continuity, or, as it's more commonly known, Retcon.

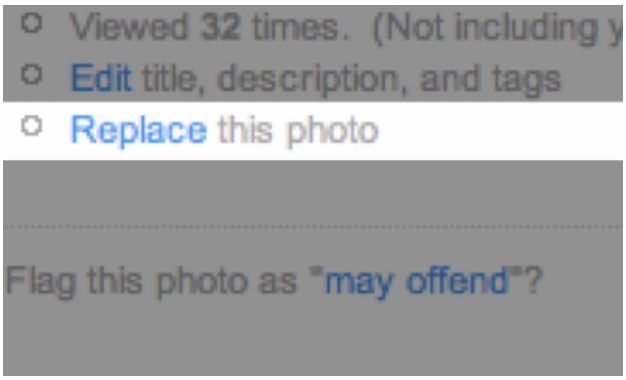
The best way to explain Retconning is probably to quote Wikipedia here as it's a pretty succinct definition:

"the adding of new information to "historical" material, or deliberately changing previously established facts in a work of serial fiction"

Essentially, it's altering continuity by throwing in a change to past history. It's most useful in long, ongoing serial works - like comic books or soap operas - especially when many writers are involved. Some Retcons merely add extra information that doesn't contradict continuity; some entirely alter the past. The most basic of these is characters, long-thought dead, coming back to life; this is a mainstay of soap opera the world over. Alternatively, differing explanations for the same phenomena might occur through a work; in Marvel's *X-Men* comics, the explanation for Jean Grey's incarnation as the Phoenix has been retconned several times. These can be more or less absurd than the "back from the dead" type. The more people you have telling a story - for instance, once spin-offs enter the fray - the more necessary retcons become, if you're to keep things sane.

Comics publishers often have exactly this problem - over twenty years, many writers had written their own takes on popular characters, in series running in parallel, and things no longer quite added up. In 1985's *Crisis on Infinite Earths*, the comics publisher DC realised they had to do something with all the multiple versions of characters, and the many parallel universes that had sprung into being - so, with a big fanfare, they did this year-long series that ultimately merged the universes, killed off a lot of characters, and rebooted continuity. That was the beginning of "explicit retcon as grand feature'; these are rare, but do happen from time to time.

So, again: what does this have to do with software?



This isn't about multiple versions of stories, like my epic metaphor; this is about revising earlier versions. Now, lots of software lets us do this: blogs, forums, bookmarking systems, wikis, photo-storage. And it was this new feature [see left] that gave me the idea. Of course I might want to alter something I've already published; not delete it, not replace it, actually modify it. And sometimes, I don't necessarily want anyone else to know.

For instance: a draft version of a photo gets replaced by a better one, or a better crop. Or I edit somebody out of it. I replace a typo that might have lead to errors later on; I correct some facts incidental to my tale, but important for accuracy; I retrospectively improve a recipe that I've now discovered could be bettered upon. Earlier, I stressed the interconnectedness of stories, so surely changing one component in secret might damage the wider structure? Perhaps. I think you can make a surprisingly wide number of changes to an individual element of data without the whole structure, interconnected over time and across the net, falling apart.

But there still can be downsides that I can't deny - alteration of things not as minor modification, but changing their fundamental nature - replacing something with an entirely new resource. You can take care of this, though. On some systems, it's not necessary for anyone to know that the content of that resource has changed; on some, like forums, where conversation is an important feature, it's very important that other users are notified of my change.

My point is: the functionality should be there. I'm going to change my mind about things I've posted, I'm going to have second thoughts. And deleting and adding again isn't enough, it has to be direct replacement of the resource - it needs the same universal identifier, the same URL. The content has changed, but the fundamentals of the data haven't, so to speak. We all retroactively alter our past, and our software should let us - where appropriate - do this as unobtrusively as possible. We don't need every edit to have a twelve-issue big-event special written about it.

So by now, I'm realising that these ideas could all end up getting quite subversive: multiple identities and logins, retconning information, giving different information to different sites. This isn't meant to be a verbatim recipe for success; these are ideas you can build on or ignore, but do recognise these are impulses that people might want to act on.

If you thought all that was subversive, though, things are going to get worse. Let's talk about one key aspect of telling stories.

4. Fiction

Let's talk about telling lies.

No, let's not. Lies are bad, lies are malicious. Instead, let's talk about telling untruths. Any piece of writing is rarely the whole truth. After all, the whole truth is usually rather boring. Whenever we tell stories, we exaggerate, we edit, not in order to deceive, but in order to make the story more compelling. We improve the narrative in this way. When I was preparing this talk, Andrew Losowsky wrote a fantastic essay (http://tinyurl.com/lug7c) on the subject of truth and fiction, which I can highly recommend it. In that essay, he defines truth as something with

"*no deliberate dishonesty*". I think that's fair enough - it separates it out from complete fiction, but it incorporates the sort of embellishment I've described.

