25 Feb 2008

I Repeat: Do Not Listen to Your Users

Paul Buchheit on listening to users:

> I wrote the first version of Gmail in one day. It was not very impressive. All I did was stuff my own email into the Google Groups (Usenet) indexing engine. I sent it out to a few people for feedback, and they said that it was somewhat useful, but it would be better if it searched over their email instead of mine. That was version two. After I released that people started wanting the ability to respond to email as well. That was version three. That process went on for a couple of years inside of Google before we released to the world.
>
> Startups don't have hundreds of internal users, so it's important to release to the world much sooner. When FriendFeed was semi-released (private beta) in October, the product was only about two months old (and 99.9% written by two people, Bret and Jim). We've made a lot of improvements since then, and the product that we have today is much better than what we would have built had we not launched. The reason? **We have users, and we listen to them, and we see which things work and which don't.**

Listening to users is a tricky thing. Users often don't know what they want, and even if they did, the communication is likely to get garbled somewhere between them and you. By no means should you *ignore* your users, though. Most people will silently and forever walk away if your software or website doesn't meet their needs. The users who care enough to give you feedback deserve your attention and respect. They're essentially taking it upon themselves to design your product. If you don't listen attentively and politely respond to all customer feedback, you're setting yourself up for eventual failure.

It's rude not to listen to your users. So how do we reconcile this with the first rule of usability-- **Don't Listen to Users**?

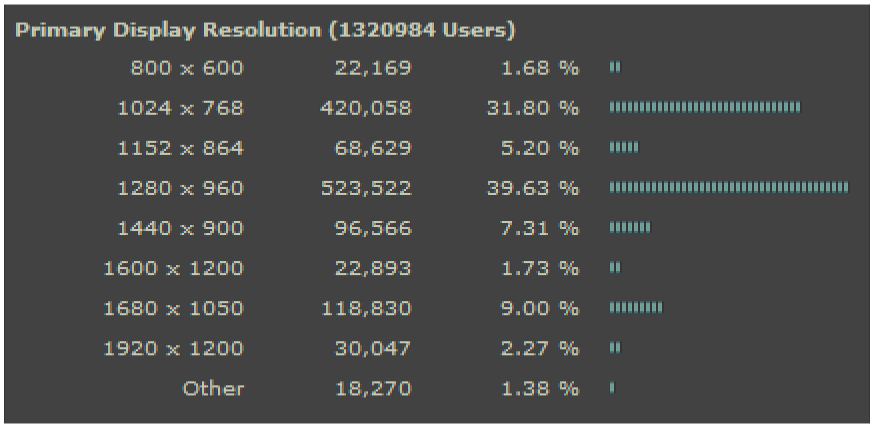
> To discover which designs work best, watch users as they attempt to perform tasks with the user interface. This method is so simple that many people overlook it, assuming that there must be something more to usability testing. [It] boils down to the basic rules of usability:
>
> - Watch what people actually do.
> - Do not believe what people *say* they do.
> - Definitely don't believe what people predict they *may* do in the future.

I think Paul had it right, but it's easy to miss. The relevant phrase in Paul's post is **we see which things work**, which implies measurement and *correlation*. There's no need to directly watch users (although it never hurts) when you have detailed logs showing what they actually did. Collect user feedback, then correlate it with data on what those users are actually doing:

> Don't just implement feature requests from "user representatives" or "business analysts." The most common way to get usability wrong is to **listen to what users say rather than actually watching what they do.** Requirement specifications are always wrong. You must prototype the requirements quickly and show users something concrete to find out what they really need.

Acting on user feedback alone is questionable. No matter how well intentioned, you're guessing. Why guess when you can take actions based on cold, hard data? Acting on user feedback *and* detailed usage metrics for your application or website-- that's the gold standard.

Consider Valve software's hardware survey. A particularly vocal set of gamers might demand support for extremely high widescreen resolutions such as 1920 x 1200 or 2560 x 1600. Understandable, since they've spent a lot of money on high-end gaming rigs. But what resolutions do most people actually play at?



