Chapter 8:

A foot in the door--
How to train sympathetic developers and infiltrate other projects

Wikipedia is the encyclopedia that anyone can edit, but there are all sorts of edits that you aren't allowed to make. Free software projects let you copy, fork, change directions, and create entirely new software. It wouldn't be free software otherwise.

Open source has a slightly different focus-- a lot of it is about the organizational culture of development. This is an obvious “in” for people with experience in organizations and the corporate world.

Free software was formerly allowed to focus on software development-- creating organizational overhead only if and when necessary (the majority of free software projects have only a single active developer, or a small handful of contributors.) Open source stresses that the project isn't really “open” until there is an entire social pathway for making the contributions allowed by the license.

	Free software development	**Open source development**
License	FOSS	FOSS
Developers	1 or more	1 or more
Website	Optional	Basically required
Source Repo	Ideal	Required
Way to Join / Contribute	Not always-- just fork or send a patch	Basically Required
Organizational Management	Optional / As needed / To be co-opted	Basically Required / Corporate / Strict

These are not always spelled out and pretty much constitute different attitudes towards software development. Being closer to corporations, Open source has more corporate culture in its processes.

While the “Open source way” may look better for letting everyone be a contributor, it carries with it extra requirements and additional reasons to exclude projects from consideration or people from projects.
There is a difference between implicit exclusion and explicit exclusion. While free software may implicitly exclude contributors, it (like Open source) allows

anyone to fork. Open source does more to fight implicit exclusion, but adds more vague rules for explicitly forcing people out of the opportunity to contribute based on organizational priorities, even if that person requires (and has) little or no contact with other developers.

Just as an example, free software used to have more focus on the needs of a single developer writing their own project. If someone had a contribution, they could typically contact the lead (or sole) developer and offer a patch or offer to join the project. They would always be free to create their own version of the software without permission (as permission is already granted by the free software license) but this is usually more effort than simply offering to help.

The lead developer is free to do basically whatever they want-- keeping a project true to its roots. If their personality is not to the liking of other developers, it doesn't necessarily matter-- they don't have to join and the lead developer doesn't have to invite anybody. This works for many projects of small to medium size.

Open source brings organizational overhead and corporate culture into every project-- you can be the leader of your own project and do what you want to with it, but now you shouldn't-- every project should have a community, a code of conduct (which will be applied unevenly between contributors and leaders, though it may ultimately threaten the structure of the leadership in the distant future) and a dedicated website. With free software, these “requirements” only draw resources when they are justified and considered necessary.

Open source kind of hates on lone developers.

Fortunately, Open source brings all this overhead to a project in a way that makes it easier to steer or influence (or purchase) the direction of a project. And since for 20 years, companies like Microsoft have sought to buy, charge royalties for, influence or eliminate the work done by competitors, Open source gives us (and even fights for) the foot in the door that we need to do so.

George Bernard Shaw once said: “I learned long ago, never to wrestle with a pig. You get dirty, and besides, the pig likes it.” With the very corporate culture of Open source, big companies are the pig. Open source is always wrestling with monopolies like Microsoft, and it is of decreasing concern how dirty everyone gets, or how much the pig likes it-- in fact, Microsoft has come out to say they don't merely enjoy wrestling with “Linux”-- they **love** it.

And why not? They love royalties from Android devices. They love royalties from USB drives. They love being able to purchase entire networks of developers, from Linked-in to Github-- which probably contain more valuable data about free/open source developers than Facebook has access to.

The amount of influence Microsoft has over all projects on Github, all projects that use .NET or Azure Cloud, many projects that use Python, and all software distributions that run on Intel-based hardware is only increasing. And Open source continues to pave the way forward for monopolies to own and direct free software-- which was originally created to be independent of control by monopolies.

Free software developers seem to care very little about this, because they have their stripped free software versions of everything open source. So what if we make things less modular, more brittle, more bloated, and more poorly designed? They only use projects with a license allows them to clean up after us, so they're content no matter what we sabotage. We can overwhelm them and send them to clean up mess after mess, with the remaining effect of steering key projects to work more the way we want, and them accepting our changes.

