Building a better community

Your software has come a long way – it's finding interest and uptake in other groups and its user community is growing. Like a proud parent you're happy with your child's achievements, now it's all grown up and successfully branching out on its own. But how can you help your progeny do even better and make even more supportive friends now it's flown the nest?

Building a community around software is an important step for its sustainability. This briefing paper is for researchers, developers and managers of a software product that has started to gain a community. This paper will discuss how to increase your community and encourage contributions from it.

The ultimate goal of community building is to nurture an active supporting community. An active community can become self-supporting: answering queries raised by community members and contributing new functionality and bug fixes back to your project. This gives you more time to develop your software and reduces the resources needed to sustain it. In this way, a community can help you make your software more useful to more people.

Why is community building important?

Software has a distinct lifecycle. As software matures, the way it is developed and people's expectations will change. One of the significant steps occurs when when your software becomes a viable product rather than just a project. Indicators of this change include having funding from more than one source, spending a larger proportion of time on support than on new development, having many active users outside your direct collaborators, and having users who are willing to contribute back to your project.

Why change what I'm doing?

The widening of your user community means that you need to consider different ways of harnessing the growing set of users. Your community could stagnate if you don't reassess the way it is run and supported. This could include changing the way you govern your project and the way you develop your software.

When your community begins to grow, it can begin to determine the future direction of your software, and this direction may not have been in your initial plan.

How to help your fledgling software

So what's the best way of looking after your software when it first ventures out into the big bad world? Become a helicopter parent!

Ask the neighbours if your kid's behaving themselves

Engage your community by actively seeking feedback from as many sensible routes as possible. As well as mailing lists, consider user surveys, end-user groups drawn from the community, exhibits and demos, hackathons, newsletters and videos.

Some of the best feedback comes from watching users as they use your software, so spending time with your community is important.

Even ask that grumpy old couple who complain about the noise

Some of the best advice can be the advice you don't want to hear. Find out what really matters to your

community and act on it, even if you feel that the issues raised are not the most important or interesting. It is important not to lead the community too much - you need to elicit their views, rather than just get them to answer your questions. Get back to community members and ask them to expand on their answers to survey questions.

Allow others to help with the DIY

Try to lower barriers to contribution. Part of the community will form the base of your extended development team, so the more people you have contributing the greater the pool of potential developers. Get rid of obstacles. Examine the governance and licensing models to ensure that it is clear how people can contribute. Make it easy to build on your code by refactoring your architecture to provide clear extension points – this will also encourage reuse. Review your documentation regularly and make sure that it is clear and up to date. And always remember to give credit for contributions, no matter how small!

What do the professionals do?

Professional software companies invest a lot of work into understanding their market and the users who form their community. Different mechanisms for engagement are used, from websites, mailing lists and discussion forums, to more formal methods such as setting up User Groups, early access/beta tester programmes and software ambassador schemes.

These methods can be very successful at nurturing an active community of users, but care should be taken with some of the more involved methods which can take relatively high levels of effort to run well.

What to do when your software finally flies the nest?

The ultimate goal of community building is to nurture an active supporting community that can answer queries and contribute back to the project. This might reduce the amount of control you have over the project, but it pays off: you will a more robust and flexible development team, which be less centralised. It also means that the software is in tune with the needs of its community, leading to wider and increased reuse.

Further information and useful resources

OSS Watch - How to build a community

http://bit.ly/1s3NsH

OSS Watch Software Sustainability Maturity Model

http://www.oss-watch.ac.uk/resources/ssmm.xml

Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel

http://producingoss.com

Bruce Berriman's talk on Software Sustainability at IPAC presented at the SSI's Software Sustainability Workshop: Stories and Strategies at AHM2010

http://tinyurl.com/2wmoka

curtis+cartwright

Software Sustainability Institute

JISC

This briefing paper has beevn preprared as part of a JISC-funded project by the Software Sustainability Institute in partnership with Curtis+Cartwright Consulting Ltd. It is part of a series of outputs that have been devveloped to support the HE sector by raising awareness of software sustainability and preservation issues in the software-development process. Images courtesy of blmurch (front) and Viajar24h (back).