Primary Display Resolution (1320984 Users)

800 x 600	22,169	1.68 %	▮
1024 x 768	420,058	31.80 %	▮
1152 x 864	68,629	5.20 %	▮
1280 x 960	523,522	39.63 %	▮
1440 x 900	96,566	7.31 %	▮
1600 x 1200	22,893	1.73 %	▮
1680 x 1050	118,830	9.00 %	▮
1920 x 1200	30,047	2.27 %	▮
Other	18,270	1.38 %	▮

Based on this survey of 1.3 million Steam users, about 10% of gamers have high resolution, widescreen displays. There are other reasons you might want to satisfy this request, of course. Those 10% tend to be the most dedicated, influential gamers. But having actual data behind your user feedback lets you vet the actions you take, to ensure that you're spending your development budget wisely. The last thing you want to do is fritter away valuable engineering time on features that almost nobody is using, and having usage data is how you tell the difference.

Valve also collects an exhaustive set of gameplay statistics for their games, such as Team Fortress 2.

> We've traditionally relied on things like written feedback from players to help decide which improvements to focus on. More recently, Steam has allowed us to collect more information than was previously possible. TF2 includes a reporting mechanism which tells us details about how people are playing the game. We're sharing the data we

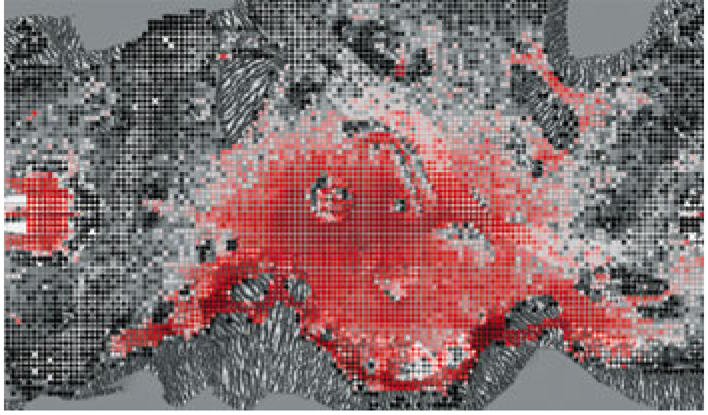
> collect because we think people will find it interesting, and because we expect to spot emergent problems earlier, and ultimately build better products and experiences as a result.

The very first graph, of **time played per class**, illustrates one problem with Team Fortress 2 in a way that I don't think any amount of player feedback ever could.

Scout	17.5%
Engineer	17.3%
Soldier	15%
Demoman	10.5%
Sniper	10.1%
Heavy	8.5%
Spy	8%
Pyro	7%
Medic	5.5%

The medic class is severely underrepresented in actual gameplay. I suppose this is because Medics don't engage in much direct combat, so they're not as exciting to play as, say, a Demoman or Soldier. That's unfortunate, because the healing abilities of the medic class are frequently critical to winning a round. So what did Valve do? They released a giant set of medic-specific achievements to encourage players to choose the Medic class more often. That's iterative game design based on actual, real world gameplay data.

Using detailed gameplay metrics to refine game design isn't new; Bungie ran both Halo 2 and 3 through comprehensive usability lab tests.



> In April, Bungie found a nagging problem with Valhalla, one of Halo 3's multiplayer levels: Player deaths (represented in dark red on this "heat map" of the level) were skewing toward the base on the left, indicating that forces invading from the right had a slight advantage. After reviewing this image, designers tweaked the terrain to give both armies an even chance.

Again-- try to imagine how you'd figure out this fundamental map imbalance based on player feedback. I'm not sure if it's even possible.

Make sure your application or website is capturing user activity in a useful, meaningful way. User feedback is important. Don't get me wrong. But never take action *solely* based on user feedback. Always have some kind of user activity data to corroborate and support the valuable user feedback you're getting. Ignoring your user feedback may be setting yourself up for eventual failure, but blindly acting on every user request is *certain* failure.

