

LESSONS LEARNED FROM OSCON

AND HOW THEY APPLY TO THE POWER TOYS

PREAMBLE

A very big thank you to [James Howison](#) for his [awesome OSCON presentation](#) on OSS Communities.

James Howison is a doctoral student and member of Kevin Crowston's NSF-funded research team at the Syracuse University Information School.

All factual data unrelated to the power toys or developers solutions team, unless otherwise noted, comes directly from James Howison's OSCON presentation.¹

OPEN SOURCE SOFTWARE OVERVIEW

BASIC DEFINITION

At a very high-level, OSS is software that meets the following requirements:

1. Source code is provided free of charge
2. The license it is released under allows users to view, modify, and redistribute it.

Redistribution means having the rights to offer your modifications to others (either free or for money). For example, assume you can purchase notepad from MSFT under a license that doesn't allow for redistribution. You buy notepad and modify notepad (include new fonts, give it some UI themes). Now your buddy wants to buy or get a copy of your super-snazzy notepad app. Under the terms of the license agreement, you wouldn't be able to sell or give (i.e redistribute) notepad.

Note that we're not going to debate the differences between [Open Source](#) and [Shared Source](#) in this paper.

WHY WOULD BUSINESSES EVER RUN OSS PROJECTS

Let's get the most popular question out of the way first.

BENEFITS

¹ [Businesses Partnering with Open Source Communities: Opportunities, Perils, and Pitfalls - James Howison](#)

It's all about the access:

1. Access to developers – the company is getting motivated people with the required skill sets and experience that may be not be available in house. There's the saying "[many hands make light work](#)", so by having more and more developers, all aspects of the project are touched / worked upon (and not just the revenue-driving feature set).
2. Access to user support community – a user in the community who is actively learning from an issue and provides information back to the community is much more valuable than a signal PSS call (although you can never write off PSS completely)

MAKING MONEY

But still, how does a OSS company make money when they are giving their intellectual property (i.e. source code) away for free?

1. *OSS*
2. *???*
3. *Profit*

Well, let's start off with, "why does a business ever give anything away for free, like buy 1 get 1 free? Because they are selling something complimentary." One example I've seen is hot dog buns that are on sale for very cheap, maybe for 50 cents, that's too good to pass up. So, you grab a bag and walk over to the very expensive hot dogs...

Three primary ways OSS businesses make money:

1. Platform Play² – make the platform or base open source, but sell add-ins and other extras.
2. Customer support services – when customers need a dedicated channel for getting technical support. Suppose a customer needs a solution within the week. Since there's no way to guarantee the user support community will be able to deliver, the company may provide a PSS-like service.
3. Services. For example, a clothing company may hire a OSS company to make a website for them, since they don't want to get into the software business. Since the software company's bread and butter isn't in the software they are writing, but in the services they are providing, using OSS is a good fit.

BEFORE GOING OPEN, CONSIDER

You must know why you want to go open

The decision to go open must be action-oriented. It can't be based on just, "hey, it's the new software fashion fad, like scrum!" or "this is how we're going to beat our proprietary software competitors."

Good OSS projects allow for co-existence and are very modular. How extensible is your software?

"Modularity or Death!" was the battle-cry in another OSCON tutorials.²

Will you be able to get honored alumni on your side? The more people you can get in the community to support your decision to go open, the easier your transition will be. So the question to us is, "How do we utilize our MVP program?"

² [The Best and Worst of Open Source Business Tactics - Cliff Schmidt](#)

Management can never force developers open. If your developers are not bought into the idea of open source, your OSS project will absolutely fail. Based on James Howison's research, the number one most difficult thing in going open is to make the cultural change within your own development team.

WHY WOULD INDIVIDUALS EVER RUN AN OSS PROJECT

Why would anyone ever take over an OSS project? This is a question we've been asking ourselves for a while now. James Howison gave us an excellent metaphor that I've modified to drive it home for our team:

Josh plays in an adult baseball league and is a general manager for the Diablo's baseball team. Josh doesn't get paid for the work (in fact, he has to pay a club fee!), and maybe he'll get a little recognition out of it by his baseball peers. But why is he really running the baseball team and playing on 2 different teams? Because he is so passionate about baseball that he would go crazy if he saw others playing baseball and didn't jump in.

The same thing applies to OSS. It boils down to one thing: **passion**

OSS PROJECT LIFECYCLE

THE CATHEDRAL AND THE BAZAAR

The OSS development model is broken down into two phases, [the Cathedral and the Bazaar](#), as coined by [Eric Raymond](#). Not knowing who Eric Raymond is at an OSS event is like graduating with a CS degree and not knowing who [Alan Turing](#) is.

