Karl Fogel

Tvorba open source softwaru

Jak řídit úspěšný projekt svobodného softwaru

Edice CZ.NIC

Karl Fogel

TVORBA OPEN SOURCE SOFTWARU
Jak řídit úspěšný projekt svobodného softwaru

Vydavatel:
CZ.NIC, z. s. p. o.
Americká 23, 120 00 Praha 2
Edice CZ.NIC
www.nic.cz

1. vydání, Praha 2012
Kniha vyšla jako 5. publikace v Edici CZ.NIC.
ISBN 9978-80-904248-5-2

CZ.NIC | SPRÁVCE DOMÉNY CZ

— Věnování

Tato kniha je věnována dvěma drahým přátelům, bez nichž by nemohla vzniknout:
Karen Underhillové a Jimovi Blandymu.

ISBN 9978-80-904248-5-2

— Karl Fogel

Tvorba open source softwaru

Jak řídit úspěšný projekt svobodného softwaru

— Edice CZ.NIC

Předmluva vydavatele

Vážení čtenáři,

naše předchozí publikace v Edici CZ.NIC byly orientovány na nějakou konkrétní technologii, od technologie IPv6 přes programování v Pythonu po používání elektronických podpisu. Touto knihou jsme se vydali do trochu jiné oblasti. Přeložili jsme pro vás knihu, která se zabývá tím, jak produkovat otevřený (open-source) software. Kniha pro vás může být zajímavá ať už jste začínající programátor s dobrým nápadem nebo zavedená firma produkující již existující software.

V Edici CZ.NIC vycházejí především knihy, které jsou blízké vám ale i nám, a není tomu jinak ani v případě této knihy. Centrální registr pro správu domén FRED byl od počátku vydáván pod svobodnou licencí a díky tomu je používán i na dalších místech naší zeměkoule. Mezi poslední přírůstky mezi národními registry používající FRED můžeme zmínit například Estonsko. Stejně se chováme i k dalšímu softwaru, který je vyvíjen ve sdružení CZ.NIC – alternativní klient pro přístup do datových schránek – Datovka, velmi úspěšný projekt směrovacího daemona Bird, nebo náš poslední projekt rychlého autoritativního DNS serveru – Knot DNS.

Autor této publikace Karl Fogel není ve světě otevřeného a svobodného softwaru žádným nováčkem. Kromě psaní o otevřeném softwaru také přispívá do významných projektů jako je Launchpad, Subversion nebo GNU Emacs.

Doufáme, že tato kniha bude přínosná pro Vás a přeneseně i pro celou komunitu okolo otevřeného softwaru, protože po přečtení této knihy bude Váš další projekt pod otevřenou licencí. Přeji Vám příjemnou četbu.

Ondřej Surý
Praha, 17. ledna 2012

Obsah

Přehled kapitol

Přílohy

Předmluva

Proč byla tato kniha napsána?

Když dnes na nějakém večírku řeknu, že píšu svobodný software, už se nesetkávám jen s prázdnými pohledy. „Aha, open source – takže něco jako Linux?“ říkají. Horlivě přikyvuji. „Přesně tak. To je můj obor.“ Je příjemné, když už vaše práce není úplně neznámá. Dříve se dalo i docela dobře předvídat, jaká otázka bude následovat: „Jak se tím dají vydělat peníze?“ Na tohle se dá odpovědět jenom shrnutím celé ekonomické situace svobodného softwaru: že existují organizace, v jejichž zájmu je, aby určitý software existoval, ale které jej nepotřebují prodávat; chtějí mít pouze jistotu, že je tento software dostupný a udržovaný – jako nástroj a ne jako zboží.

V poslední době se to ale změnilo a další otázka se ne vždy týká peněz. Obchodní stránka open source softwaru[1] už není tolik záhadná a velká část neprogramátorské veřejnosti chápe nebo přinejmenším není překvapena tím, že je možné mít v tomto oboru zaměstnání na plný úvazek. Ta otázka, která následuje, poslední dobou stále častěji zní: *„To je zajímavé, ale jak to vlastně funguje?“*

A na to jsem žádnou uspokojivou odpověď dlouho neměl. Čím víc jsem se snažil nějakou najít, tím víc jsem zjišťoval, jak složité téma to vlastně je. Řízení projektu svobodného softwaru je něco trochu jiného než běžné podnikání – představte si, že byste se o podobě svého produktu museli neustále dohadovat se skupinou dobrovolníků, z nichž jste většinu nikdy osobně nepotkali! Z různých důvodů se to nepodobá ani vedení klasické neziskové organizace, ani řízení vlády. Všem těmto věcem se to sice v něčem podobá, ale časem jsem dospěl k závěru, že svobodný software je entitou *sui generis*. Najdete mnoho věcí, k nimž jej lze celkem výstižně přirovnat, ale nic, co by bylo stejné. Ostatně už samotné tvrzení, že by bylo možné projekt svobodného softwaru nějak „řídit“, je celkem odvážné. Projekt svobodného softwaru může být nepochybně spuštěn; může být také ovlivněn zainteresovanými stranami, často velmi silně. Ale výsledky jeho práce se nemohou stát vlastnictvím žádného jednotlivce, a dokud se někde – a to naprosto kdekoliv – najdou lidé, kteří mají zájem v něm pokračovat, nemůže jej ani nikdo zastavit. Všichni mají neomezenou moc a zároveň nikdo nemá žádnou.
A to vytváří zajímavou dynamiku.