NEXT
Douchebaggery

PREVIOUS
On Escalating Communication

Written by Jeff Atwood

Indoor enthusiast. Co-founder of Stack Overflow and Discourse. Disclaimer: I have no idea what I'm talking about. Find me here:

http://twitter.com/codinghorror

Continue Discussion 68 replies

Evan_M Feb '08

Jeff, no new achievements (that I know of) have actually been released for the medic. All that's happened so far is that someone found a bunch of logos names for medic centric achievements.

Caleb Feb '08

Evan, but they *have* announced that achievements will be released. Also, IIRC the article I read stated the medic will be the first class to receive new weapons, which are unlocked when you reach a certain level of medic specific achievements.

Alan Feb '08

I suppose this is because Medics don't engage in much direct combat, so they're not as exciting to play as, say, a Demoman or Soldier.

The other factor amongst gamers is that Medics/Healers are the first to be targeted and killed making them more frustrating to play.

Eikern Feb '08

The other factor amongst gamers is that Medics/Healers are the first to be targeted and killed making them more frustrating to play.
Oh I'm not so sure… often they are hard to get if they are behind a spamming Heavy. And players are often targeting the Heavy, because they don't know better.

Ben Feb '08

That was an awesome post! You put into words what I have been thinking/doing (and not necessarily succeeding at) over the past year.

anthropocentric Feb '08

Henry Ford said something like: "customer's can't envision the future, but they inform the present."

teh_gamer Feb '08

Nobody plays medic because the class got butchered in transition from TFC to TF2.

Schnapple Feb '08

The other factor amongst gamers is that Medics/Healers are the first to be targeted and killed making them more frustrating to play.

Heck, there’s even a T-Shirt on this…

http://www.pennyarcademerch.com/pat070411.html

xaos Feb '08

one of the things I use to help improve the software i write for my school is an error email account. i wrote an error class to send the details of errors to that account. i and the support staff can check that account and use the data to be proactive about fixing bugs and also as objective data to accompany help tickets that may stem from the bugs. we usually find that the source of the problem is users doing things we never expected them to do. it’s an invaluable tool.

madlep Feb '08

Hey Jeff. Cheers for the link back to ubercharged.net. Been following this blog for a while, but mainly from the day job coding side of things. Saw the link coming in when I was checking stats, and did a double take when I saw it was linking to my TF2 blog (my other blog is about coding at www.rawblock.com , but is in a state of a little disrepair at the moment), weird how the interweb works sometimes.

The new medic achievements have been due to be released in the “coming weeks” for a few months now. I guess that’s running on valve time (http://developer.valvesoftware.com/wiki/Valve_Time), so it means we’ll see a release any time in the next year or so 😛

Nicolas Feb '08

Take privacy into account of course.

MagicKat Feb '08

Great post, I agree with a good deal of it. After working in sales for a while, I have learned that people say what they think that they want, but not what they need. You need to qualify responses that you get from users so that you truly understand the needs, instead of just blindly running off to meet their wants.

shooshx Feb '08

A great example of where “listening to the users” leads to very poor usability is the one and only: Microsoft Live.

Cyde_Weys Feb '08

I think you overstate the point that listening to users is a bad idea. In particular, the Team Fortress 2 argument doesn't really support your point of view. I actually play Team Fortress 2, as do some of my friends, and ask any of us why we don't play medic and you'll get the same answer without even having to gather detailed statistics.

Medics are the bitches of the game. They don't get to do much combat, and they get yelled at if they don't heal everyone who needs it, even though the best personal strategy for them to get more points is to stick like glue to one player. They spend all this time charging up their uber, which seems like it should be cool, and then what do they get to do with it? More holding the button down on the medigun, only this time instead of healing, it makes you invulnerable. Because actually letting them participate in combat with a super attack would've been too cool.