Free software and Open source used to have smaller differences between them, and yet for two decades our sponsorships and initiatives (and purchases) have put us directly in the middle of Open source development.

Since this is fine according to the culture of Open source, they can't really point to a reason why we shouldn't-- they adopt our rules, we play by our rules, they can't complain. It's too late to change to the rules to exclude us, and we aren't doing anything they didn't welcome in the past.

In exchange for software with more churn, more bloat, less choice and less user control and reliability-- they get “cooler” software tools, larger sponsorships, bigger marketing and events that feature their software-- everything they would enjoy if we took over their world and did things our way. And we still get royalties and the chance to steer development away from things that help our competitors more than they help us.

The plan has never actually changed, but our way of framing it has changed entirely. We de-commoditize protocols. We add features we want and deprecate ones that people rely on, and we tell them to get with the program. We create the same kind of lock-in (in practice) by decreasing the compatibility with trusted development tools and utilities, so we can move more quickly (and drag users along) from one industry fad to another.

We say this leads to more compatibility-- but it's more compatibility with the things we care about, and less compatibility among the free software ecosystem they created for themselves. Essentially we drag them out of their world, and back into ours. It's a reunion we have already monetized, already seized projects with, already used to purchase things that supposedly belong to everyone.

The shift in these projects demonstrate that they belong a little more to us, than they do to anybody else. And just as the icing on the cake, we make more money from the development of these projects than the developers sometimes do-- it's as if we hired them ourselves, but all we did was participate and influence. Recently, a company used Wikipedia servers as a blank canvas for their own corporate advertising and message. They were called “bastards” and “vandals” and their changes were reverted.

It's quite different in the world of Free and Open source software. We can behave exactly like The North Face, do just as much to vandalize and be bastardly, we can even stake claim to their work (and have them agree it is our own!) and become wealthier and gain a heroic reputation in the process.

It takes all the professionalism in the world not to quote what Mark Zuckerberg said when he realized how many people trusted him with their personal lives and information.

Instead, we will just sum up the majority response to our infiltration and increasing control of free software development, no matter how aggressively we buy up and steer projects the way we did before free software entered the mainstream:

“**Suckers**.”

Relevant quotes from the Halloween documents:

“it provides us with a very valuable look past Microsoft's dismissive marketing spin about Open Source at what the company is actually thinking”

“If publication of this document does nothing else, I hope it will alert everyone to the stifling of competition, the erosion of consumer choice, the higher costs, and the monopoly lock-in that this tactic implies.”

“The parallel with Microsoft's attempted hijacking of Java, and its attempts to spoil the 'write once, run anywhere' potential of this technology, should be obvious.”

“'De-commoditizing' protocols means reducing choice, raising prices, and suppressing competition.”

“for Microsoft to win, **the customer must lose**.”

“other OSS process weaknesses provide an avenue for Microsoft to garner advantage in key feature areas such as architectural improvements (e.g. storage+), integration (e.g. schemas), ease-of-use, and organizational support.”

“To us, open-source licensing and the rights it grants to users and third parties are primary, and specific development practice varies ad-hoc in a way not especially coupled to our license variations. In this Microsoft taxonomy, on the other hand, the central distinction is who has write access”

“Open source software has roots in the hobbyist and the scientific community and was typified by ad hoc exchange of source code by developers/users.”

“to understand how to compete against OSS, we must target a process rather than a company.”

“Coordination of an OSS team is extremely dependent on Internet-native forms of collaboration. Typical methods employed run the full gamut of the Internet's collaborative technologies”

“OSS projects the size of Linux and Apache are <u>only</u> viable if a large enough community of highly skilled developers can be amassed to attack a problem.”

“This summarizes one of the core motivations of developers in the OSS process -- solving an immediate problem at hand faced by an individual developer -- this has allowed OSS to evolve complex projects without constant feedback from a marketing / support organization.”

“Because the developers are typically hobbyists, the ability to `fund' multiple, competing efforts is not an issue and the OSS process benefits from the ability to pick the best potential implementation out of the many produced.”