Proto jsem se tedy rozhodl napsat tuto knihu. Projekty svobodného softwaru daly vzniknout jedinečné kultuře, étosu, v němž je hlavní zásadou svoboda psát software, který dělá, cokoliv budete chtít, ale kde tato zásada nevede k tomu, že by si každý psal svůj vlastní kód zcela nezávisle na ostatních, ale k nadšené spolupráci. Ostatně schopnost dobře spolupracovat s ostatními je v oblasti svobodného softwaru jednou z dovedností, které se cení nejvíc. K řízení takovýchto projektů je nutné zapojit se do jakési hypertrofní formy spolupráce, při níž může softwaru výrazně prospět jak schopnost pracovat s ostatními, tak i objevování nových způsobů takové spolupráce. Tato kniha se pokouší popsat techniky, jimiž toho lze dosáhnout. Neklade si ambice celé téma vyčerpat, ale přinejmenším je to dobrý začátek.

---

[1] Termíny „open source“ a „svobodný“ (free) software jsou v tomto kontextu v podstatě synonymní. Podrobněji budou probírány v části **„Free“ versus „open source“** v kapitole **1. Úvod.**

Dobrý svobodný software je sám o sobě hodnotným cílem a já doufám, že čtenáři, kteří hledají způsob, jak jej dosáhnout, budou s obsahem této knihy spokojeni. Ale kromě toho také doufám, že se mi podaří předat něco z té čiré radosti, jež vzniká ze spolupráce v motivovaném týmu open source vývojářů a ze sympaticky přímého kontaktu s uživateli, který se v open source projektech často objevuje. Účast na úspěšném projektu svobodného softwaru je totiž velká zábava, což je koneckonců ten hlavní důvod, proč celý systém může fungovat.

Kdo by měl tuto knihu číst?

Tato kniha je určena vývojářům softwaru a těm, kdo softwarové projekty řídí, a to ať už v situaci, kdy o spuštění projektu svobodného softwaru teprve uvažují, nebo když už nějak začali a teď si nejsou jisti, jak dál. Měla by pomoci také lidem, kteří by se do nějakého open source projektu chtěli zapojit, ale nikdy předtím nic takového nedělali.

Čtenář této knihy nemusí být nutně programátor, ale měl by znát základní koncepty softwarového inženýrství, jako je zdrojový kód, kompilátory a záplaty (patche).

Není potřeba ani mít předchozí zkušenost s open source softwarem, a to ani jako uživatel, ani jako vývojář. Těm, kteří již v projektech svobodného softwaru někdy pracovali, budou některé části knihy připadat celkem samozřejmé, takže je možná budou chtít přeskočit. Protože jsem se snažil psát pro potenciálně velmi široké spektrum čtenářů, dal jsem všem podkapitolám srozumitelné nadpisy a vždy zdůraznil, které části mohou ti zkušenější bez obav vynechat.

Zdroje

Mnoho zdrojového materiálu pro tuto knihu pochází z pěti let práce na projektu Subversion (**http://subversion.tigris.org/**). Subversion je open source systém pro správu a verzování zdrojových kódů, který byl napsán zcela od nuly; byl zamýšlen jako náhrada pro CVS, jehož použití bylo v té době v celé open source komunitě *de facto* standardem. Projekt byl zahájen na začátku roku 2000 mým zaměstnavatelem, společností CollabNet (**http://www.collab.net/**), která naštěstí už od začátku pochopila, že je třeba jej vést v duchu spolupráce a distribuované činnosti. Velmi záhy se k projektu začalo přidávat mnoho dobrovolníků; dnes se jej účastní kolem 50 vývojářů, z nichž jenom několik je zaměstnáno v CollabNet.

V mnoha ohledech je Subversion zcela klasickým příkladem open source projektu a nakonec jsem z něj čerpal víc, než jsem původně očekával. Do určité míry je to proto, že to pro mě bylo nejjednodušší – když jsem potřeboval příklad nějakého konkrétního jevu, obvykle se mi hned vybavil nějaký, který jsme zažili při práci na Subversion. Je to ale také zdroj pro porovnávání informací. Ačkoliv se v různé míře účastním i jiných projektů svobodného softwaru a hovořím s přáteli a známými, kteří jsou aktivní v mnoha dalších, při psaní textu určeného k vydání si člověk rychle uvědomí, že by si

všechna svá tvrzení měl ověřit. Nechtěl jsem se vyjadřovat k událostem, které se staly v jiných projektech, jen na základě toho, co jsem si mohl přečíst v archivech veřejných diskuzních skupin.Věděl jsem totiž, že pokud by se někdo o něco takového pokusil u Subversion, dospěl by ke správnému závěru jen asi v polovině případů, zatímco v té druhé by se mýlil. Takže když jsem čerpal inspiraci nebo příklady z projektu, s kterým jsem neměl přímou zkušenost, zkusil jsem nejdříve oslovit někoho, kdo je o něm lépe informovaný a komu můžu věřit, že mi řekne, jak to skutečně bylo.

