Paul Buchheit

Thursday, January 22, 2009

Communicating with code

Some people can sell their ideas with a brilliant speech or a slick powerpoint presentation.

I can't.

Maybe that's why I'm skeptical of ideas that are sold via brilliant speeches and slick powerpoints. Or maybe it's because it's too easy to overlook the messy details, or to get caught up in details that seem very important, but aren't. I also get very bored by endless debate.

We did a lot of things wrong during the 2.5 years of pre-launch Gmail development, but one thing we did very right was to always have live code. The first version of Gmail was literally written in a day. It wasn't very impressive -- all I did was take the Google Groups (Usenet search) code (my previous project) and stuff my email into it -- but it was live and people could use it (to search my mail...). From that day until launch, every new feature went live immediately, and most new ideas were implemented as soon as possible. This resulted in a lot of churn -- we re-wrote the frontend about six times and the backend three times by launch -- but it meant that we had direct experience with all of the features. A lot of features seemed like great ideas, until we tried them. Other things seemed like they would be big problems or very confusing, but once they were in we forgot all about the theoretical problems.

The great thing about this process was that I didn't need to sell anyone on my ideas. I would just write the code, release the feature, and watch the response. Usually, everyone (including me) would end up hating whatever it was (especially my ideas), but we always learned something from the experience, and we were able to quickly move on to other ideas.

The most dramatic example of this process was the creation of content targeted ads (now known as "AdSense", or maybe "AdSense for Content"). The idea of targeting our keyword based ads to arbitrary content on the web had been floating around the company for a long time -- it was "obvious". However, it was also "obviously bad". Most people believed that it would require some kind of fancy artificial intelligence to understand the content well enough to target ads, and even if we had that, nobody would click on the ads. I thought they were probably right.

However, we needed a way for Gmail to make money, and Sanjeev Singh kept talking about using relevant ads, even though it was obviously a "bad idea". I remained skeptical, but thought that it might be a fun experiment, so I connected to that ads database (I assure you, random engineers can no longer do this!), copied out all of the ads+keywords, and did a little bit of sorting and filtering with some unix shell commands. I then hacked up the "adult content" classifier that Matt Cutts and I had written for safe-search, linked that into the Gmail prototype, and then loaded the ads data into the classifier. My change to the classifier (which completely broke its original functionality, but this was a separate code branch) changed it from classifying pages as "adult", to classifying them according to which ad was most relevant. The resulting ad was then displayed in a little box on our Gmail prototype ui. The code was rather ugly and hackish, but more importantly, it only took a few hours to write!

I then released the feature on our unsuspecting userbase of about 100 Googlers, and then went home and went to sleep. The response when I returned the next day was not what I would classify as "positive". Someone may have used the word "blasphemous". I liked the ads though -- they were amusing and often relevant. An email from someone looking for their lost sunglasses got an ad for new sunglasses. The lunch menu had an ad for balsamic vinegar.

More importantly, I wasn't the only one who found the ads surprisingly relevant. Suddenly, content targeted ads switched from being a lowest-priority project (unstaffed, will not do) to being a top priority project, an extremely talented team was formed to build the project, and within maybe six months a live beta was launched. Google's content targeted ads are now a big business with billions of dollars in revenue (I think).

Of course none of the code from my prototype ever made it near the real product (thankfully), but that code did something that fancy arguments couldn't do (at least not my fancy arguments), it showed that the idea and product had real potential.

About Me

Paul Buchheit

http://en.wikipedia.org/wiki/Paul_Buchheit
https://twitter.com/paultoo

View my complete profile

Blog Archive

The point of this story, I think, is that you should consider spending less time talking, and more time prototyping, especially if you're not very good at talking or powerpoint. Your code can be a very persuasive argument.

The other point is that it's important to make prototyping new ideas, especially bad ideas, as fast and easy as possible. This can be especially difficult as a product grows. It was easy for me to stuff random broken features into Gmail when there were only about 100 users and they all worked for Google, but it's not so simple when there are 100 million users.

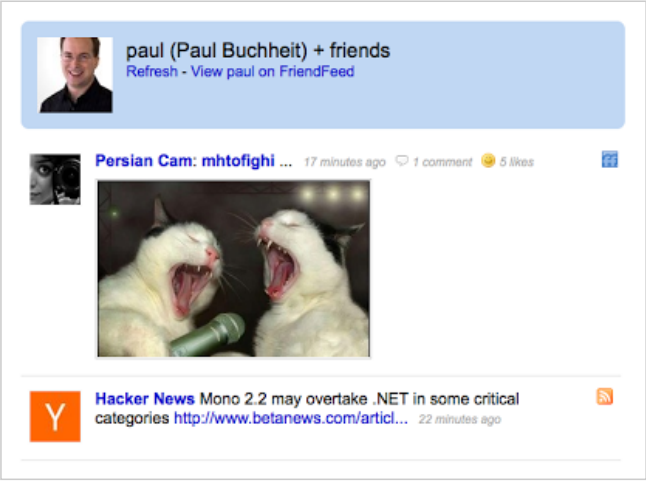
Fortunately for Gmail, they've recently found a rather clever solution that enables the thousands of Google engineers to add new ui features: Gmail Labs. This is also where Google's "20% time" comes in -- if you want innovation, it's critical that people are able to work on ideas that are unapproved and generally thought to be stupid. The real value of "20%" is not the time, but rather the "license" it gives to work on things that "aren't important". (perhaps I should do a post on "20% time" at some point...)

One of the best ways to enable prototyping and innovation on an established product is though an API. Twitter is possibly the best example of how well this can work. There are thousands of different Twitter clients, with new ones being written every day, and I believe a majority of Twitter messages are entered though one of these third-party clients.

Public APIs enable everyone to experiment with new ideas and create new ways of using your product. This is incredibly powerful because no matter how brilliant you and your coworkers are, there are always going to be smarter people outside of your company.

At FriendFeed, we discovered that our API does more than enable great apps, it also reveals great app developers. Gary and Ben were both writing FriendFeed apps using our API before we hired them. When hiring, you don't have to guess which people are "smart and gets things done", you can simply observe it in the wild :)

In my previous post, I asked people to describe their "ideal FriendFeed". Since then, I've been thinking about ideas for my "ideal FriendFeed". Unfortunately, it's very difficult for me to know how much I like an idea based only on words or mockups -- I really need to try it out. So in the spirit of prototyping, I've used my spare time to write a simple FriendFeed interface that prototypes some of the things I've been thinking about. This interface isn't the "future of FriendFeed", it's just a collection of ideas, some that I like, and some that I don't. One thing that's kind of cool about it (from a prototyping perspective) is that it's written entirely in Javascript running in the web browser -- it's just a single web page that uses FriendFeed's JSON APIs to fetch data. This also means that it's relatively easy for other people to copy and change -- you don't even need a server!



If you'd like to try it out, you can see everyone that I'm subscribed to (assuming their feed is public), or if you are a FriendFeed user, you can see all of your public subscriptions by going to **http://paulbuchheit.github.com/xfeed.html#YOUR_NICKNAME_GOES_HERE**. The complete source code (which is just several hundred lines of HTML and JS) is here. In this prototype, I'm experimenting with treating entries, comments, and likes all as simple "messages", only showing comments from the user's friends (which can be a little confusing), and putting it all in reverse-chronological order. As I mentioned, this interface isn't the "future of FriendFeed", it's just a collection of ideas that I'm playing with.

If you're interested in prototyping something, feel free to take this code and have your way with it. As always, I'd love to see your prototypes action!

Posted by Paul Buchheit at 2:36 AM

Newer Post Home Older Post