The potential for real fiction - not just minor embellishment - is one you can't deny social software has, if you accept it's a publishing medium. Blogs, obviously, can be used to tell stories; the podcast makes dramatic monologue quite easy. Losowsky has coined what he calls "Flicktion" in a project called The Doorbells of Florence [see right] - photographs taken from life which then have stories attached to them. It's fusing the boundaries of the real and the fictive. And then consider the potential for playful fictions on forums, messageboards, and especially on the massively multiplayer games that are so popular.

Zhu Xiu Yun is one of the most highly regarded m
audiences with manipulated sounds so strange th
reported to medical staff (who are now on standb

- 8 cases of dizziness
- 32 cases of amnesia
- 34 cases of mild vertigo
- 2 inexplicable sensations that a "presence" was
ear
- 12 people having a sudden desire for each othe

and every single one of the assembled 1700 coul
music critic R P Yan as "slighty sweet but not", an

And Forums and Massives raise a really important point when it comes to telling stories: identity.

In any of these systems, I am not me. I'm an online representation of me. On my blog, that's `Tom Armitage` - the resemblance of that name to my real name is important. To all extents and purposes, I'm me. On most irc channels, I'm `twra2` - my old university email address, and also my initials. More casual, but still representational of me.

On LJ I'm `infov0re`. No mention of my real name, but you can probably google and get my blog. On XBox Live, I'm `PalefaceX`. Which is nothing like my real name, and though (because of voice-communication) you can hear me, it's still an abstraction: I don't want my real name on that service, really; I want to control that information. And I'm not telling you what I am on some things.

My point is: the further my handle is from my real name, the less association I want. This is another reason I'm wary of single sign-on. These are not just names; these are personas. My blog is like me, but on mailing lists, I'm more terse and cynical; on XBox Live, I swear more. You have to give people the chance to use things other than their real names - there are few social applications I can think of (LinkedIn is one) where the real name actually has some currency. This is how the internet has worked for a long while: usenet, email addresses don't *have* to be your real login. Go back further - radio hams, CB enthusiasts – they use handles rather than real names. A degree of anonymity in a public space.

So: you should expect people to use false details. They might not, but assume that as a baseline, and prepare way of working around it. And you should expect people to tell untruths.

I remember the first big online hoax I saw; I started blogging in early 2001. This girl [left] is Kaycee Nicole. She was dying of leukemia, and she kept a blog about it. A lot of people were very sympathetic to her, and gave her many kind words - over the internet, over the phone. In May 2001, she died. And no-one could find an obituary, or get in touch with her family. Through a process of investigation, it turned out to be a hoax; a forty-year old woman writing diary entries, photos of a neighbour. People didn't lose money; they lost faith. They hadn't thought it could be a story, all be fake. Why not? There's no truth/fiction flag on the net; the medium is equally suited to both. But people had assumed - because of the writing, the pictures - that it was real, and given Kaycee their hope. Every well-wisher learnt a lesson that day: truth is not the default.

Do you remember the Fakesters? People creating entries for fictional characters on Friendster; everyone wanted to be friends with Harry Potter and Lord Voldemort. This didn't fit how Friendster wanted the system to be used - they wanted it to be a dating service, not a playground for geeks - so when someone pretended to be Vincent Gallo, they deleted his account.

Except it really *was* Vincent Gallo. Why did it matter? These fake users weren't fooling anyone. And if they did, then why should that matter? Why shouldn't famous people, celebrities, want to play with these services. Friendster has pretty much gone down the pan, but it could have been a lovely place to play.

Of course, with any open system, you have to keep an eye on it, and consider changing the rules. Trolls emerge; liars emerge; look at the issues that have plagued MySpace in the past year, and you can see the problems. But this has always been the case - it's just there are more people online than ever before, and more unwary people. Caution, as ever, is advised.

But fiction is healthy for your product - it encourages creativity, pushes what you can do with it. And if it turns out to be unhealthy, well, you'll know about it. But don't shy away from it by default. We all tell untruths all the time. Who are you to say whether it's a significant untruth or not.

So that's been a bit of a look at the ways storytelling can influence software. Let's round up with a look at how you can encourage that storytelling, what you can control, and what you can't.

5. Telling the story

I've asserted that people are going to use social software to tell stories. How they are encouraged to tell those stories will affect the stories they tell, and how they tell them. When you make strong use of the second person - you, yours - the first person - I, me - will come out in return. So the language your application uses is important.