THE CATHEDRAL

All of the internal work is done in this phase. The analogy is a team building a cathedral. I'm not sure exactly what the cathedral represents, but here are my two guesses:

The cathedral project owner hires an team to implement the design. Note that a project owner wouldn't hire two competitive companies to do the same work (thinking they would be done in half the time), so the one team works exclusively on the project. And the blueprints probably wouldn't be in the public domain either.

Or, the cathedral represents a team in a proprietary software company working in isolation (hence, working within the confines of the cathedral behind closed doors.)

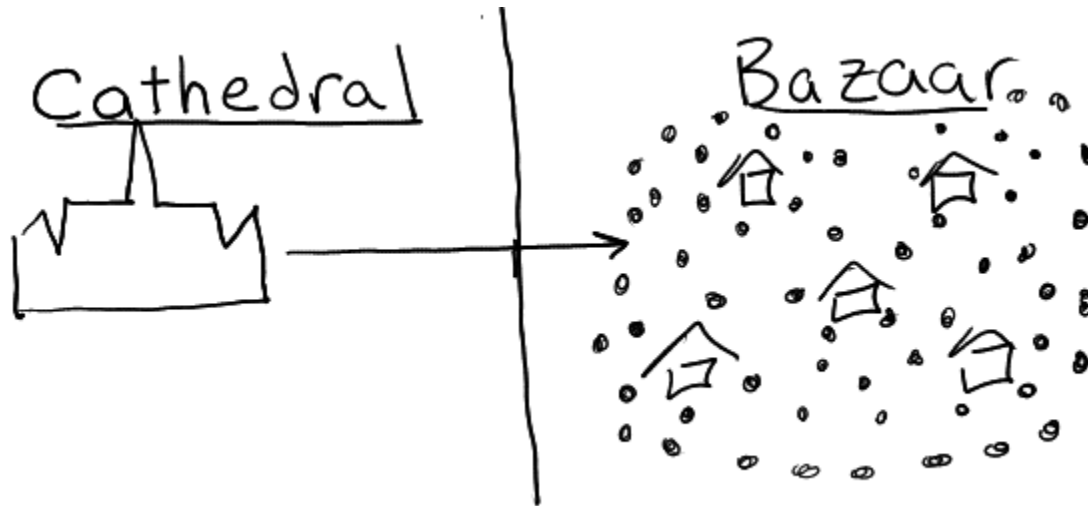
THE BAZAAR

First time I heard this word, I thought James Howison said, "The Bizarre," and I thought, "oh, wow, this is going to really be something." But no, he's referring to an actual bazaar, like the Redmond Sunday Market at Redmond Town Center.

The idea here is that you'll have a theme, like a "Sunday Farmer's Market", where vendors will come in and sell their produce. As time goes on, you'll begin to establish "regulars", both vendors and customers that come to shop.

This analogy holds true in the OSS world. The vendors are the devs from the community who contribute code to the project and your customers are your active end-users who come to leave bug reports, feature requests, and

especially download your tool. And just like in the Farmer's Market example, you'll establish regulars, both community developers and active users that will form the core of your community.



MAKING THE TRANSITION

When transitioning to the bazaar, the project must show **plausible promise**. We don't want it too polished or too finished, or there's nothing for the "bazaar vendors" to setup a market around. But we don't want it so buggy either that it is frustrating, doesn't build, or doesn't show its potential. No one is going to come to a buggy Farmer's Market, pardon the horrible pun.

One of the most critical aspects of changing the cultural mindset on your team is accepting the fact that **once you go open, you stay open**. There is no more of this us versus them, internal versus external distinctions. **All meetings, conversations, decisions must be conducted in the open**. If you fail to do this, the community will feel they are just being used.

TRICKS TO GOING OPEN

Below are some primary indicators whether your OSS project is off to a great start. Let's see how well we're doing here:

Anonymous CVS access – anyone has access to the source code. Well, as I've stated before in previous blog posts, we could have a better model for source control access on CodePlex; however, anonymous users do have access to the source code

Discussion Forums – now do you see why we had to take the hit of transferring our forums to the CodePlex forums.

Build instructions – the statistics are alarming how many OSS projects don't have simple build instructions. The goal here is to make it as click-once as possible to build. I think we're off on the right track with our wiki build guidelines and the conversations we're having about "simple builds", although we need to move these hallway conversations into the "open".

WHAT A OSS COMMUNITY LOOKS LIKE

Now that we have a drawing of what an OSS development model looks like, what sorts of users are represented in these dots?