“Note, that this is very dependent on:

- A large group of individuals willing to submit code
- A strong, implicit componentization framework (which, in the case of Linux was inherited from UNIX architecture).”

“although debugging requires debuggers to communicate with some coordinating developer, it doesn't require significant coordination between debuggers. Thus it doesn't fall prey to the same quadratic complexity and management costs that make adding developers problematic.”

“Some very large projects discard the `benevolent dictator' model entirely. One way to do this is turn the co-developers into a voting committee (as with Apache).”

“The GPL and its aversion to code forking reassures customers that they aren't riding an evolutionary `dead-end' by subscribing to a particular commercial version of Linux.”

“In particular, larger, more savvy, organizations who rely on OSS for business operations (e.g. ISPs) are comforted by the fact that they can potentially fix a work-stopping bug independent of a commercial provider's schedule”

“Strongly componentized OSS projects are able to release subcomponents as soon as the developer has finished his code.”

“Up till now, Linux has greatly benefited from the integration / componentization model pushed by previous UNIX's. Additionally, the organization of Apache was simplified by the relatively simple, fault tolerant specifications of the HTTP protocol and UNIX server application design.”

“One of the exponential qualities of OSS -- successful OSS projects swallow less successful ones in their space -- implies a pre-emption business model where by investing directly in OSS today, they can pre-empt / eliminate competitive projects later -- especially if the project requires API evangelization.”

“Linux can win as long as services / protocols are commodities”

“The `folding extended functionality' here is a euphemism for introducing nonstandard extensions (or entire alternative protocols) which are then saturation-marketed as standards, even though they're closed, undocumented or just specified enough to create an illusion of openness. The objective is to make the new protocols a checklist item for gullible corporate buyers, while simultaneously making the writing of third-party symbiotes for Microsoft programs next to impossible. (And anyone who succeeds gets bought out.)”

“(This standards-pollution strategy is perfectly in line with Microsoft's efforts to corrupt Java and break the Java brand.)”

“**Monitor OSS news groups**. Learn new ideas and hire the best/brightest individuals.”

“Linux and other OSS projects make it easy for developers to experiment with small components in the system without introducing regressions in other components”

“OSS projects have been able to gain a foothold in many server applications because of the wide utility of highly commoditized, simple protocols. By extending these protocols and developing new protocols, we can deny OSS projects entry into the market.”

“DAV is complex and the protocol spec provides an infinite level of implementation complexity for various applications (e.g. the design for Exchange over DAV is good but certainly not the single obvious design). Apache will be hard pressed to pick and choose the correct first areas of DAV to implement.”

“Systems management functionality potentially touches all aspects of a product / platform. Consequently, it is not something which is easily grafted onto an existing codebase in a componentized manner. It must be designed from the start or be the result of a conscious re-evaluation of all components in a given project.”

“**Ease of Use**. Like management, this often must be designed from the ground up and consequently incurs large development management cost. OSS projects will consistently have problems matching this feature area”

“How can we leverage the client base to provide **similar integration requirements** on our servers? For example, MSMQ, as a piece of middleware, requires **closely synchronized client and server codebases.**”

From https://antitrust.slated.org/halloween/halloween1.html

“These tools are the 'old standbys' of the UNIX development world and are widely used across all Unix platforms. This mass commoditization of development/debug tools is a key contributor to the common skillset efficiencies realized by the Linux process.”

“Additionally, due directly to GPL + having the full development environment in front of me, I was in a position where I could write up my changes and email them out within a couple of hours (in contrast to how things like this would get done in NT). Engaging in that process would have prepared me for a larger, more ambitious Linux project in the future.”

“The endless customizability of Linux for specific tasks - ranging from GFLOP clustered workstations to 500K RAM installations to dedicated, in-the-closet 486-based DNS servers - makes Linux a very natural choice for "isolated, single-task" servers such as DNS, File, Mail, Web, etc. Strict application and *OS* componentization coupled with readily exposed internals make Linux ideal. “

“There are hundreds of stories on the web of Linux installations that have been in continuous production for over a year. Stability more than almost any other feature is the #1 goal of the Linux development community (and the #1 cited weakness of Windows)”

“Linux developers are generally wary of Sun's Java. Most of the skepticism towards Java stems directly from Sun's tight control over the language - and lack of OSS.”

