Apple's Mistake

paulgraham.com · Paul Graham · 2009-11 · source

|

November 2009* * *
I don't think Apple realizes how badly the App Store approval process is broken. Or rather, I don't think they realize how much it matters that it's broken.

The way Apple runs the App Store has harmed their reputation with programmers more than anything else they've ever done.
Their reputation with programmers used to be great.
It used to be the most common complaint you heard about Apple was that their fans admired them too uncritically.
The App Store has changed that. Now a lot of programmers have started to see Apple as evil.

How much of the goodwill Apple once had with programmers have they lost over the App Store? A third? Half? And that's just so far.
The App Store is an ongoing karma leak.

How did Apple get into this mess? Their fundamental problem is that they don't understand software.

They treat iPhone apps the way they treat the music they sell through iTunes. Apple is the channel; they own the user; if you want to reach users, you do it on their terms. The record labels agreed, reluctantly. But this model doesn't work for software. It doesn't work for an intermediary to own the user. The software business learned that in the early 1980s, when companies like VisiCorp showed that although the words "software" and "publisher" fit together, the underlying concepts don't. Software isn't like music or books. It's too complicated for a third party to act as an intermediary

between developer and user. And yet that's what Apple is trying to be with the App Store: a software publisher. And a particularly overreaching one at that, with fussy tastes and a rigidly enforced house style.

If software publishing didn't work in 1980, it works even less now that software development has evolved from a small number of big releases to a constant stream of small ones. But Apple doesn't understand that either. Their model of product development derives from hardware. They work on something till they think it's finished, then they release it. You have to do that with hardware, but because software is so easy to change, its design can benefit from evolution. The standard way to develop applications now is to launch fast and iterate. Which means it's a disaster to have long, random delays each time you release a new version.

Apparently Apple's attitude is that developers should be more careful when they submit a new version to the App Store. They would say that. But powerful as they are, they're not powerful enough to turn back the evolution of technology. Programmers don't use launch-fast-and-iterate out of laziness. They use it because it yields the best results. By obstructing that process, Apple is making them do bad work, and programmers hate that as much as Apple would.

How would Apple like it if when they discovered a serious bug in OS X, instead of releasing a software update immediately, they had to submit their code to an intermediary who sat on it for a month and then rejected it because it contained an icon they didn't like?

By breaking software development, Apple gets the opposite of what they intended: the version of an app currently available in the App Store tends to be an old and buggy one. One developer told me:

> As a result of their process, the App Store is full of half-baked applications. I make a new version almost every day that I release to beta users. The version on the App Store feels old and crappy. I'm sure that a lot of developers feel this way: One emotion is "I'm not really proud about what's in the App Store", and it's combined with the emotion "Really, it's Apple's fault."

Another wrote:

> I believe that they think their approval process helps users by ensuring quality. In reality, bugs like ours get through all the time and then it can take 4-8 weeks to get that bug fix approved, leaving users to think that iPhone apps sometimes just don't work. Worse for Apple, these apps work just fine on other platforms that have immediate approval processes.

Actually I suppose Apple has a third misconception: that all the complaints about App Store approvals are not a serious problem. They must hear developers complaining. But partners and suppliers are always complaining. It would be a bad sign if they weren't; it would mean you were being too easy on them. Meanwhile the iPhone is selling better than ever. So why do they need to fix anything?

They get away with maltreating developers, in the short term, because they make such great hardware. I just bought a new 27" iMac a couple days ago. It's fabulous. The screen's too shiny, and the disk is surprisingly loud, but it's so beautiful that you can't make yourself care.

So I bought it, but I bought it, for the first time, with misgivings. I felt the way I'd feel buying something made in a country with a bad human rights record. That was new. In the past when I bought things from Apple it was an unalloyed pleasure. Oh boy! They make such great stuff. This time it felt like a Faustian bargain. They make such great stuff, but they're such assholes. Do I really want to support this company?

* * *

Should Apple care what people like me think? What difference does it make if they alienate a small minority of their users?

There are a couple reasons they should care. One is that these users are the people they want as employees. If your company seems evil, the best programmers won't work for you. That hurt Microsoft a lot starting in the 90s. Programmers started to feel sheepish about working there. It seemed like selling out. When people from Microsoft were talking to other programmers and they mentioned where they worked, there were a lot of self-deprecating jokes about having gone over to the dark side. But the real problem for Microsoft

wasn't the embarrassment of the people they hired. It was the people they never got. And you know who got them? Google and Apple. If Microsoft was the Empire, they were the Rebel Alliance. And it's largely because they got more of the best people that Google and Apple are doing so much better than Microsoft today.

Why are programmers so fussy about their employers' morals? Partly because they can afford to be. The best programmers can work wherever they want. They don't have to work for a company they have qualms about.