Your example on the Halo maps does support your argument. But Team Fortress 2 doesn't.

T **TraumaPony** Feb '08

Although I agree with your post, I want to point out that adding 1920 x 1200 or 2560 x 1600 resolutions is usually nothing more than a menu item, an extra enumeration member, and two extra lines of code.

T **Taufiq** Feb '08

A comment on your example of Valve's hardware survey - it's also important to know how the data was gathered. In this case I believe the Steam application took the specs outside of game time, that is, in the background while you're doing your everyday tasks. In my case my computer (a laptop) is usually connected to a 1900x1200 screen which is fine for coding and desktop use, but for the newest games I usually drop it down to 1024x768. However, this means that I'd be reported as having 1900x1200, which wouldn't be so useful if they're deciding what gaming resolutions to support.

C **Clark** Feb '08

This medic argument brings up an important point. When you have data like this, how do you decide between multiple hypotheses of what the user actually needs?

Clearly, there are at least two competing arguments as to why the medic class is the least played. (This is just my interpretation from the comments here. I don't play TF2.)

1. Balance. The Medic class has fundamental weaknesses that contribute to a low desire for players to play this class that are best addressed by changes to the classes abilities.

2. Learning curve. The Medic is balanced, but is simply difficult to play. The best way to address this is to add

additional achievements to reward skilled and dedicated Medic players to help them get through the learning curve.

3?) Fun factor. The Medic is balanced, but game play elements are not as fun as other classes. People like achievements though, give them those, because re balancing is hard, and costly to get right.

That 5.5% number simply doesn't provide enough information about what is going on. It would be interesting to see what the real process was to arrive at the answer they did. The canonical way to do this would be to gather some new data, designed to test the competing hypotheses, but in my experience this tends to be an "expert's" call. Time is a finite resource, so people tend to make do with the data they have.

Brennan Feb '08

I suspect part of the lack of medics is the results of the third and sixth charts, "Average kills per class per hour" and "Average lifespan per class in minutes" where medics are also at the bottom. I wouldn't play the thing where I get the least kills and die the most, either.

Bob Feb '08

Your users are experts in what problems they have. But not necessarily experts in solving the problems.

"Listen to your users" is still a different concept to "outsource your entire product design department to the most vocal disgruntled user you can find".

AC122 Feb '08

You're conflating two issues. You MUST listen to your users, but you need to understand their needs. They DO know what they want. Sometimes what will make them happy doesn't even make sense, or no matter how wrong they are, sometimes they won't be satisfied until you give them something they don't need. There's just no way around that.

Matt Feb '08

Great Article!

I have stumbled and watched so many others as well who "listen" to their users too much. Designing what the user says they want, rather than really listening, understanding and designing what they really want.

I still struggle from time to time, but life has become so much easier now that I listen correctly.

I think Apple does this well… think about it. The original pre-iPod user wants was something probably close to "I want to listen to my music, let me view any song, any album, sort, shuffle, pause, play and million other things, all with buttons" I can imagine a blackberry style amount of buttons to do that if handled by someone else (and I think I have bought some…), iPods really has only 2 buttons, sure one is more like a touch pad, but it is very very simple, and is easy for people like my parents just to hit "Play", while I am able to (using iTunes) create complex playlists to let me play graded songs from certain genres, periods of music and artists. It does anything I can think of (for the most part) and yet meets the "do not make me think".

Joe_Chung Feb '08

Don't listen to your users, but pay attention to how they use your software.

BoredG Feb '08

It's been a while, but Great post! Waiting for stuff like this to come up is why I keep returning here 🙂

Now this article covers just what I meant back when the UnitTest/UsabilityTest article came out a few weeks ago.

As engineers we can get solid numbers about usability and this will almost always trump some designer's intuition and sense.

This is why I always look at my users using what I make, and why I always use my own stuff for a while before handing it out to others. We can eat out own dog food before having to force my customers to eat it.

hatedance Feb '08

What if we listen to what the users say, observe what they do, and carefully merge them.

