Developing Linux for the Common Woman (and her husband)

Abstract

Moving from proprietary software to Open Source is a cause of fear and trepidation for many users.

There is such a strong technical base in the GNU/Linux and BSD user base that making the move from proprietary to Open Source is almost taken for granted. Or, potentially even more damaging, seen as a challenge to be overcome where the average user sees a terrifying ordeal to be faced.

Companies spend huge amounts of time and money managing change (and resistance to it) amongst their employees to new working practices and methods. Yet, on the home desktop, the fear of change is often ignored or even forgotten. The average user has no helpdesk available to steer her through the foibles of a new piece of software or a new way of doing things.

If Open Source can succeed and become the tool of choice in the workplace, then the desktop market can follow. How can the developer and day to day user work together to ease this adoption, driving development in the future?

It is the authors' intention to look at the future of Linux from a new (albeit fairly technically competent) users standpoint and look for ways in which this migration can be eased.

The author will present his solution to this problem, using his own experience of software testing, training and integration to highlight the pitfalls that can face the complacent developer and slow the growth and acceptance of new developments.

These simple yet often ignored development ideas can ensure usability and user understanding at all levels of the development process or even keep the developer focused and on track when she wanders off on her own wish list of tools and toys.

After all, she can programme the video so she cannot be wrong. Right.....?

Introduction

Many if not all of the people who sit here today are long standing users of OSS. They are comfortable with the thought of 'getting their hands dirty' building and configuring their distro of choice to achieve what they wish.

So comfortable they take their skills and knowledge for granted.

While Linux (and other OSS such as BSD) have became not only accepted in the server room but actively requested and sought, there is still resistance to widespread adoption on the desktop.

Big wins such as that in Schwäbisch Hall[1] earlier this year are slowly pushing Linux onto the desktop and into the public eye. Combined with the open consideration of Linux by many governments all over the world, particularly in EMEA and South America[2], there is now a need to look at how we can make 'our' OS more welcoming to those who can't, won't or just plain don't want to learn anything new.

Governments the world over are now realising that they are committed to supporting proprietary developers who are not necessarily committed to supporting their users.

The three common themes espoused by most for considering the change away from developers such as Microsoft are:

Cost. Governments can save hundreds of thousands and even millions by moving away from the Microsoft lock in.

Security. Why trust data at all levels to a company with no proven record of security?

Interoperability. Software should work. There should be transparency and an ability to take your tools with you wherever you go.

[1] A quick over view of the decision is available online at http://www.oss.gov.za/modules.php?op=modload&name=News&file=article&sid=35

[2] A quick rundown of the major world governments considering OSS is available at http://www.pcworld.com/news/article/0,aid,101879,00.asp

So Where Do We Go From Here?

To succeed on the desktop, OSS needs to be open and transparent in use. To quote a famous DIY advert here in the UK, ‘it does what it says on the tin’. We have to go one step better and allow the user access without reading the instructions. When was the last time you bought a new TV/phone/video recorder/widget/whatever and actually read the manual *before* you even plugged it in?

Thought so :)

To achieve this transparency and ease of use, the OSS developer needs to take a step back and change her approach to systems design and testing. To take a prominent role on the desktop, Linux needs to be approachable.

Asking a new user to ‘Start… Find… Files or Folders… ABC.doc’ or ‘Click the Foot… Search for Files… ABC.sxw’ is something that any harassed helpdesk worker or even the most basically knowledgeable user in the 21st Century can do. Asking the average user to ‘Open a terminal window. The icon at the bottom left that sort of looks like a black monitor or a TV. Yes, that one. That’s right. You have a white window with black writing. Just type exactly what I say. OK so type grep –i –d recurse ABC.sxw $PWD. Yes, $PWD. Oh, by the way. The $PWD must be in capital letters. Yes, capitals. OK now read it back to me. Seems OK now press enter. Oh. You are back to the $ sign? OK so between……’ and so on and so forth isn’t. How do we strive to ensure the same fundamental ease of use and description for all our work?

