

High Tech Marketing/Business Model Boot Camp, Part 3: Building a Strong(er) Ecosystem

by Nilofer Merchant, CEO, Rubicon Consulting

In grade school, one of the key determinants of popularity on the playground was how quickly you were selected when the time came to choose up sides for basketball, baseball, or soccer. In the same way, the developing business model for the next 10 years depends hugely on which set of developer and ecosystem partners pick you.

However, unlike grade school, you might have more ability to influence this selection.

Do developers really matter?

Major companies like eBay, Microsoft, Adobe, Nokia, Motorola, Intuit and others have focused on the developer community. Within Web 2.0, the developer world is a critical consideration for technology companies. This applies even if your company isn't doing software or delivery on the Web. Developers have become what the channel was in the 80s: a necessary part of the chain that extends your reach.

Developing technology software and hardware has grown far beyond just coming up with a "widget." When vendors meet customers' needs, it's not a "design it, build it, sell it" loner activity. Today, technology is much more about connecting lots of different technologies and solutions. For example, Zillow is built using Google Maps and a housing database. Yet, Google Maps is built using Telcontar technology and Navteq. None of these individual companies is the entire solution. But the combination of different technologies allows a customer solution (or, in this case, many) to be created.

Thus, no one vendor creates winning solutions alone.

Only a few industry insiders know that Telcontar is the primary technology behind Google Maps. What the broader public is familiar with is Google, because it's Google that formed the developer community to enable a high number of applications to be built on the platform. This case illustrates that success occurs as more value is added to a platform, because the platform then attracts developers, businesses, and consumers eager to develop on it, use it, and transact business with it.

The multiplier effect

Developers are the most amazing creatures on earth. It's through them and with them that any platform becomes interesting. Back when I worked at Apple, I fell in love with a simple screen saver. It was the first of its kind and it made my screen look like a fish tank. Way cool. Besides that coolness effect, developers add to the foundation that a vendor creates, then they multiply it.

Developers design new applications that will change users' lives. And from a vendors' perspective, it's like going from 2000 software engineers to 20,000. It's a multiplier, a relatively inexpensive multiplier.

Through this multiplier, a vendor wins. When eBay creates a developer community, it no longer provides a service for ecommerce. Instead, it provides a proven global platform for ecommerce. By releasing the APIs, and leveraging their user interface, developers enable full integration into many models. When that happens, their company achieves critical mass and the competition must use much more energy to displace it. Sometimes the competition decides the market is too difficult to enter. The company gains strength geometrically as it becomes a platform that 1,000 (or better yet, 100,000) companies support.

Developer strategy must be considered a fundamental company vision. For example, the mobile market has been getting a lot of attention about their developer strategy. Nokia, the predominant leader of mobile phones in Europe, and Motorola, the leader in the U.S., have been investing in new talents and developer programs lately. Why? If Nokia gets more developers on its platform vs. Motorola, then it gets a stronger set of applications that users want, thus creating real market differentiation.

The resulting cascade of *better developers* → *better solutions* → *better value* → *more customers* means a better value proposition to your developers. An upward cycle is built that will cause some form of category domination. Only when a developer-centric model is built into a business model does that company fundamentally add more value, faster results, and more benefits for the customer. This is no minor issue and it's one that has been put on the back-burner and ignored in this industry.

What does it take to build a developer community?

At Apple, I created a developer program for server products. The task at hand was to figure out a differentiation strategy by adding a specific application along with the high-profit servers, find a route to market, and create more value for the customer. What I learned during that project was this: Developers as a whole are some of the most intelligent, creative, and curious folks that exist. And getting them committed is both easy and hard.

They choose platforms for two reasons. One is the coolness factor, and the second is the economic factor. I net that out to "Feed 'em, and Entertain 'em!"

Software engineers need to have a passion about what they develop. They build something because it's their form of invention. By offering them an opportunity to use their creativity—their talent—on your platform, you'll get involvement. That's one reason Apple has always done a good job with its developer community. Apple offered an alternative platform to the Microsoft option, and differentiated itself by being the company that cared about user elegance.

Perhaps, as with other inventors, developers likely didn't take many business and marketing classes. They need to know ways to get to market. They need the vendor they select to build them an economic path for their products. Without a clear route to market, and the opportunity to

make money, their work will be wasted. Any company that wants a developer community needs to think about this as part of their proposition.

Regardless of its shape and size today, your company can benefit from a developer program. Getting in shape for the developer community is more than a 20-push up model. Your overall fitness level needs to rise. Here are some things to work on:

1. Do you have a developer-friendly culture?

Building a great developer program can only happen in a culture where developers matter. This includes getting a solid developer kit of APIs, establishing proper technical support, and providing access to the business development teams. This can't be done independent of the rest of the company; it has to be central in all you do, and the commitment has to be reflected at the highest levels.

Macromedia, prior to getting acquired by Adobe, was known for its Flex platform and the developer community it fostered. Knowing how strategic these developers were to Adobe, one of the first big post-acquisition speeches made by Shantanu Narayen, the president of Adobe, was about the developer community, the staff retention plan, and the planned increased investment. (<http://www.stephencollins.org/adobe-developer-relations-speech-from-adobe-president/>)

Google it, and you can see 1,000 views and commentaries done on the speech itself. Why? Because the developers were agreeing that this was a good sign for Adobe.

2. Can you motivate developers that really matter to you?

Dilbert would probably be the first to say there are two kinds of developers. The hip ones that do things from passion, and the pointy-head folks that make sure resources are invested well. The best kind of developer program is one that enables both to play by keeping the cost of entry low, and supports an economic return that scales. Having more developers play on your platform is the ultimate compliment. Those vendors that can lower their threshold to a nominal amount (to keep non-professionals from taking up valuable time), are the ones that win.

I remember, a few years ago, a vendor of mine wanted to charge \$5,000 for his developer kit. Since developers have lots of choices about which platform they create on, make sure you're not pricing them out of your play. Think through the economics to be long-term focused.

3. Do you have a developer marketing program in place that will help you get solutions to market?

One of the reasons that Palm took a dominant position in mobile platforms was due to its developer strategy. It created an online, one-stop shop to showcase all things Palm. Not only could you get Palm direct software and gadgets, you could search by different categories of entertainment software (Bejeweled, anyone?), Boy Scouts knot tying, daily planner tools, and financial calculators. If you could think of it, you could look it up and buy it right then and there. It created a platform of 10,000 software applications. Having those 10,000 solutions made Palm

cooler than Nokia or Motorola. To this day, neither company has anything that rivals Palm's extensive library of applications. It's a strategic differentiation. There might be 10,000 apps on the Nokia platform but you wouldn't know it.

August, 2006