Josh59 Feb '08

This post has come at an extremely useful time - so thank you Jeff.

At uni, we've just been given basic design specs for a team project we have to work on over the course of the year for an outside client. As this is all of our first real-world applications that we're going to be developing, it's really helpful to have some ideas behind us that'll help us build a more useful system.

Cheers Jeff.

NurseR Feb '08

I guess it's hard coming up with a title, but the titles suck lately, and as others point out, issues are conflated. Lately, this blog feels like it just doesn't take itself seriously. Pretend we're all doctors or something, would you discuss medical advances and research in the same tone?

(sarcasm)
I Repeat: Don't listen to your patients. Patients never know what's really the problem. They just sit there and complain about a sore bum and headaches and stuff, and never seem to be able to run up their own bloodwork. Man patients suck. (/sarcasm)

Don't JUST listen to your users is more apt.

Michael Feb '08

As usual when universals are made and rejected, reality lies in between.

While it's generally true that users don't know what they want/need, it's also often true that some of them have very insightful ideas about how to improve things.

Also, your most vocal users can also be your most vocal supporters. Simply ignoring their complaints/suggestions may hurt your user base.

One more point, the game design domain may not be as closely related to the web design domain. The designers and users are different in significant ways. (Successful) game designers are a lim

MichaelH Feb '08

As usual when universals are made and rejected, reality lies in between.

While it's generally true that users don't know what they want/need, it's also often true that some of them have very insightful ideas about how to improve things.

Also, your most vocal users can also be your most vocal supporters. Simply ignoring their complaints/suggestions may hurt your user base.

One more point, the game design domain may not be as closely related to the web design domain. The designers and users are different in significant ways. (Successful) game designers are a very limited subset of programmers. They have already proven they can build a popular and usable system – opposed to all the games that fail in development. Web designers and developers are both numerous and have a lower bar to pass. The users tend to be a bit more hard core, perhaps younger, and a bit less "insightful" in their comments than general web users.

That said, your point should is well taken – ethnographic analysis of users (both in the lab and in situ) should always be a major part of application design and revision. "Watching what they do" is a critical of the usability puzzle.

MichaelH Feb '08

Jeez… so much for editing in a text entry box:

That said, your point should be well taken – ethnographic analysis of users (both in the lab and in situ) should always be a major part of application design and revision. "Watching what they do" is a critical piece of the usability puzzle.

walterm Feb '08

The point is to listen to you user's needs, but professionally advise their decisions

titrat Feb '08

Don't believe the steam-resolution data.
Many players lower the resolution before playing Half-Life 2 in order to gain better frame rates. On a notebook with a poorer graphic-card some will even lower to 800x600.
I've seen for years no one who uses 1280x960 - most people use today 1280x1024 (or 1280x800 on a notebook), as webstats show. This fact shows that the Steam-Data is totally wrong.
And don't underestimate marketing: Fulfill the wishes of the high-end, because these are the opinion-leaders, who write in blogs or magazines.

Dr_Bob Feb '08

I just like to point out to NurseRatchett that the whole (sarcasm) (/sarcasm) thing was not needed. We got the point. Listen to me - I'm a user.

ICR45 Feb '08

"I actually play Team Fortress 2, as do some of my friends, and ask any of us why we don't play medic and you'll get the same answer without even having to gather detailed statistics."
That may well be true, but in this instance looking at a 5% figure is much quicker in highlighting the problem than collating many responses, most of which won't be in direct communication with Valve but instead spread around many forums (which, as I understand, they do look at quite a bit). Once you can see a problem, you can ask the direct questions such as "Why aren't you playing as Medic". But if you just asked people who gets played the most, whilst I would have said Medic barely gets played I would also have rated spies much higher. Other people would similarly skew figures based on their own experiences, which class annoys them the most, which they think are most valuable etc.