“**GNOME** — Next generation UI initiative for Linux loosely based on X-windows +CORBA . More info at http://www.gnome.org. Many of the key developers for Gnome work for RedHat.”

“A lot more thought and work needs to go into formulating Microsoft's response to Linux. Some initial thoughts on how to compete with Linux in particular are contained below. One "blue sky" avenue that should be investigated is if there is any way to turn Linux into an opportunity for Microsoft.”

“A more generalized assessment of how to beat the Open Source Software process... is contained in the 'Open Source Software' document.”

“Relative to other UNIX's Linux is considered more *customizable*. Addressing this functionality involves more than just the embedded Windows NT project. Greater componentatization & general dependency reduction within NT will improve not only it's stability but also the ability of highly skilled users/admins to deploy task-specific NT installations.”

From https://antitrust.slated.org/halloween/halloween2.html

“Linux is a philosophy as much as technical phenomena. On the positive, and Microsoft is interested in better understanding and finding ways to accommodate this dynamic, it provides for extensive peer review, and for a lot of independent parallel work on a variety of features.”

From https://antitrust.slated.org/halloween/halloween3.html

“We should line up some small acquisitions here to jump start this if we
do it. We shoudl also do this ASAP. Microsoft also indicated there was
a lot more money out there and they would clearly rather use Baystar
"like" entities to help us get signifigantly more money if we want to
grow further or do acquisitions”

“The will help us a lot and if we execute we could exit and Unix componients we have build potentially back to Microsoft or MCS.”

“There you have it. At least a third of SCO's entire market capitalization, and their entire current cash reserves, is payoffs funnelled from Microsoft. Their 10Qs reveal that every other line of cash inflow is statistical noise by comparison. The brave new SCOsource business model is now clear: sue your customers, shill for Microsoft, kite your stock, and pray you stay out of jail.”

“Five days after this memo was written, SCO's PR chief Blake Stowell responded to widespread speculation that Microsoft was behind the Bystar deal by vehemently denying it.”

“We think the kindest interpretation we can put on these events is that Blake Stowell isn't lying through his teeth, but was kept out of the loop so he could honestly deny all knowledge of Microsoft's involvement. If so, we wonder what else SCO's director of PR doesn't know...”

From https://antitrust.slated.org/halloween/halloween10.html

“(Writing code that doesn't suck always has to be our base-level and most important response, but the propaganda war matters too. If it's not already obvious to you why, keep reading.)”

“Let's start by reminding ourselves of the stakes. For Microsoft (or at least its present business model) to survive, *open source must die*. It's a lot like the Cold War was; peaceful coexistence could be a stable solution for us, but it can never be for them, because they can't tolerate the corrosive effect on their customer relationships of comparisons with a more open system.”

“Because coexistence is not a stable solution for them, it cannot be for us either. We have to assume that Microsoft's long-term aim is to crush our culture”

“They seem to have abandoned using the "open source is intellectual-property cancer" argument directly. This follows the advice their own survey group gave them two years ago that this tactic was backfiring badly. Instead they're pushing this line through bought proxies at SCO and elsewhere.”

“Like the dog that didn't bark in the night-time, these omissions are significant, because Microsoft marketing is thorough and ruthlessly opportunistic. You can bet money that the reason they're not making these arguments is because they tried them on smaller focus groups, or individually with key customers, and they didn't fly.”

“Microsoft's underlying problem is that it employs about 22,000 programmers; the open-source community can easily muster ten times that number. That means the capability gap that has opened up between the open-source codebase and Windows is only going to get worse from Microsoft's point of view, not better. Time, technology, and market forces are not on their side — so, to survive, they're going to have to change the game so that market forces and the open-source advantage in technology become irrelevant.”

From https://antitrust.slated.org/halloween/halloween11.html

About the author:

Ted MacReilly is a technologist and tech writer concerned with modern trends in software design and development. He does not work for Microsoft, Apple, or Google, but would like them to continue offering proprietary software and cloudware, without getting too cozy with free software developers.