Subversion byl mým zaměstnáním posledních pět let, ale svobodným softwarem se zabývám už delší dobu – dohromady 12 let. Mezi další projekty, které obsah této knihy ovlivnily, patří:

- Projekt textového editoru GNU Emacs nadace Free Software Foundation, ve kterém udržuji několik malých balíčků.
- Concurrent Versions System (CVS), na kterém jsem intenzivně pracoval v letech 1994–1995 s Jimem Blandym, ale do nějž jsem se od té doby zapojoval jen příležitostně.
- Sbírka open source projektů známá jako Apache Software Foundation, zejména Apache Portable Runtime (APR) a Apache HTTP Server.
- OpenOffice.org, Berkeley Database od Sleepycat a databáze MySQL – těchto projektů jsem se osobně neúčastnil, ale sledoval jsem je a v některých případech jsem hovořil s jejich tvůrci.
- GNU Debugger (GDB) (totéž).
- The Debian Project (totéž).

Tento seznam samozřejmě není úplný. Podobně jako většina open source programátorů i já zdálky sleduji mnoho různých projektů, abych měl přehled o celkovém stavu věcí. Nebudu je tady všechny vyjmenovávat, ale na příslušných místech se o nich v této knize zmiňuji.

Poděkování

Psaní této knihy zabralo čtyřikrát víc času, než jsem si původně myslel, a často jsem při tom měl pocit, jako by mi pořád viselo nad hlavou koncertní křídlo. Za to, že jsem ji dokázal dokončit a zachovat si zároveň své duševní zdraví, vděčím mnoha lidem.

Andy Oram z O'Reilly je typem redaktora, o němž všichni autoři knih sní. Kromě toho, že celou problematiku důvěrně zná (a navrhl mnohá z témat knihy), má i vzácný dar rozpoznat, co jste chtěli říci, a pomoci vám najít ten správný způsob, jak to říci. Práce s ním pro mě byla ctí. Děkuji také Chucku Toporkovi za to, že můj návrh rovnou přesměroval na Andyho.

Brian Fitzpatrick recenzoval téměř veškerý text knihy hned po jeho napsání, což nejen zvýšilo jeho kvalitu, ale také mě udrželo při psaní ve chvílích, kdy jsem chtěl být kdekoliv jinde, jen ne u počítače. Ben Collins-Sussman a Mike Pilato také dohlíželi, zda práce pokračují, a vždycky se mnou ochotně diskutovali – někdy docela dlouze – o jakémkoliv tématu, které jsem se v daném týdnu snažil popsat. Všímali si také, když jsem začínal zpomalovat, a když to bylo nutné, jemně mě postrkovali. Díky.

Biella Colemanová psala svou dizertační práci ve stejné době, kdy jsem dával tuto knihu dohromady i já. Ví, jaké to je každý den sednout a psát, a byla mi inspirujícím příkladem a chápavou posluchačkou. Na hnutí svobodného softwaru také aplikuje svůj fascinující pohled antropologa, čímž mi poskytla jak zajímavé nápady, tak i odkazy na další materiál, který jsem v knize mohl použít. Alex Golub, další antropolog, který je jednou nohou ve světě svobodného softwaru a jenž zároveň také dokončoval svou dizertační práci, mě v počátcích mimořádně podporoval, což mi hodně pomohlo.

Micah Anderson vždycky vypadal, že ho jeho vlastní psaní nijak zvlášť nezatěžuje, což pro mě bylo velkou inspirací (i když přiznávám, že jsem mu to spíš záviděl); hlavně byl ale vždy připraven poskytnout přátelskou radu nebo (přinejmenším při jedné příležitosti) technickou podporu. Díky, Micahu! Jon Trowbridge a Sander Striker mi dodali jak povzbuzení, tak i konkrétní pomoc. Jejich rozsáhlé zkušenosti v oblasti svobodného softwaru mi poskytly materiál, k němuž bych se jinak jen těžko dostal.

Děkuji Gregu Steinovi nejen za přátelství a za dobře načasovanou podporu, ale i za to, že v projektu Subversion ukázal, jak jsou pravidelné revize kódu důležité pro budování programátorské komunity. Děkuji také Brianu Behlendorfovi, který nám taktně vtloukal do hlav, jak důležité je vést diskuze na veřejnosti. Doufám, že se tento princip odráží v celé knize.

Děkuji také Benjaminu „Makovi“ Hillovi a Sethu Schoenovi za rozličné rozpravy o svobodném softwaru a o jeho politických aspektech. Děkuji Zackovi Urlockerovi a Louisi Suarez-Pottsovi za to, že si ve svých nabitých plánech našli čas na rozhovor se mnou. Děkuji Shaneovi ze skupiny Slashcode za to, že svolil k citaci svých příspěvků, a Haggenovi So za jeho nesmírně užitečné srovnání serverů nabízejících kompletní hosting projektů.

Děkuji Alle Dekhtyar, Polině a Soni za jejich neustálé a trpělivé povzbuzování. Jsem velmi rád, že už nebudu muset ukončovat (nebo se spíš neúspěšně pokoušet ukončit) naše společné večery předčasně, abych mohl jít domů a pracovat na „té knize“.

Děkuji Jackovi Repenningovi za přátelství, mnohé rozhovory a tvrdohlavé odmítání přijmout jednoduchou chybnou analýzu, pokud je k dispozici analýza obtížnější, ale správná. Doufám, že se alespoň něco z jeho dlouhodobých zkušeností s vývojem softwaru a se softwarovým průmyslem jako takovým na této knize nějak odrazilo.