One other thing that's really important is where the focus lies: on the teller or the tale. I got thinking about this reading Anthony Swofford's excellent *Jarhead* - which I do recommend, incidentally, it's far better and quite different to the Sam Mendes film. Anyhow, there's a sequence in that book where some characters attend a friend's funeral, in his home town, and they meet his friends.

"They recounted combat events that Troy had told them, and we realised by the context of their stories that Troy had made us heroes for his friends because we'd been heroes to him.

At this point I was the saddest I'd yet been over Troy's passing, because the true friend from war is the friend who obliterates his own story by telling the stories of others. Through briefing his hometown friends about the rest of the platoon, Troy had effectively diminshed his own role; he'd become the tale teller and removed himself from the tale."

Aristotle says a similar thing of Homer in his *Poetics*:

"[Homer] alone among the epic poets has not failed to understand the part the poet himself should take in his poem. The poet should, in fact, speak as little as possible in his own person, since in what he himself says he is not an imitator."

So how can we do these in software? How can the focus - be it teller or tale be shifted? One example, that I've been thinking out: what people see when they log in. If you log into Flickr, you see a combination of your pictures, your friend's pictures, and other people's pictures. The focus is partly on you, but partly on others - there's a balance between teller and tale, and that product lets users find out which they prefer - the social side, the "me" side, or the simple publishing, the "telling" side. If you log in and were to go straight to "my pictures", the software is placing the emphasis on you, the teller, and not the stories you tell; by placing the emphasis on other tellers, away from the current user - for instance, by logging into the "all my contacts' pictures" page, you indicate that it's the stories being told, and not the individuals who matter. Consider an online social recipe book, for instance. A real recipe book isn't just "my recipes", it's "my recipes + everything I've cut out or copied". So you want a combination of "me" and "my favourites". The views of data you give users can be more complex than "my things, their things"; think about what combination, what layout, strikes the balance between the teller and the tale you want your software to inspire.

Another metaphorical balance I'd like to explore is that of plot *versus* narrative. You can instill plot in your software - a rough structure, goals, achievements - but narrative will be generated via user experience. Best explained with this quotation from Mark Kermode from the *Observer*, looking at why videogames make terrible films:

"An individual game may be laden with 'plot points' but its narrative is always up for grabs. It is a format of scenarios rather than stories, elements which can be bolted together in differing orders with varying outcomes."

He acknowledges that an adaptive form of narrative might have worked for serial authors like Dickens, but "episodically published tales and constantly evolving stage performances are hardly an adequate paradigm for movies which, like paintings and sculptures, must be first unveiled fully formed."

Well, fortunately, it's a perfectly good paradigm for social software. Narrative is something to be navigated between, manipulated by your users. You can still do a lot with plot - remember, Flickr changed the plot of their software entirely - it started out as a Flash- based collaborative shoebox, and took a while to transform into the HTML-based product we see today.

Do be aware that you can generate plot people don't like, though, or may react badly to. Does anybody remember Breedster? It was social software as insect life - insects wandering around the map, eating, shitting, having casual sex with strangers, breeding eggs, into which you could invite new users. But because it was only intended as a short-lived experiment, about a month or two into its life, the creators introduced a sexually-transmitted disease. It made people sterile. Which, of course, eventually infected the entire population. And they were angry - and desperate for a cure. Because, surely, something like this has to have a cure, right?

No - there was no cure. It was cruel genocide; condemning a race to infertility. It was a statement on the part of the creators - sometimes shit happens - but its user-base, for a few months, genuinely felt attached to their little insects, to this product, and now the plot was heading to a tragic ending. No amount of narrative on their part could save them. It's a drastic example, but do realise that you can't work around every plot change with some snappy user-experience narrative.

Time to wrap up, I guess.

Conclusion

Stories are told in so much more than paragraphs, chapters. When we give users a publishing medium - be it a textbox, a file-upload box, a voice-recorder, or, to be honest, whatever products I can't dream of that are in the future - we should expect them to tell stories.

As a side effect, we should also expect fiction to creep in. We'll notice it long after it began emerging - it begins the first time somebody embellishes a "story" that they're telling to make it better, or joins two stories together into one for pacing, not just the first time someone writes a novel. When that happens, we shouldn't fight it (necessarily), but see how we can work around it, and with it.

If you're working in anything that we could consider a publishing medium, don't just look to software for inspiration; don't get bogged down thinking about how to steal more Web 2.0 interface design. Look back to the stories - novels, poetry, films, plays - that you've enjoyed, that thousands have enjoyed, over many, many years, and consider what they can teach you. And then let your users tell a million more stories.