ROLES

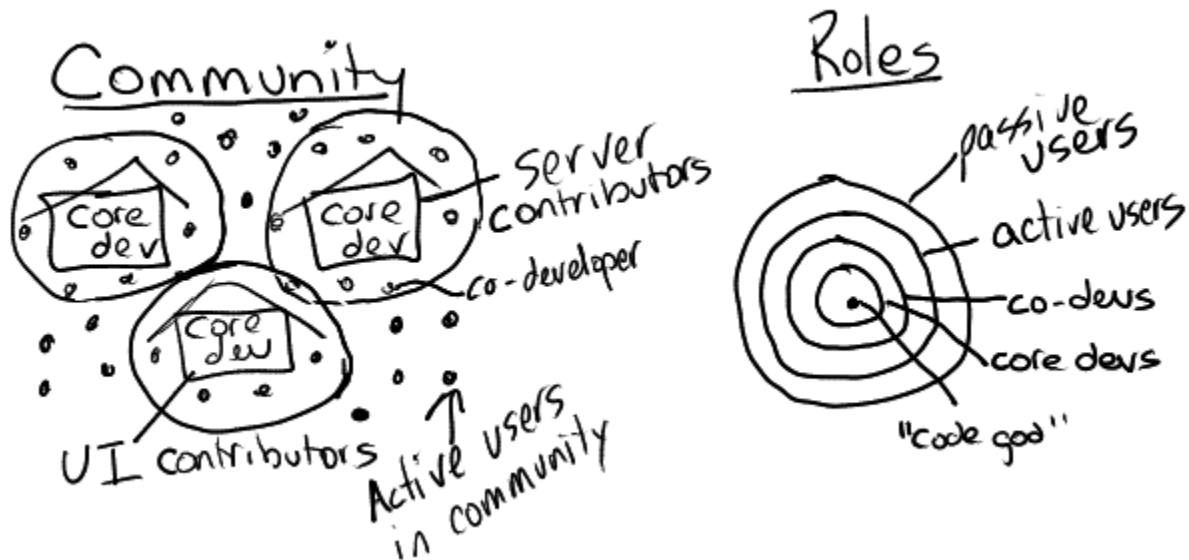
Passive users – just download the code

Active users – ask questions, report bugs, speak their minds (do not provide code gifts)

Co-Developer – submits a patch or code gift, but doesn't have access (or full access) to source. This is the group that also answers questions on the forums, etc.

Core Developer – full access to source code, fixes bugs and adds features, reviews and approves submissions

Benevolent Dictator / Code god - person who owns the project, drives the vision, rejects code gifts, calls the shots, makes the rules.



SIZE

99% of OSS projects have more than 9 active members. The goal is to have 5 active members, which encompasses all roles.

CONVERSATIONS

We have conversations around forums "How do I" and bug reports "am I using the feature correctly," "can anyone else repro this," and "how many other people think we should fix this and why?" Now the question is, "Can we have conversations around the source code with our co-developers and core developers?"

ACTIVE USERS VERSUS CO-DEVELOPERS

The role of the active user is to provide great bug reports, because with so many people running the tool, we'll get excellent coverage across a wide array of edge cases. However, we shouldn't be alarmed to see a high turnover rate here (kinda like employment at a fast food restaurant). I'm guessing similarly to the PFC and Forums, many users will just post just once, get the info they came for, and leave. Thus, we need to **focus our energy into the co-developers**. The more energy we put into them, the more cycles our "core team" will have to move onto other power toy projects.

PARTICIPATION DRIVERS

So, it makes sense why active users exist. They have a huge ROI for simply asking a question, but what about the co-developers who are actually giving us code-gifts? And especially those who we elect as core developers?

The results will surprise you.

As James Howison stated during his talk, if there was only one thing we took away from the 4 hours with him, it was this: **Developers participate in OSS projects for highly diverse reasons**.

Top reasons from James Howison's surveys:

"Scratching an itch" / the high you get off of writing code (think of Josh and Baseball)

Peer recognition and social connections (why do people become MVPs?)

No politics or budget restraints – you have freedom to innovate! (probably why a lot of us are on this team)

Educational benefits (some love to learn; others love to teach)

But the main thing to take away here is that the underlining driving factor is **CREATIVITY**.

TO COMPENSATE OR NOT TO COMPENSATE

THE PAYMENT PLAN – BOO!

OSS projects usually cannot get away with paying their developers; otherwise it looks like they are trying to use their community for cheap labor.

Bounties won't work either. When a OSS project offers an award for bug fixes, the act of contribution becomes a competition and will shut down the conversation we're trying to foster.

THANK YOU GIFTS – YEAH!

Unlike payment, gifts are different sorts of creatures. Are there ways we can *thank* members of our community?