Společnost CollabNet prokázala mimořádnou velkorysost, když mi při psaní povolila velmi pružný harmonogram a nestěžovala si, když to trvalo mnohem déle, než se původně plánovalo. Nevím, jak přesně vypadá proces, jímž vedení společnosti k takovým závěrům dospěje, ale řekl bych, že v tom měli prsty Sandhya Kluteová a později Mahesh Murthy – za což jim oběma děkuji.

Celý vývojový tým projektu Subversion mi byl v uplynulých pěti letech velkou inspirací a mnohé z toho, co jsem napsal v této knize, jsem se naučil právě při práci s ním. Nebudu jim zde děkovat jmenovitě, protože je jich příliš mnoho, ale vyzývám každého čtenáře, který někdy narazí na přispěvatele projektu Subversion, aby ho pozval na nápoj podle jeho vlastní volby. Já osobně to dělat určitě budu.

Rachel Scollonovou jsem mnohokrát otravoval nekonečnými stížnostmi na aktuální stav knihy. Vždycky mi ochotně naslouchala a nějak dokázala, že mi po našem rozhovoru všechny problémy rázem připadaly menší. Moc mi to pomohlo – díky.

Děkuji (opět) Noelu Taylorovi, který se určitě musel divit, proč chci psát další knihu, když jsem si u té minulé tolik stěžoval, ale jehož přátelství a vedení sboru Golos pro mě znamenalo, že ani v okamžicích největší pracovní vytíženosti se z mého života nevytratila hudba a dobré vztahy s lidmi. Děkuji také Matthewovi Deanovi a Dorothei Samtlebenové, přátelům a dlouhodobě trpícím hudebním partnerům, kteří projevili velké pochopení pro mé stále se hromadící výmluvy, že nebyl čas cvičit. Megan Jenningsová mě neustále podporovala a upřímně se zajímala o zpracovávané téma, ačkoliv pro ni bylo neznámé – což je pro nejistého autora velká posila. Díky, Megan!

Knihu přečetli čtyři zasvěcení a pilní recenzenti: Yoav Shapira, Andrew Stellman, Davanum Srinivas a Ben Hyde. Kdybych byl schopen do knihy začlenit všechny jejich skvělé návrhy, byla by lepší. Časová omezení mě donutila vybrat si jen některé, které ale přesto celému textu výrazně prospěly. Za všechny chyby, které v knize zůstaly, jsem zodpovědný pouze já.

Mí rodiče, Frances a Henry, pro mě jako vždy byli báječnou oporou, a protože tato kniha je méně technická než ta předchozí, doufám, že pro ně bude i čitelnější.

Nakonec bych rád poděkoval těm, kterým jsem knihu věnoval – Karen Underhillové a Jimovi Blandymu. Karenino přátelství a pochopení pro mě znamenalo vše – nejen během psaní této knihy, ale také během posledních sedmi let. Bez její pomoci bych ji nikdy nemohl dokončit. Podobné je to s Jimem, opravdovým přítelem a skvělým hackerem, který mě jako první učil o svobodném softwaru – asi tak, jako by pták učil letadlo létat.

Prohlášení

Myšlenky a názory vyjádřené v této knize jsou mé vlastní. Nemusí nezbytně reprezentovat pohled společnosti CollabNet nebo projektu Subversion.

1. Úvod

1. Úvod

Většina projektů svobodného softwaru ztroskotá.

O těchto selháních ale obvykle není moc slyšet. Jednak proto, že pozornost přitahují pouze ty úspěšné, a jednak proto, že projektů svobodného softwaru existuje tak velké množství[2], že i když uspěje jen malé procento z nich, pořád to bude vytvářet navenek zcela opačný dojem. O krachu projektů se obvykle také nedozvíme proto, že jejich selhání se nedá popsat jako událost. Nelze poukázat na konkrétní okamžik, kdy nějaký projekt ztratil svou životaschopnost – lidé se od něj zkrátka časem nějak odklonili a přestali na něm pracovat. Existuje sice moment, kdy byla do projektu zanesena poslední změna, ale ti, kteří ji provedli, obvykle ještě netušili, že dál už nic nebude. Neexistuje dokonce ani jednoznačná definice toho, kdy projekt zanikl. Je to tehdy, když se na něm aktivně nepracovalo po dobu šesti měsíců? Když jeho uživatelská základna přestala růst, aniž by převýšila počet vývojářů? Co když jej vývojáři opustili, protože zjistili, že existuje jiný projekt se stejným cílem? A co když se pak k tomuto druhému projektu připojili a rozšířili jej o mnohé z toho, co vytvořili dříve? Skončil vůbec ten první projekt, nebo se jen přestěhoval?

Z těchto a podobných důvodů není možné přesně zjistit, kolik procent projektů uspěje a kolik ne. Různé přibližné informace, které jsem nashromáždil za svých více než deset let v open source, posbíral na SourceForge.net a našel pomocí Googlu, naznačují všechny zhruba totéž: že počet těch, které selžou, je obrovský, pravděpodobně v řádu 90–95 %. Toto číslo pak ještě naroste, pokud započítáme i ty projekty, které sice dosud přežívají, ale už nefungují, tedy takové, které stále ještě produkují funkční software, ale jejichž pracovní prostředí je nepříjemné, nebo jež se nevyvíjejí tak rychle nebo tak spolehlivě, jak by mohly.

