

Saturday, September 29, 2007

## How a proposal process could work - if retaining contributors is a goal

Over the last year in open source PHP I've spent time wheeling through various projects making proposals and it's apparent that some I work really well with (some people are a genuine joy even), and others just end up getting me worked up to the point of ranting. I rarely rant - it's unconstructive, often makes you look like an ass, but given enough motivation I'll do it anyway to vent a little.

The problem with ranting from a Project POV, is that it's coming from the one person a proposal process is supposed to motivate: the proposer. When a proposer turns up, you are lucky to have found someone willing to give freely of their time. To write code and documentation. To shepherd or advocate some idea which (hopefully) would be useful to a project's users. So you really don't want to give these folk the wrong message that they are wasting their time, or contributing to so much empty air, or will be beaten into submission by everyone else with an IDE and a PHP Manual.

But I've ranted twice this Summer already. So let's go simple.

What features would Paddy's preferred style of proposal process have?

1. A proposal is not judged on how much code it contains unless code is the basis of its acceptance


Because it's a proposal - for code that WILL be written. Judge on its merits - it's goals, objectives and ultimate utility. Don't put code on a pedestal. Put goals on a pedestal. If code IS required - then for God's sake say so up front instead of wasting my time. Otherwise the code is likely some tracer implementation for illustrative purposes.

Here is the last comment I made about such code in one proposal:

"Please be aware the svn code is only intended as an experimental example and it has known issues which may require hand editing. It has served its purpose to demonstrate what actual production code could be implemented."


2. A proposal is not an invitation for others to implement its content

My number one frustration is having others dissect, implement, re-implement, and spend unbelievable quantities of email/IM space debating meaningless, implementation points. I do NOT care whether the "example" (i.e. cobbled together in 1 hour) code used `array()` or `ArrayAccess`. I propose, I advocate and if lucky I will later implement - with whatever development methodology I prefer - probably while thankfully deleting the cobbled up tracer code.

To an agile developer this sort of stuff is about as bad as it gets. NO TDD, no BDD, no set defined goals that have been discussed, no timeline other than a version number - and everyone with a pastebin full of code I'll be judged against. This while at the same time witnessing serious talented people take that example code and using it in production because it's so incredibly useful. And it works - mostly 

3. A proposal sets goals - so take a hint and debate the goals.

Implementation is something a monkey can do, if you set the right goals. I may propose we do A, and you want B. So fine, explain B and see if I'll compromise and adopt it. When the goals are finalised (and minimised to their core) I can assemble a picture of how the eventual code will behave.

4. A proposal does not lead to the belief nobody trusts you (well, unless it's deserved 

)

It's sad, but after everyone has implemented their pet vision of your proposal, and compared it to your cobbled together code which has scarce test coverage, and then detailed why yours sucks - well, you start to wonder why. Do they think you're an idiot? Do they not trust you to go away and implement it yourself?

It's not even funny when the proposal is accepted after everyone else has implemented it.

#### 5. A proposal should leverage off the proposer's strengths

Some people know a topic, and know it insanely well. So when they propose a new library or class which draws on that experience a project (if open to that idea) should be pretty happy. Now re-read points 1-4.

That's not an environment I ever want to spend time in.

#### 6. Stealing a proposer's thunder, alienating them, and ignoring their work without real cause is just plain wrong

Irishman makes proposal X. Adds Y some time later. Announces Y to a public mailing list for comment. Project informs Irishman on the same public list, to the same subject, that they are about to propose something like Y. They are going to announce that same week. Irishman predicatable goes a bit crazy - being Irish. Irishman figures out in time this new Y that nobody else knows about took 3 days to cobble together. Irishman gets slightly more crazy. Irishman goes to PEAR and has actual fun. Project figures out that they are missing X (secret component to make Y useful). Irishman eventually re-proposes X and Z but not his Y.

Irishman gets invited to Y's Foundation in Europe as a Member-Subscriber advocating OpenID in Europe. Irishman finally sees the funny side. It gets funnier still later on.

#### 7. A proposal should be granted a timeline for review and acceptance.

This way the Proposer isn't on vacation when it's supposed to be reviewed.

#### 8. A proposal should have an informed motivated proposer

Not one so out of the loop they decide they're clueless about how the proposal process actually works outside it's documented workflow (or if the workflow just plain makes no sense).