JoeB Feb '08

The one item that is missed in all this is that I am keeping existing users happy, but many times I need to make changes to attract new users. Having the 5 happiest users in the world won't make you a lot of money unless your product can grow to be useful to many more users (or you charge those 5 users a ton of money). I pay attention to both my current users and also try to address the needs of "future" users.

SomeW Feb '08

Reading your post made me think of what kind of data we are collecting from our users. We record the errors that happen but nothing about what actually works. We kind of assume everything works if they don't complain or the application doesn't record an error.
Instead of an error log maybe have a log that runs once or twice a month that records most used screens, buttons, clicks, keystrokes, etc.
Great post! Thanks.

Kaitain Feb '08

I've found it works best to listen intently to your users, take notes, nod knowingly, promise to do everything the user requests, then go do the software the way you know it needs to be done and later try to convince the user that's what they asked for…

GTA Feb '08

I think the key is to ask whether you have *enough information* about the phenomenon you're studying and whether that information is *reliable* or not.

Any data can be relevant (user feedback and statistics included), but if you don't collect the correct data (or enough of it), any action you take will be based on guesswork and personal experience.

Most poor HCI choices are made by people who

- don't really know what it is they want to know
- asked the wrong questions
- asked too few questions
- have chosen whom to ask inappropriately, or
- think the data they collected has more information than it really does

---

Mike_Hofer Feb '08

In the application I've been working on for the last three years, we implemented two features that turned out to provide "Big Brother" type usability observation. This is a mainline business application, not a game, so it should show how this is feasible in just about any application.

1.) An internal audit trail. Every change that the user makes to any data in the database is recorded in a database table that records the following data: the date time, the original table that held the data, the column name of the field that was changed, the original value, the new value, the page it was changed from, and the user who made the change.

What this did for us was unintentionally reveal *which* features the users were actually using. We found that the most used feature in the application turned out to be one that was added late in the development cycle, as an afterthought at the customers' request. It got a lot of attention in the second release iteration.

It also tells us which users are actually using the system. During testing, it tells us which features actually got tested. For defect resolution, we can see how data was changed, and what data a user was working with when an error occurred.

2.) An internal event log. Any time an exception is thrown, it's recorded in this log. Any time a user logs in, it's recorded in this log. Any time a user uses the Log Off button, it's recorded in this log.

This tells us which users are actually logging into the system, and how many are logging off correctly as opposed to just letting their sessions expire. It also gives us the detailed exception information without having to scrape the Application Event Log in Windows. Because the table includes the ID of the user who was logged in at the time and the date and time it was thrown, we can find relevant information in the Audit Trail at the same time.

Jeff is absolutely right: where errors and usability problems occur, you just can't rely on users to provide accurate feedback. No one is *actively* thinking about those details when you ask them about them. Or, they want to present the best possibly face to the developer (who frequently intimidates them) when they're asked.

Catto Feb '08

Hey Now Jeff,
After reading this it reinforces the importance of UAT (user acceptance testing). When there are testers that look from a users perspective rather than just a quality assurance perspective the produced application is going to be better, even if its something small such as tab order on a login in screen. Nice post.
Coding Horror Fan,
Catto

Nick_Healey Feb '08

Yay, Nurse Ratchett - beautifully put. It's no big secret: Use what data you can get (advisedly); use what user feedback what data you can get (advisedly), use what user feedback you can get (advisedly).

And “advisedly” applies to both. Eg maybe your feedback *does* show that they used one particular feature more than others, but, oh I dunno, maybe the other features were too complex for them to understand, so they abandoned trying to use them?

The best way to get user feedback is not just to log them, or listen to them, or even to watch them use it, but to watch them use it *having asked them to speak aloud their thoughts as they do*. You will learn ten times more about what is wrong with your product this way (and it will stagger you what’s going on in normal users’ heads - that sure ain’t your System Model that they’re working to).