Tato kniha pojednává o tom, jak se selhání vyhnout. Nezkoumá jen to, jak dělat věci správně, ale také to, jak se dělají špatně, abyste mohli problémy včas rozpoznat a napravit. Doufám, že po jejím přečtení budete mít k dispozici řadu technik, které vám umožní nejen vyhnout se častým nástrahám vývoje open source softwaru, ale také vypořádat se s růstem a údržbou jakéhokoliv úspěšného projektu. Úspěch není hra s nulovým součtem a tato kniha se nezabývá tím, jak vyhrát nebo jak předběhnout vaši konkurenci. Koneckonců jednou z důležitých součástí vedení open source projektu je i dobrá spolupráce s ostatními souvisejícími projekty. Z dlouhodobé perspektivy přispívá každý úspěch svým dílem k prosperitě veškerého svobodného softwaru světa.

Bylo by lákavé říct, že projekty svobodného softwaru selhávají ze stejných důvodů jako projekty proprietárního softwaru. Nepochybně platí, že svobodný software nemá monopol na nerealistické požadavky, nejasné specifikace, špatné řízení zdrojů, nedostatky ve fázích návrhu i všechny ty ostatní strašáky, kteří jsou v softwarovém průmyslu již dobře známí. Těmito tématy se už zabývalo mnoho

[2] SourceForge.net, jeden z populárních hostingových serverů, měl v polovině dubna 2004 registrováno 79 225 projektů. Tohle číslo ale samozřejmě ani zdaleka neodpovídá celkovému počtu projektů svobodného softwaru na internetu – jsou v něm pouze ty, které si vybraly SourceForge.

knih, proto se jim zde pokusím vyhnout. Místo toho se budu snažit popsat problémy, které se objevují jen ve světě svobodného softwaru. Pokud takový projekt selže, je to často proto, že jeho vývojáři (nebo vedoucí) podcenili specifické problémy spojené s vývojem open source – třebaže na ty známější obtíže spojené s vývojem closed source mohli být dobře připraveni.

Jednou z nejběžnějších chyb jsou nerealistická očekávání výhod open source přístupu jako takového. Otevřená licence nezaručí, že se na váš projekt hned vrhnou davy aktivních vývojářů a začnou mu dobrovolně věnovat svůj čas; to, že otevřete zdrojový kód problematického projektu, neznamená automatické vyřešení všech jeho potíží. Ve skutečnosti je to přesně naopak – otevřením projektu vytváříte celou řadu nových problémů a v krátkodobém horizontu vás to může stát více, než kdyby projekt zůstal uzavřený. Otevření projektu znamená, že je kód potřeba uspořádat tak, aby byl srozumitelný i těm, kdo ho vidí poprvé, že se musí vytvořit webové stránky a diskuzní skupiny (mailing listy) pro vývojáře a často i že je potřeba napsat vůbec první verzi dokumentace. To všechno představuje spoustu práce. A pokud se přece jen objeví vývojáři se zájmem o věc, bude to nějakou dobu znamenat zase jenom další zátěž – předtím, než začnou projektu jakkoliv přispívat, bude nutné nějakou dobu odpovídat na všechny jejich otázky. Jak řekl vývojář Jamie Zawinski o problematických začátcích projektu Mozilla:

> *Open source funguje, ale v žádném případě to není všelék. Pokud z toho všeho plyne nějaké poučení, je to fakt, že nemůžete vzít skomírající projekt, pokropit jej živou vodou zvanou „open source", a vše tak zázračně vyřešit. Psát software je těžké. Jednoduchá řešení tu neexistují.*
>
> (citováno z **http://www.jwz.org/gruntle/nomo.html**)

S tím souvisí i další chyba, kterou je odbývání prezentace a přípravy instalačních balíčků s odůvodněním, že to přece můžeme dodělat později, až se projekt lépe rozjede. Prezentace a příprava instalačních balíčků v sobě zahrnují celou řadu úkolů, které mají společný cíl: snížit bariéru, jež brání vstupu. Pokud má být projekt přístupný pro nezasvěcené, je potřeba napsat dokumentaci pro uživatele a pro vývojáře, vytvořit webové stránky s informacemi pro nováčky, co nejvíc zautomatizovat kompilaci a instalaci softwaru atd. Mnozí programátoři, bohužel, považují uvedené práce za druhořadé – důležitější je podle nich kód samotný. Mají k tomu několik důvodů. Zaprvé jim to může připadat jako zbytečná námaha, protože tato činnost nejvíc prospívá těm, kdo projekt znají nejméně, a naopak. Koneckonců ti, kteří píšou zdrojový kód projektu, instalační balíčky vůbec nepotřebují. To, jak daný software nainstalovat, spravovat a používat, už dávno vědí, protože jej sami napsali. Zadruhé na to, abyste prezentaci a vytváření balíčků dělali dobře, potřebujete mít schopnosti, které se často značně liší od těch, které musí mít dobrý programátor. Lidé se obvykle zaměřují na to, v čem jsou dobří, a to i v případech, kdy by projektu lépe posloužilo, kdyby svůj čas věnovali i něčemu, co jim zas tolik nesedí. **V kapitole 2. Zahájení projektu** se budeme zabývat prezentací a vytvářením balíčků podrobněji a vysvětlíme si, proč je zcela zásadní, aby jim byla dána priorita už od samého začátku projektu.