But the other reason programmers are fussy, I think, is that evil begets stupidity. An organization that wins by exercising power starts to lose the ability to win by doing better work. And it's not fun for a smart person to work in a place where the best ideas aren't the ones that win. I think the reason Google embraced "Don't be evil" so eagerly was not so much to impress the outside world as to inoculate themselves against arrogance.
[1]

That has worked for Google so far. They've become more bureaucratic, but otherwise they seem to have held true to their original principles. With Apple that seems less the case. When you look at the famous
1984 ad
now, it's easier to imagine Apple as the
dictator on the screen than the woman with the hammer.
[2]
In fact, if you read the dictator's speech it sounds uncannily like a prophecy of the App Store.

We have triumphed over the unprincipled dissemination of facts.The other reason Apple should care what programmers think of them
is that when you sell a platform, developers make or break you. If anyone should know this, Apple should. VisiCalc made the Apple II.
We have created, for the first time in all history, a garden of pure ideology, where each worker may bloom secure from the pests of contradictory and confusing truths.

And programmers build applications for the platforms they use. Most

applications—most startups, probably—grow out of personal projects. Apple itself did. Apple made microcomputers because that's what Steve Wozniak wanted for himself. He couldn't have afforded a minicomputer.
[3]
Microsoft likewise started out making interpreters for little microcomputers because Bill Gates and Paul Allen were interested in using them. It's a rare startup that doesn't build something the founders use.

The main reason there are so many iPhone apps is that so many programmers have iPhones. They may know, because they read it in an article, that Blackberry has such and such market share. But in practice it's as if RIM didn't exist. If they're going to build something, they want to be able to use it themselves, and that means building an iPhone app.

So programmers continue to develop iPhone apps, even though Apple continues to maltreat them. They're like someone stuck in an abusive relationship. They're so attracted to the iPhone that they can't leave. But they're looking for a way out. One wrote:

> While I did enjoy developing for the iPhone, the control they place on the App Store does not give me the drive to develop applications as I would like. In fact I don't intend to make any more iPhone applications unless absolutely necessary.

[4]
Can anything break this cycle? No device I've seen so far could. Palm and RIM haven't a hope. The only credible contender is Android. But Android is an orphan; Google doesn't really care about it, not the way Apple cares about the iPhone. Apple cares about the iPhone the way Google cares about search.

* * *

Is the future of handheld devices one locked down by Apple? It's a worrying prospect. It would be a bummer to have another grim monoculture like we had in the 1990s. In 1995, writing software for end users was effectively identical with writing Windows applications. Our horror at that prospect was the single biggest thing that drove us to start building web apps.

At least we know now what it would take to break Apple's lock. You'd have to get iPhones out of programmers' hands. If programmers used some other device for mobile web access, they'd start to develop apps for that instead.

How could you make a device programmers liked better than the iPhone? It's unlikely you could make something better designed. Apple leaves no room there. So this alternative device probably couldn't win on general appeal. It would have to win by virtue of some appeal it had to programmers specifically.

One way to appeal to programmers is with software. If you could think of an application programmers had to have, but that would be impossible in the circumscribed world of the iPhone, you could presumably get them to switch.

That would definitely happen if programmers started to use handhelds as development machines—if handhelds displaced laptops the way laptops displaced desktops. You need more control of a development machine than Apple will let you have over an iPhone.

Could anyone make a device that you'd carry around in your pocket like a phone, and yet would also work as a development machine? It's hard to imagine what it would look like. But I've learned never to say never about technology. A phone-sized device that would work as a development machine is no more miraculous by present standards than the iPhone itself would have seemed by the standards of 1995.

My current development machine is a MacBook Air, which I use with an external monitor and keyboard in my office, and by itself when traveling. If there was a version half the size I'd prefer it. That still wouldn't be small enough to carry around everywhere like a phone, but we're within a factor of 4 or so. Surely that gap is bridgeable. In fact, let's make it an
RFS. Wanted:
Woman with hammer.

Notes

[1]
When Google adopted "Don't be evil," they were still so small
that no one would have expected them to be, yet.

[2]
The dictator in the 1984 ad isn't Microsoft, incidentally;
it's IBM. IBM seemed a lot more frightening in those days, but
they were friendlier to developers than Apple is now.

[3]
He couldn't even afford a monitor. That's why the Apple
I used a TV as a monitor.

[4]
Several people I talked to mentioned how much they liked the
iPhone SDK. The problem is not Apple's products but their policies.
Fortunately policies are software; Apple can change them instantly
if they want to. Handy that, isn't it?

Thanks to Sam Altman, Trevor Blackwell, Ross Boucher,
James Bracy, Gabor Cselle,
Patrick Collison, Jason Freedman, John Gruber, Joe Hewitt, Jessica Livingston,
Robert Morris, Teng Siong Ong, Nikhil Pandit, Savraj Singh, and Jared Tame for reading drafts of
this.

|