As a designer, not a coder, I have an alternative headline for you:

*** Do Not Listen To Your Techies. ***

🙂 because what techies (and marketers, and…etc) think is a good idea at the UI/feature level usually isn’t, until you have an idea of what is really going on in normal users’ heads. Turn yourself into a half-decent designer by finding out: spend some time doing this *spoken aloud* user test with some normal users (using your early version, or using competing products, or whatever relevant that you can get hold of). It will change your life - and, hopefully, the lives of your future users.

Of course, few self-respecting techies ever do this - you’ll typically just say to yourself “yeah, sure, but I’m not that bad, I think I understand normal people pretty well, and we do get user feedback, so, no problem, continue as before”. Mean time, normal people are having a truly horrible time with technology, and just hanging in there doing whatever they’ve managed to get to work, too scared to change or add anything in case that breaks their current tenuous usage.

Snappy headlines are the order of the day, so I can’t go with “Do Not Listen To Anyone Who Doesn’t Sit Through Some Spoken-Aloud User Tests In Order To Discover The Real World Of Users”. But since 90% of the people who design and affect the UI/features of tech products are indeed techies, and who *don’t* do this, let’s go with it again:

*** Do Not Listen To Your Techies. ***

Daniel Feb '08

Another developer I worked with told me about a story from World War II where the Allies wanted to address the loss rate of planes from bombing missions, which was in excess of 50%. They went to study the planes when they returned to determine where the planes were being most damaged and worked on reinforcing those areas.

However the loss rate did not improve. The problem was that

However the loss rate did not improve. The problem was that the data they had was from the planes that did make it back. Their damage was in the non critical areas to begin with. The data they needed was for the planes that were lost.

The point about data collecting is good, but make sure you have the right or relevant data is key.

BillE Feb '08

I agree 100%. Your examples are gaming based but your points are especially true in my experience with business software.

At my company we have a periodic user/developer meeting to discuss improvements that can be made to our proprietary systems. Without fail about 75% of requested features are never supported by our data. And whenever we are required by the powers that be to add the new features, regardless of what the data suggests, the features are inevitably not used and eventually forgotten about.

Your article is well timed as I have been compiling data the past two days which debunks one such request.

BillE Feb '08

One more comment on this. My comment above is presupposing that good, objective analysis is being used when designing and building systems to begin with and not overemphasizing any one source of info such as data or user input.

That being said, one of the best bits of advice I ever received regarding user feedback during analysis is to ask the user why five times. (The actual number of why's will obviously vary depending on the situation but the basic principal is sound.)

If a user tells you something isn't working properly or that something would be better if it work another way, be sure to dig down far enough to get to the true root of the issue. And this is best accomplished by asking open ended questions, like why. It gets through all the fluff and does not lead the user in one direction or another.

M__Ibrahim Feb '08

Isn't that listening to users anyway? In the examples given, developers weren't literally asking them for feedback or feature requests but they were 'listening' anyway?

Matt_V Feb '08

This whole article basically boils down to two (fairly) well-known quotes:
“Actions speak louder than words” - unknown

"One accurate measurement is worth a thousand expert opinions." - Adm. Grace Hopper

Billkamm Feb '08

I personally love the medic class. If your team has a medic and the other doesn't that usually makes the difference in winning vs. losing.

mastadebata Feb '08

great post! spot on IMHO. my friends and i just launched a website and we're looking for some user feedback. we're also interested in seeing how user will really use our site. we're in a private beta period right now but if you check it out and enter you email address, we'll get you approved usually within an hour. www.createdebate.com is the address. sorry for posting a plug but, seriously, i can really relate to this blog post. thanks.

T_E_D136 Feb '08

I'd mostly agree with this. Generally when I go talk to users, it is to educate myself enough to *become* a user like them. Then I can see what needs doing, what needs streamlining, reorganizing, rearranging, etc.

