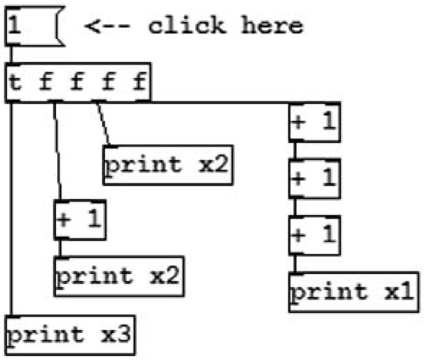
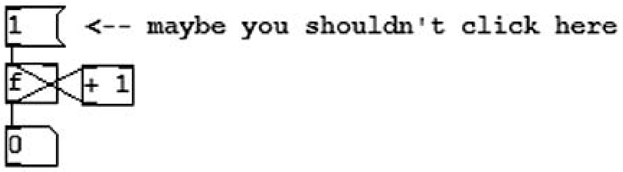
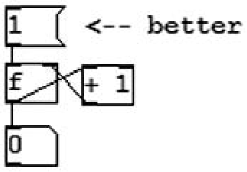
In Pd, message passing is depth first, so that in this patch:



... you get "x1" first, notwidthstanding the fact that "x2" and "x3" appear to be closer to the source. This means that you shouldn't do this:



... because the "depth" is infinite. The counters you've seen always have the message chain terminated somewhere in a cold inlet:



updated for Pd version 0.34

Why Do People Develop Free Software?

Susanne Schmidt

Every year around the time of German finals at school, I get some emails concerning a howto I wrote a few years ago. It is a short “anti-slacking howto for young nerds”, written after my experiences working as a project manager, when I saw some young nerds and geeks quitting school due to the enticements of the “new economy” and its job offerings. The emails I get are (mostly) grateful and friendly. Every time I read one of them, I think: “Well, it was worthwhile writing it.” If I ask myself why I published the howto instead of just emailing it to friends who could possibly benefit from some hints against slacking around, well – I don’t know. I never even considered anything other than publishing it. After 12 years with Linux and Open Source software I have become very used to the idea of publishing for free and for everyone. Shareware? Freeware? Well, why bother with half of the cake, if I can have all of it – the complete source? The phenomenon of Open Source has crept into everyone’s computer and Internet behavior: It is not just software and source code, published freely, it is also projects like Wikipedia and more and more scientists publishing their research results and theories, making them available to everyone on the net. The main issue here is familiarity with the idea of creating Open Source software and sharing it with others. Why do people do that?

It is obvious that people need and use Open Source – not just a small howto, but documentation, information, raw data, software, icons and pictures, algorithms, patterns, sounds and music, and they like it, value it and sometimes they are even truly grateful. This is very motivating. We sometimes call it “fame and glory through Open Source” – more in an ironic sense, but all the developers I have ever met are rather flattered if someone tells them their software really is used and valued. It is also sometimes amazing to see what people do and create with the developer’s software – especially if your software facilitates a very immediate, direct experience like music applications, sound, graphics, pictures or movies do. You might say that one motivation could be to make users happy. Or, in dry, passionless words: “In the world of Open Source software development, actors have one more degree of freedom in the proactive shaping and modification of technologies, both in terms of design and use.”[1]

Why do developers spend hours, weeks, months and even years programming a piece of Open Source software in their spare time? Well, it’s fun. You may call it more sophisticated “intrinsic motives” and wrap an intimidating theory all around it, but there is the pure joy of craftsmanship. Just as I like – maybe love – writing, they have a passion for hacking and coding. It is satisfying to look at a final piece of code, checked in into the source code repository, and see peo-

[1] Brent K. Jesiek, “Democratizing Software: Open Source, the Hacker Ethic, and Beyond”, First Monday, Volume 8, Number 10 (October 2003), <http://firstmonday.org/issues/issue8_10/jesiek/index.html>

ple using and adopting it. Even if people start to nitpick about your style, your code or your programming language of choice, you can still remain happy with the project. A student of graphical design took my howto one day and made a print version to be distributed freely as a students' brochure at the university. In this way, the effort that one person makes may lead to an interesting project or idea from another person – we all are happy, if someone actually uses our library, our patch or our piece of code and finds something useful to do with it. For my part, I was rather proud and flattered – who wouldn't be? The satisfaction of craftsmanship and puritanical, Protestant "usefulness" can be very forceful: making things work, because you can, making things work even better, because you give all your ability and perfectionism into it – that can be very rewarding. Having hacked a small tool or having written a nice piece of documentation feels the same way as looking onto a freshly painted wall in your apartment: one can see one's work and one can see if it is well done. Sometimes, writing Open Source software is truly a labor of love, and in a way it is the geek's way to express political engagement rather than walking a protest march through Washington – Berlin – Paris – Tokyo.