Často se také objevuje mylný názor, že open source projekty potřebují řídit jen velmi málo, popřípadě vůbec, nebo zcela naopak, že je lze spravovat s použitím stejných metod, jaké se používají při uzavřeném vývoji. Řízení open source projektů nemusí být navenek příliš vidět, ale u úspěšných projektů

se přesto děje – obvykle v nějaké formě v zákulisí. Abychom si uvědomili, proč tomu tak je, stačí si představit následující situaci: Každý open source projekt je tvořen náhodným seskupením programátorů (a tato kategorie lidí, jak známo, bývá často poněkud nonkonformní), kteří se velmi pravděpodobně nikdy nesetkali a kteří prací na projektu mohou sledovat různé osobní cíle. Teď si představte, co by se stalo, kdyby taková skupina nebyla vůbec žádným způsobem řízena. Pokud by se nestal nějaký zázrak, buď by se okamžitě zhroutila, nebo rychle rozutekla. I kdybychom si to přáli sebevíc, věci se samy spravovat nebudou. I když může být projekt řízen celkem aktivně, děje se to často neformálními, jemnými a poměrně nenápadnými způsoby. Jediná věc, která drží skupinu vývojářů pohromadě, je společná víra, že dohromady toho zvládnou víc než každý zvlášť. Hlavním cílem vedení je tedy zajistit, aby této myšlence nepřestávali věřit, a to vytvořením zásad komunikace, zabraňováním tomu, aby byli užiteční vývojáři vytlačeni na okraj projektu jen kvůli svým osobnostním rozdílům, a obecně tím, že projekt bude místem, kam se vývojáři budou chtít vracet. O tom, jak přesně těchto věcí dosáhnout, pojednává zbytek této knihy.

Nakonec existuje ještě obecná kategorie problémů, kterou bychom mohli pojmenovat jako „kulturní nedorozumění“. Před deseti, možná i před pěti lety by bylo poněkud předčasné mluvit o něčem jako globální kultuře svobodného softwaru; to se ale změnilo. Pomalu se zde vyvinula skutečná, jedinečná kultura, která sice rozhodně není jednolitá – k vnitřním rozporům a k vytváření frakcí je přinejmenším stejně náchylná jako libovolná kultura vymezená geograficky –, ale jejíž jádro je v zásadě soudržné. Většina úspěšných open source projektů sdílí některé nebo všechny charakteristiky tohoto jádra. Tyto kultury odměňují určité typy chování a trestají jiné a vytvářejí atmosféru, která podněcuje neplánovanou spoluúčast, i když někdy za cenu centrálního řízení. Mají své vlastní pojetí toho, co je nezdvořilé a co se naopak sluší, které se může podstatně lišit od toho, co je běžné jinde. A co je nejdůležitější, dlouhodobí účastníci tyto standardy většinou přijali za své, takže v názoru na to, jaké chování se v této kultuře očekává, se zhruba shodují. Neúspěšné projekty se od tohoto jádra obvykle výrazným způsobem odchylují, třebaže možná neúmyslně, a často v nich nepanuje shoda, jak by rozumné standardní chování mělo vypadat. To znamená, že pokud se objeví jakýkoliv problém, může se situace začít prudce zhoršovat, protože účastníci nemají dostatečně zažité kulturní reflexy, které by se pro řešení takových rozporů daly použít.

Tato kniha je praktická příručka a ne antropologická studie nebo historický výklad. Nicméně alespoň základní povědomí o původu současné kultury svobodného softwaru je pro to, abychom mohli poskytnout nějaké praktické rady, zcela nezbytné. Člověk, který této kultuře rozumí, se ve světě open source může dostat na lecjaká místa, kde sice najde řadu místních odlišností ve zvycích a v dialektu, ale kde se přesto bude moct bez větších problémů a efektivně zapojit do spolupráce. Naproti tomu člověku, jenž tuto kulturu nechápe, bude připadat proces organizace projektu nebo účasti na něm obtížný a plný překvapení. Protože počet lidí vyvíjejících svobodný software stále prudce roste, mnoho z nich spadá do té druhé kategorie; celou kulturu zatím do velké míry tvoří čerství přistěhovalci a tenhle stav ještě nějakou dobu potrvá. Pokud si myslíte, že byste mohli být jedním z nich, poskytne vám další podkapitola základní informace pro pochopení diskuzí, s nimiž se setkáte později – jak v této knize, tak na internetu. (Na druhou stranu pokud v open source už nějakou dobu pracujete, je možné, že už o jeho historii víte dost, takže následující oddíl klidně přeskočte.)

Historie

Sdílení softwaru je tak staré jako software sám. V době, kdy počítače začínaly, byli výrobci přesvědčeni, že jejich konkurenční výhoda spočívá hlavně v inovaci hardwaru, a možnostem, jak zpeněžit software, proto nevěnovali příliš velkou pozornost. Většina lidí, kteří tyto první počítače kupovali a používali, byli vědci nebo technici, kteří byli schopni software dodávaný spolu s počítači upravovat a rozšiřovat sami. Tito zákazníci někdy nejen že posílali své úpravy zpět k výrobci, ale nabízeli je i ostatním vlastníkům podobných počítačů. Výrobci to často tolerovali a někdy k tomu dokonce i vybízeli – vylepšení softwaru, ať už ho provedl kdokoliv, činilo jejich počítače přitažlivějšími pro další potenciální zákazníky.