Some examples are

Is there hardware we can provide our users with?

Could we schedule a 2 day coding festival where we fly members of our core development team to campus to provide those extra features the community has been waiting for?

Swag isn't that useful. Without looking at your pen, can you name the company written on it? ³

LOVE AND FEAR AND UNDERSTANDING WHEN GOING OPEN

TO LOVE

Strive for

Discussions that revolve around action. It is a sign our community is doing well. If the discussion ends in process decisions, we may be at risk at getting nowhere.

Constant discussions and constant activity. A constant conversation is a healthy conversation. Don't freak out if we see a lot of buzz at the beginning then nothing for a while. The community may still be trying to find balance between work life and our OSS project life.

Clearly define the members of your core development team, so they can get respect and kudos from the community.

TO FEAR

Avoid

Slow release cycles – as stated earlier, OSS frees us from traditional software development methodologies and gives us the freedom to innovate, including releasing smaller releases more frequently.

Infrastructure issues are the primary cause for burnout in OSS. This is something we'll have to keep an active eye on and if we see signs that our community is burning out because of it, we need to have some serious discussions with the CodePlex team.

A tired benevolent dictator. Apparently, the strongest OSS projects have shown to have switched project leaders 1-2 times over their lifecycle.

Slow turnaround times to security fixes. Cannot happen. Period. QED.

TO UNDERSTAND

All OSS projects will wind-down. However, some projects will wind-down more gracefully than others. It is a question of how gracefully will our projects reach end-of-life.

CALL TO ACTION

7 key takeaways for our power toys...

TRANSITION TO BAZAAR

³ [How to market to people who hate marketing – Doc Searls](#)

It is better to "release" lower-quality code that allows people to jump in (it is a risk), than it is to release high-quality code with tons of downloads with no bug fixes (no one will contribute). We need to define what this middle ground looks like.

Proposal

I propose we "go open" at Beta timeframe. Then we can release a 1.0 version with the community's support. Lastly, we can strive for a 1.1 or a 2.0 release with a new project owner at the helm.

ROADMAPS ARE NOT EFFECTIVE

When OSS projects provide timelines to their releases, the owners are basically saying to the community, "someone somewhere will meet these deadlines, so you all can just sit back and wait for the release to come." Why should they help out when they know it will arrive soon enough?

Proposal

Although we've discussed vision docs for our projects, what we need to do is keep a floating "todo" list (aka our Work Item Tracker in CodePlex) around and display a variety of feature areas to work in and difficult levels (aka having our own version of KDE's Junior Jobs – remember many people are here for educational benefits).

DEFINING THE CORE DEVELOPMENT TEAM

Currently, whenever we receive our contributor agreements, we're granting users access to the CodePlex repository. However, this blurs our lines between co-developers (those who submit contributions via the Work Item Tracker) and core developers (those who have full access to the source code repository).

Proposal

Need to reinforce the model that only core development team members have full source code repository access.

PARTNER WITH AN OSS COMMUNITY

James Howison suggested to me during the tutorial break that we might consider partnering with an OSS project to learn the tricks of the trade, given the background we're coming from. Also, there's the school of thought that "one should seek to hear before seeking to be heard."

Proposal

Does it make sense to have someone on our team join a non-MSFT OSS project?

ONLINE OSS EVALUATION TOOLS

From their [website](#),

Business Readiness Rating™ (BRR) is being proposed as a new standard model for rating open source software. It is intended to enable the entire community (enterprise adopters and developers) to rate software in an open and standardized way.

Proposal

We should take the [Business Readiness Rating](#) survey to evaluate how well we're doing.

This might be a OSS community partnering opportunity, considering their homepage clearly states they are looking for feedback and people interested in helping improve the site.

EVENTS / THANK YOU GIFTS

The marketing strategy I came up with, including the power toy action figure, is wrong – well, probably it's just too early right now. I thought I needed incentives to attract the community, but now I realize that it is the type of projects that is the only incentive that counts.

What we need to do first is establish the community, not via incentives, but through providing code that they want to embrace and work on – to scratch that itch. Then can we provide thank you gifts.

But if they can have a [SQL Server Gal](#), I can have a Power Toy Action Figure.

Proposal

Focus on events and thank you gifts

- Fly people to Microsoft, give them a tour, have a 2-day power toy coding event
- Give people in the community tickets to PDC

TERMINOLOGY CHANGES

During the break, James Howison told me not to use the word “ship” to refer to releasing our projects. The terms “Ship” and “Microsoft” imply certain things, like code quality, proprietary software, do not redistribute, and so forth.

Proposal

Let's remove “ship” from our vocabulary and replace it with “release.”