The second reason is the insistent, nagging feeling of "But someone has to!". Dissatisfaction with existing projects, sometimes the unaffordable price of commercial software for a small library one needs right now, badly written software crashing and leaking your memory, the non-existence of something that would be really useful or is needed for another project – these are just a handful of reasons why developers spend so much time on Open Source software. All this may lead to an effort of "ok, I will just do it now", and it can be rather simple: If you don't start with it, maybe no one ever will. The Italian economists Andrea Bonaccorsi and Cristina Rossi just put it in economic terms of "incentives" and filling a gap in the market.[2]

Sometimes it is the sheer lack of a tool you need that leads to a new project. Sometimes it is the vast of existing tools that another operating system offers, but yours does not, and sometimes it is the poor quality of an existing piece of software that leads to many weeks with short nights and bad food – and a new tool. Sometimes it may even be a feeling (or simply imagining) that "I can do better!" that results in a new project.

The amazing thing with Open Source software is that if you have the ability to program, in a way the world is yours. You may compare it to someone who is a great manager and has the skills to organize and manage huge exhibitions or establish a political movement. With the development of software, we have the ability to change circumstances and environments – at least from time to time. Sometimes, the idea of making software Open Source can be very powerful – just look at projects like the former StarOffice or Netscape, which changed into Mozilla which leads us to Firefox. See the revolving influence of any peer-to-peer software and the fights around DeCSS, and you get an idea of what is possible today with the right software project.

---

[2] Andrea Bonacorsi and Cristina Rossi, "Altruistic individuals, selfish firms? The structure of motivation in Open Source software." First Monday, Volume 9, Number 1 (January 2004), <http://firstmonday.org/issues/issue9_1/bonaccorsi/index.html>

The tendency of some industries to consider their customers as too stupid to handle a tool of any complexity often hampers the capabilities a program could have, and suddenly you are limited solely by marketing decisions in what you can do with your application and what is no longer possible. Years ago, I found an interesting marketing study somewhere on the web about "Lego" and why there is so much "theme-oriented" Lego, like Star Wars-Lego, Lord of the Rings-Lego, Harry Potter-Lego and Biff and Bim-Lego, and not much basic Lego bricks anymore, as I remember from my childhood: Because "the customer is too overwhelmed by the flexible choices and isn't able to develop the creativity needed to build really cool things". In other words, your spaceship does not look as cool as that stylish "bird of prey" superfighter on the wrapper and never will. This is exactly the same for many applications, which are intended to be "more smart" than the user, "make things convenient" for the user, and don't make it too difficult for me! I'm not talking about a very well developed user interface to improve the handling of application X – by all means, using it should be as fast and easy as possible – I am talking about hiding capabilities. The price one pays is always vanishing capabilities and the disappearance of flexibility in the application: what you can do is suddenly limited by design and industry, not by your ideas and – let's say – physical laws. There is the wonderful German term of "Volksverdummung" (brainwashing of the people) which applies perfectly here. Or, even worse, software overzealously hides certain capabilities due to possible copyright frauds or legal problems somewhere in some country on this planet. Don't mess with business.