#### 9. A proposal process should support (or allow for) best practice in development

Not quite literally work in such a way that the only option is to take anything written by Martin Fowler, Kent Beck or Ward Cunningham and toss it out the nearest window. That not only sucks, it basically tells anyone who actually likes those guys that you don't want them.

I really like those guys. Why don't you?

#### 10. Acceptance notices will not hamstring the proposer's core methodology until they have sufficient time to review the notice and respond to it. I don't care how long the vacation was.

This ends the mighty list of proposal process wishes. Way better than my earlier rant!



Posted by Pádraic Brady in PHP General, PHP Security at 17:57

Completely unrelated, I'm sure, but does Zend Framework have an OpenID consumer? Anonymous on Sep 28 2007, 23:16

Hi Chris,

Dmitry Stogov is working on one in the Incubator - the only holdup is for me to commit the ancillary components like Zend\_Crypt and Zend\_Service\_Yadis which push compatibility with OpenID 2.0 even further - I got Incubator access last week so that should happen inside a few days. After that there are a few Draft 12 changes introduced in late August to integrate, followed by a code review to figure out which edge cases need to be addressed (like moving HTML parsing to DOM and allowing for dual OpenID 1.1/2.0 delegation data points in the parsed HTML link tag).

I'm working on an acceptance test suite for PEAR's OpenID package (only thing holding up the proposal for comment) which I will propose porting to the Zend component if it might be useful in ensuring Spec compliance frequently.

Work, work, work



. Anonymous on Sep 28 2007, 23:40

It's sad to read. For us your proposals are absolutely essential. It's not about that Zend Enhanced View is a nice enhancement to ZF. For us "Zend Enhanced View" IS Zend Framework. ZF MVC just doesn't work for us without it. (And yes the code is not the nicest I have ever seen, but I don't care. I just fixed and tested it and hope a better, final version will be in ZF 1.1)

Even more sad is that Zend could be very happy to have such talented people in there community and, in my view, they should make everything that's possible to support this community (no matter how long vacations are or how much else work there's around.)

They should really make the community process faster and more agail. There's absolutely no excuse to not have all this good proposals not in ZF 1.1. Anonymous on Sep 29 2007, 10:54

Excellent read, Paddy. I love #9



Anonymous on Sep 29 2007, 13:43

Hi Paddy et al, we're currently putting together a proposal



that I hope you will find dramatically improves our proposal process and gets the community more involved. We (the ZF team here at Zend) haven't finished a final draft for review yet, but only because we couldn't afford to distract ourselves from the 1.5 effort right now, and we wanted to have the community's full attention for what we hope will be a lively discussion.



BTW, I'm a big fan of agile methodologies/practices, TDD, and 'agile' requirements gathering practices (the one area that IMO has been surprisingly overlooked in most agile circles/methodologies). I've established Scrum in a few organizations, which- if you've ever had the pleasure- takes a remarkable amount of fastidiousness, patience, and raw faith in the process to convince and change the habits of those developers who are unfamiliar with the concepts. We can have a longer discussion about why I prefer Scrum to XP for almost all organizations/projects, but I can boil it down to 'agile methodologies should improve group productivity/good developers know how to manage their personal productivity'. And, as a rule, I don't waste my time working with bad developers.




If I had to explain why we haven't yet put all of these agile best practices in to place at Zend, I'd say it's because AFAIK, no one else on the team has significant experience in a truly agile environment, and I simply don't understand enough about agile methodologies as applied to OS projects to bring the community along in any significant way at this point. Honestly, I'm just happy that I've gotten people to stop talking about ZF promoting and using 'agile methodologies' when it clearly doesn't (yet). If any of you guys have experience in or suggestions for the implementing an agile methodology in an OS project, please send them my way!

,Wil Anonymous on Feb 6 2008, 03:44

Oddly, I find it comforting someone can use the word Scrum



. I am far more of an XP follower - but then I'm used to working in an existing tight knit team of about 5 members at most. Nature of my development experience. I've been applying XP across a few projects in OS - in fact you probably could pick the ones where I didn't. They tend to be the ones sitting in limbo for far too long (on my

side - Dmitry did review those eventually). The ones with XP advance far more rapidly - usually the main slowdown there is that communication is often via email across two timezones 

. Anonymous on Feb 6 2008, 16:47