Lessons Learned

From Running
an Open Source Project
for more than a decade.

Dirk Deimeke

May, 23rd 2017

Taskwarrior Academy @ LinuxERFA

Prolog

Taskwarrior – taskwarrior.org

Taskwarrior Philosophy

- Openness
- Low Friction
- No Penalty
- Methodology Agnostic
- Toolkit
- Extension Friendly
- Community
- Focus

taskwarrior.org/docs/philosophy.html

Lessons learned

What Have We Learned From This Open Source Project?

Here is the collected *wisdom* that we have gained from running the Taskwarrior project for more than a decade.

It has been **rewarding**, **enjoyable**, and sometimes **frustrating**.

We learned a lot about users and Open Source expectations.

I will speak about our experiences
and would like to hear yours.

Start an open source project if you want to learn all you can about software design, development, planning, testing, documenting, and delivery; **enjoy technical challenges**, administrative challenges, compromise, and will **be satisfied hoping that someone out there is benefitting from your work**.

Do **not** start an open source project if you need praise, warmth and love from your fellow human beings.

If you could draw a boundary between that which is already supported, and that which is not, you would find that all the activity, discussion and drama occurs at that boundary.

Feature requests only nibble at the periphery.

Bold changes originate elsewhere.

People will get excited about something a project doesn’t yet support.

Deliver it, and they will get excited about the next thing.

If a feature works well, you’ll never hear about it again.

There is a fine line between *richly-featured* and *bloated*.

There may not be a line at all.

If you demo two features, and talk about twenty more, users still only know about the two.

Visual demonstrations have far greater impact.

Every change will ruin someone’s day.

They will be sure to tell you about it.

The same change will improve someone’s day.

You will not hear of this.

People will disguise feature requests as bugs,
which means either they consider difference of opinion a defect,
or believe that calling it a flaw will force implementation,
but hopefully they just forgot to set the issue type to *enhancement*.

Some people find it very difficult to articulate what they want.

It’s worth being patient and finding out what they need.

What you keep out of a project
is just as important as what you allow in to a project.

Many new users will submit feature requests,
just to show that they are knowledgeable and clever.

They don’t really want that feature, it’s a form of positive feedback.

Beware of suggestions from users
who have used your software for only a day or so.

Be equally aware of suggestions from users
who have used your software for a long, long time.

People will threaten to not use open source software because it lacks a feature, thereby mistaking themselves for paying customers.

Many believe that if a change is small, it deserves to be in the project, regardless of whether it makes sense for it to be there.

Users will go to the effort of seeking you out online,
to directly ask you a question
that is answered two clicks from the front page of a website.

A looping, animated GIF will be watched over and over, scrutinized and understood.

A paragraph of text will be ignored.

Man pages are too densely crammed with information,
and too lengthy, for most modern humans to ingest.

The best question to identify time wasters:

What have you tried so far?

People will pick a fight with you about all your incidental choices.

Your issue tracker,
your branching strategy,
your version numbers,
the text editor you use,
and so on.

You can choose the most permissive software license,
and people will still argue with you about your choice.

SEO consultants are not very good at searching the web,
and learning that you operate an open source, non-profit project.

It says a lot.

No one has ever complained about an algorithm choice,
code structure, or code comments,
but dozens have told us that our use of whitespace is wrong.

Complaints have not been about the code, but the gaps between code.

Prioritize the complaints.

We hard-coded XTerm color control sequences, bypassing termcap.

That was more than ten years ago.

No one has noticed.

Sometimes, what looks like an expedient shortcut is perfectly good.

We had a very long and detailed tutorial page on the site for years.

To go through and read it all would have taken at least an hour.

At the very bottom, was a video of a band playing the
Mexican Hat Dance using only hand-fart noises.

No one ever mentioned this.

Keep your tutorials short.

Presence at industry events is important.

Offering talks and workshops helps make people aware of your project.

Virtual teams work well,
but it gets even better after meeting in real life.

It is good if the members of your team share the same sense of humor.

If not, be careful writing messages with an ironic tone.

A development-class machine is no indication
of the kind of hardware and software your users are running.

Dependencies and tools are often far behind the latest versions.

Respond to every means of communication.

It is worth it.

Have a recognizable logo.

Do not make the logo yourself, if you are not a designer.

If you have no budget, ask a designer to judge your work.

Offering gratis stickers is great,
having SWAG – Souvenirs, Wearables And Gifts – users can choose from is even better.

People love to make mashups of two things, or add an extension to a thing.

Very few contributors want to work on the thing.

Create a website containing the philosophy behind your project to help people understand what your project is about.

Calm down, take a deep breath and look back at what you achieved.

Details, mistakes, compromises, incomplete plans and unfulfilled wishes are only visible from inside the project. Be proud, and make new plans.

Epilog

That’s all!

Dirk Deimeke, Taskwarrior-Team, 2017, CC-BY

dirk@deimeke.net

d5e.org – speakerdeck.com/ddeimeke

Download the PDF from Speakerdeck for clickable links.