Open Source software may give you (back) that freedom and choice. That is another reason why people hack Open Source. One may argue that this freedom is a liberty for a few initiates, those developers who are capable of handling complex and demanding applications – but it is the principle that counts here. In principle, you can change the code. I still agree with Eric Raymond's optimistic view on Open Source, even if some people see a barrier for "ordinary people" because of the complexity of large Open Source projects: Nikolai Bezroukov wrote in "First Monday" that "[...] Open Source, in Raymond's view, is just source code, not all of the complex infrastructure and implicit knowledge that are used in large software projects."[3] Nevertheless, in principle, you can build any kind of application on your own to suit your personal needs. It is like arguing that certain philosophers can no longer be read, because they are too demanding and customers cannot be expected to learn a foreign language. "Dear citizen, due to the fact that Marx is too demanding to understand, and his ideas are not compatible with the existence of the Department of Trade and may lead to license and copyright fraud and can cause a revolution, and you cannot be expected to consider learning German, we offer you our new product PaterNal 2.0, now with a superior decisionless graphical interface!" The steep learning curve of software sometimes asks for much effort on the part of the user, but most of the time the effort leads to a deeper knowledge and understanding of the inner world of computer software – an ability much needed and very useful in a thoroughly networked

[3] Nikolai Bezroukov, "Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)", First Monday, Volume 4, Number 10 (October 1999), <http://firstmonday.org/issues/issue4_10/bezroukov/index.html>

and computerized society. As Open Source developers seek to make their project the best one in the sense of what the application can do, rather than in terms of being easy to use, many projects in turn give users the flexibility they demand. An Open Source developer is not required to meet the demands of a marketing department, and that can be very liberating.

Nevertheless, Open Source is not healing the planet and it is not happy faces everywhere. If one takes a look at Sourceforge or Freshmeat, it sometimes looks like a huge graveyard with tombstones of alpha-versions dedicated to unfinished and abandoned software projects. One may also ask if we really need 36 web servers (search string "httpd") instead of developing four or five for different purposes and scenarios. Freshmeat even lists over 1000 projects with the search string "editor" – well, if that isn't freedom of choice! Things have changed during the last 12 years and so has the type of personality of the developers programming Open Source software. Today, even 10-year-olds have the possibilities and technical resources to start developing a new operating system and can rely on the large amount of code examples available on the Internet. Open Source software today has some tendency to be focused more on cute, neat and funny things, rather than on solving boring or difficult problems or bothering with necessary clean-ups of some projects. This is the other side of the coin that everyone can write code they like – it is not that easy to convince a young developer to engage in projects that are more necessary than "cool". Many developers of the first generation of Linux tools, Internet applications or device drivers are disappointed, over-worked or simply have other things to do. Therefore, the world of Open Source is changing rapidly.

Open Source software development is also still a male-dominated business, I am sad to say – and this is not due to the fact that an Open Source project is not the ideal environment for curious girls to take their first lessons in computer programming. Even with all the possibilities and freedom and access for everyone, women still are a very small minority in the free software business. I can offer no explanation for that phenomenon – obviously even the impact of Open Source software and the public recognition it has reached in the last decade is still not enough to attract (many) women. Here and there we can see a few female developers, and I'm glad to see them, but we are still far away from taking half of the cake.

Yet even though the history of free software and the last decade have been very bright and promising, things are changing rapidly, and there are some clouds in the sunny sky of free software: software patents, for example, just to mention one problem that can throw projects into serious trouble with one legal stroke. One might remember the legal issues of distributing strong encryption - we now face legal issues forcing programmers to find a way to creatively bypass patented graphical algorithms or mechanisms to perform a search. It is not just the Department of Defense giving computer enthusiasts a hard time "in times of terror" – it is the movie and music industry and the World Trade Organization's demands that developers face today.

The cultural battle in the software business is fought along the lines of legal issues of Digital Rights Management, software patents, privacy and civil rights. Now and then, on an IRCchannel, you can more and more often hear a developer talking quietly about not releasing code or hesitating to publish an idea, because he is afraid of legal issues. And this is not the bright future where young developers may eagerly start new Open Source software projects that the world still needs.

Further Readings:

Eric Raymond. The Cathedral and the Bazaar. 09 Jan. 2006 <http://www.catb.org/~esr/writings/cathedral-bazaar/>.

Eric Raymond. Homesteading the Noosphere. 09 Jan. 2006 <http://www.catb.org/~esr/writings/cathedral-bazaar/>.

Bruce Sterling. A Contrarian View of Open Source. 09 Jan. 2006 <http://www.oreillynet.com/pub/a/network/2002/08/05/sterling.html>

Michelle Levesque. Fundamental Issues with open source software development. 09 Jan. 2006 <http://www.firstmonday.org/issues/issue9_4/levesque/>