Ačkoliv toto rané období současnou kulturu svobodného softwaru v mnohém připomínalo, lišilo se od ní ve dvou zásadních ohledech. Zaprvé byl hardware dosud jen velmi málo standardizován – byla to doba mnoha zářných inovací v návrhu počítačů, ale tato rozmanitost výpočetních architektur znamenala, že všechno bylo nekompatibilní se vším. Takže software napsaný pro jeden počítač obvykle nefungoval na druhém. Programátoři se obvykle stávali odborníky na konkrétní architekturu nebo rodinu architektur (zatímco dnes by se spíš zdokonalovali v programovacím jazyce nebo v rodině jazyků a spoléhali by na to, že jejich odborné znalosti budou přenositelné na jakýkoliv počítačový hardware, se kterým kdy budou pracovat). Protože obvykle uměli opravdu dobře pracovat jen s určitým typem počítače, měla tato jejich specializace za následek to, že byl tento počítač pro ně a pro jejich kolegy přitažlivější. Bylo tedy v zájmu výrobců, aby se co nejvíce šířil kód závislý na jejich konkrétním produktu.

Zadruhé ještě neexistoval internet. Ačkoliv právních předpisů omezujících sdílení dat bylo mnohem méně než dnes, existovalo nesrovnatelně více technických překážek: prostředky pro přenos dat z místa na místo byly poměrně složité a nepraktické. Existovaly sice malé lokální sítě, které se daly používat pro sdílení informací mezi zaměstnanci ve stejné výzkumné laboratoři nebo ve firmě, ale pokud jste chtěli něco sdílet se všemi bez ohledu na to, kde se nacházejí, narazili jste na značné potíže. V mnoha případech lidé našli způsoby, jak tyto bariéry překonat. Někdy se různé skupiny navzájem kontaktovaly a posílaly si pak disky nebo pásky poštou, někdy fungovali jako středisko pro výměnu úprav samotní výrobci. Věci pomohlo i to, že v době prvních počítačů mnozí vývojáři pracovali na univerzitách, kde se očekávalo, že budou své znalosti publikovat. Ale způsoby, jimiž přenos dat fyzicky probíhal, znamenaly, že proti sdílení působil značný odpor, přímo úměrný vzdálenosti (ať už geografické nebo organizační), kterou musel software překonat. Celosvětové, okamžité sdílení dat, jaké známe dnes, nebylo jednoduše možné.

Vzestup proprietárního softwaru a svobodného softwaru

Jak počítačový průmysl dozrával, došlo k několika vzájemně souvisejícím změnám. Divoká rozmanitost hardwarových návrhů byla postupně vytlačena několika jasnými vítězi – ať už za jejich vítězství mohla lepší technologie, lepší marketing nebo obojí dohromady. Zároveň, a to nejen shodou okolností, se začaly vyvíjet takzvané „vyšší programovací jazyky“, což znamenalo, že bylo možné napsat

program jednou, v jednom jazyce, a nechat jej automaticky přeložit („zkompilovat“) tak, aby mohl běžet na různých typech počítačů. Výrobci hardwaru si dobře uvědomili, jaké důsledky bude tento vývoj mít: jejich zákazníci se teď mohli pouštět i do velkých softwarových projektů, aniž by se museli nezbytně vázat na konkrétní počítačovou architekturu. Jak byly z trhu postupně vytlačovány méně efektivní návrhy, začaly se snižovat i výkonnostní rozdíly mezi různými počítači, což znamenalo, že těm, kteří by obchodovali pouze s hardwarem, by začaly v blízké budoucnosti výrazně klesat zisky. Z hrubé výpočetní síly se stalo zaměnitelné zboží; jazýčkem na vahách se stal software. Prodávání softwaru, nebo přinejmenším jeho akceptování jako nedílné součásti prodeje hardwaru, se najednou stalo dobrou strategií.

To znamenalo, že výrobci museli začít dbát na přísnější dodržování autorských práv spojených se svými programy. Pokud by je uživatelé mezi sebou dál neomezeně sdíleli a upravovali, mohli by nezávisle implementovat některá zlepšení, která dodavatel začal prodávat jako „přidanou hodnotu“. A co by bylo ještě horší, sdílený zdrojový kód by se mohl dostat do rukou konkurence. Ironií všeho bylo, že se toto vše dělo přibližně ve stejné době, kdy se začal šířit internet. Právě v době, kdy začalo být neomezené sdílení softwaru konečně technicky možné, se stalo následkem změn v celém počítačovém průmyslu ekonomicky nežádoucím – přinejmenším z pohledu jednotlivých firem. Dodavatelé přitvrdili buď tím, že uživatelům odmítali přístup ke zdrojovému kódu, který na jejich počítačích běžel, nebo tím, že trvali na podepsání dohod o zachování důvěrnosti, které sdílení zamezovaly.