In his infamous paper ‘The Cathedral and the Bazaar’[3], Eric S. Raymond puts forward the idea that peer acceptance is what drives OSS development. If it is working (and sexy) you can and will impress your friends and peers. Mr. Raymond suggests (and is widely accepted as right) that it is this desire to impress that drives our Community to achieve bigger and better things at all times.

Now, though, we perhaps need to seek acceptance and social standing amongst a larger and less select group of peers. The end user community. How, without spending hundreds of man hours, do we achieve such transparency of interface?

[3] Many online sources for this work exist including http://fringe.davesource.com/Fringe/Computers/Philosophy/Cathedral_Bazaar/

Meet the Testers

We all have our friends, family and peers who test our work in different ways.

Friends test it and say either ‘Cool’ then ask if you are going to the match on Saturday or ‘After it sends that query out over the network, I want it to time the latency in milliseconds and remind me to put the kettle on’. They rarely ask you ‘Why did I have to type SELECT * FROM gadget, thingummy,wotsit SORT ASC; <enter> just to get a recipe for chicken noodles?’

Family tend to say ‘After Eastenders’.

Peers tend to say ‘So you used a vector? OK but would it have not been more efficient to…’ or ‘Hey, good work. I’d forgotten all about the LaserBlasterPhotonCannon class and would have never thought to implement it and just override the ‘KillAliens’ function. Isn’t that the one Peter wrote the ‘Kill_Kill_Kill_Them_All’ function for on his wedding day? I remember it made him three hours late and he turned up in his ‘Real men do it on the C side’ T Shirt and carrying a flask of hot java. Last time I was talking his wife Mary through recompiling her kernel she still hadn’t forgiven him for the photo’s.’

None of the above really help you convince the ‘average’ user that there are better options than ‘Windows 98 and that nice little paper clip man who winks at me’ anymore.

At this moment I’d like to introduce you to my three hard core testers.

Meet Donna. She works in admin for a large corporation, using proprietary products every day. Most of her data is stored in a couple of local databases and one central store. She works with the data locally, using a number of customised spreadsheets and occasionally a word processor. All Donna wants to do is complete her work efficiently and effectively. While she considers herself to be technically weak, she knows the tools she uses well and manages some surprisingly complicated data sets.

So why is Donna important to you as a developer? While you may never meet her or even pass her in the street, she is your end customer. The person you most need to convince your product has value. Make her life hard and you have made your own impossible.

Next we have Adele. When not subjecting her father to sleepless nights of worry, she is a secondary school student just over half way through her GCSE's. Again, she only uses proprietary software at school. At home she uses anything from IM software to keep up with her friends through card games to 3D games. As far as productivity tools go she uses a word processor regularly, spreadsheets occasionally and once or twice has had to face small database systems.

So what does a fifteen year old pimply youth have to offer the professional developer or hard core hacker? Well, I remember being fifteen and am sure I knew *everything* about everything. In two to three years time she'll be in college or maybe even University, studying IT and maybe even programming in two or three languages. When she wants to push her limits of learning, she'll turn to the tools she is used to. Give her the tools she needs now and make sure she can use them. In five years time she'll be either a developer herself or specifying the tools her developers and users have access to.

Finally we have Tom and Bill. Tom is a senior manager in a national company. Bill is a line manager in the same company. While Tom signs the cheques and is the guy the salesmen wine and dine, Bill actually uses (or not) the end product. Again, Tom is important but we have to identify and identify with Bill. Make his life hard and you have lost him. With him you have lost the exposure, the users and the contract.

Conclusion

To ensure the future success of OSS development and to gain mind and market share, developers have to move away from the present development model.

Developing for peer acceptance creates a set of very powerful but often complicated and unfriendly tools.

The challenges facing Linux growth and development are not lack of skills or suitable tools, but understanding the expectations and abilities of a whole new user base who have different needs from computing.

Identifying and communicating with that user group is the first obstacle to mainstream acceptance and the sooner we overcome it, the sooner we'll see OSS in its rightful place as the software of choice for anyone and everyone who needs reliability and integrity in their computing environment.