If you simply listen to them and do whatever they ask, you end up throwing tons of unrealted new features onto whatever their favorite screen or tab happens to be. A user doesn't really know what the possibilites (and difficulties) are for the software, so you can't expect them to design things for you. So you shouldn't let them.

o_s80 Feb '08

Excellent post Jeff. I must admit I'm guilty as charged of believing that user feedback was king of kings. When I did watch a user use one of my applications it was only to mainly see if any bugs would crop their ugly head up in my program. I'm glad that I know better now. With actual DATA to look at AND listening to user feedback I can pick out the more overly critical users from the sane ones. Thanks a million.

Alex Feb '08

One of the trickiest things about managing my personal project (Migratr) online has been learning to filter user input. I have a confirmation "Do you really want to do this?" dialog setup for a fringe case that most people hit by accident instead of intentionally, and one user claimed it was "annoying" and I should just get rid of it. That's the first time I recall ever telling one of my users "no". Not "I'll consider that for future release", just "I'm not going to do that." He wasn't a troll or impolite or anything like that- he actually had a fairly decent argument for it. Which, honestly, made it a little more difficult. It’s always easy to ignore a troll.

Currently, my problem is this (and this would be an interesting follow-up post): The applications you mentioned, TF2, Halo etc, are ONLINE games, run through servers maintained by the developers- So all the data they need to analyze to improve their product already passes through a point they can control. How does someone writing a desktop application get data on how their product is being used, without depending on user feedback? It becomes an issue because even if I were to have my app send a small, anonymous packet of information to the server every time it was used, it would quickly be labelled “spyware” and people would get up in arms over it. Even if I had a checkbox, “send anonymous usage statistics, yada yada yada” in an options screen, I’m afraid it would still carry that spyware stigma.

D

Dan Feb '08

You just pointed out the fundamental flaw with many corporate IT organizations. We rely on business users to create requirements for us to execute. They say add a button so I can do this. But if we had dug a little deeper we would have realized that the action they want the button to do, should be done automatically without the additional UI clutter.

At the end of the day most IT professionals don’t do well at root cause analysis of user problems, because we aren’t great communicators. So it’s easier to do what they say, than to get into a meaningful discussion and solve a real issue.

Unless we get data like you suggest. We like data.

K

kenneth17 Feb '08

Take your stinking paws off my software you damn dirty user!!

hehe.

M

MikeW Feb '08

In an environment where you have direct contact with the user, the skill lies in figuring out what the real problem s/he is describing. The developer needs to have sufficient understanding of the domain for the user to be able to describe what pain they’re suffering so that the developer can offer possible cures based on their expertise.

If a colleague tells me they need a report, I’m usually off into Five Whys to establish what they are trying to achieve. The solution is seldom what they asked for in the first place and is usually more satisfying to all of us.

Once they’re well-trained, of course, asking for something as specific a report may very mean that their problem actually is that they don’t have the report. Perhaps they need to deliver it to a regulatory body, for example.

Erick5 Feb '08

Using TF2 as the example, part of the statistics analysis also means deciding what, if anything, needs fixing. In order to do that, there has to be an expectation of what distribution is acceptable.

Spy and Sniper are niche classes, so you would expect their usage to be lower than the Soldier, a solid all-purpose character. The question, then, is how far do you let it slip out of balance before you intervene? Is there even a reason to intervene, or were your original expectations too idealistic or failed to consider some other state of equilibrium?

Would the Pyro be next on the list after the Medic gets some love just because he is underused, or is he fine the way he is? Is it the players' fault that he is underused because they aren't using him properly? Maybe the maps are too open and don't take advantage of the Pyro's love of ambush.

Are the Scout and the Engineer the top two classes because they are too powerful, easier to play, more fun, or more necessary?

There's a lot more to consider than just raw percentages.

Matt Feb '08

Awesome post Jeff, spot on.

Tom Feb '08

Back in the dark ages (middle 90's), we rolled out a new version of our application to a group of production users who had been designated as testers. We had asked them to keep some sort of log about issues they encountered.

After spending a couple of days talking to the users and getting no useful feedback, I implemented something that was radical back then; I added a line to the network Login script that wrote to a text file every time the user logged in. Each