Vědomé hnutí odporu

Ve stejné době, kdy začal svět neomezované výměny kódu zvolna mizet, začala v hlavě přinejmenším jednoho programátora uzrávat myšlenka, jak se tomu vzepřít. Richard Stallman pracoval v sedmdesátých a raných osmdesátých letech minulého století v Laboratoři umělé inteligence (Artificial Intelligence Lab) na Massachusetts Institute of Technology – tedy při zpětném pohledu přímo uprostřed zlatého věku a v ideálním prostředí pro sdílení programů. V Laboratoři umělé inteligence se vyznávala přísná „hackerská etika“[3] a lidé zde nejen že byli vybízeni, aby se o svá zlepšení systému podělili s ostatními, ale dokonce se to od nich očekávalo. Jak Stallman později napsal:

> *Našemu softwaru jsme neříkali „svobodný software“, protože tento pojem ještě neexistoval, ale v zásadě to bylo totéž. Kdykoliv si chtěl někdo z jiné univerzity nebo nějaké firmy zkopírovat náš program, aby jej mohl u sebe používat, s radostí jsme mu ho poskytli. Pokud jste viděli, že někdo používá nějaký vám neznámý a zajímavý program, vždycky jste mohli požádat o zdrojový kód. Mohli jste si jej prostudovat, změnit jej, nebo z něj vykousnout nějaké části a použít je pro nový program.*
>
> (citováno z **http://www.gnu.org/gnu/thegnuproject.html**)

[3] Stallman používá slovo „hacker“ ve smyslu „někdo, kdo rád programuje a koho těší, že je v tom dobrý“, a ne v relativně novém významu „někdo, kdo se nabourává do počítačů“.

Změny, které se děly ve zbytku počítačového průmyslu, ale nakonec nedlouho po roce 1980 dostihly i Laboratoř umělé inteligence a celá rajská komunita kolem Stallmana se rozpadla. Jedna začínající společnost přetáhla z laboratoře mnoho programátorů, aby u ní pracovali na vývoji operačního systému. Ten se podobal tomu, na němž pracovali v laboratoři, ale měl velmi přísnou licenci. Ve stejné době Laboratoř umělé inteligence samotná získala nové zařízení, které bylo dodáno s proprietárním operačním systémem.

Stallman v tom, co se dělo, začal vnímat širší souvislosti:

> *Moderní počítače té doby, jako byl VAX nebo 68020, měly své vlastní operační systémy, ale žádný z nich nebyl svobodným softwarem – předtím, než jste mohli získat pouhou spustitelnou kopii programu, jste museli podepsat dohodu o důvěrnosti.*

To znamená, že prvním krokem při používání počítače byl slib, že nebudete ostatním pomáhat. Jakákoliv spolupráce v rámci komunity byla zakázána. Pravidlo, jež vlastníci proprietárního softwaru vyhlásili, znělo: „Pokud sdílíte s někým jiným, jste pirát. Pokud chcete nějaké změny, požádejte nás, abychom je provedli."

Něco v Stallmanovi se tomu ale vzpíralo; rozhodl se proto celému trendu postavit. Místo toho, aby pokračoval v práci v teď už téměř nefunkční Laboratoři umělé inteligence nebo aby přijal místo programátora v jedné z nových firem, kde by výsledky jeho práce ležely zamčené v trezoru, z laboratoře odešel a založil Projekt GNU a Free Software Foundation (FSF; Nadace svobodného softwaru). Cílem GNU[4] bylo vyvinout zcela svobodný a otevřený počítačový operační systém a balík aplikačního softwaru, v němž uživatelům nikdy nebude bráněno cokoliv měnit nebo své úpravy šířit. V podstatě začal Stallman znovu vytvářet to, co bylo v Laboratoři umělé inteligence zničeno, ale tentokrát v celosvětovém měřítku a bez slabin, kvůli nimž se kultura laboratoře nakonec rozpadla.

Kromě prací na novém operačním systému Stallman vymyslel autorskou licenci, jejíž znění zaručovalo, že jeho kód zůstane trvale svobodný. Celá GNU General Public License (GPL) je velmi rafinovaný právnický výkon. Říká, že kód může být kopírován a upravován bez omezení a že jak kopie, tak odvozená díla (tedy upravené verze) musí být distribuovány se stejnou licencí jako originál, bez jakýchkoliv dalších omezení. Používá tedy zákon o autorském právu tak, aby dosáhl zcela opačného efektu, než tradiční copyright má mít – místo toho, aby distribuci softwaru nějak omezoval, zabraňuje všem, dokonce i svému autorovi samotnému, aby jakékoliv omezení ustanovil. Pro Stallmana byl tento postup lepší, než kdyby jednoduše prohlásil svůj kód za volně šiřitelný. V takovém případě by totiž mohla být jakákoliv jeho kopie začleněna do proprietárního programu (což se u programů s tolerantními autorskými licencemi stávalo). I když by takové zabudování žádným způsobem nesnížilo dostupnost původního kódu, znamenalo by to, že by ze Stallmanova úsilí mohl těžit jeho nepřítel – proprietární software. O GPL lze uvažovat jako o jisté formě protekcionismu vztahující se na svobodný software,

[4] Název projektu je zkratka z „GNU's Not Unix" („GNU není Unix"), přičemž „GNU" v této delší verzi je zkratka pro… přesně to samé.

protože nesvobodnému softwaru zabraňuje naplno využít výhod kódu vydaného pod GPL. Na licenci GPL a její vztah k ostatním licencím svobodného softwaru se podíváme podrobněji v kapitole **9. Licence, autorská práva a patenty.**

S pomocí mnoha programátorů, z nichž někteří Stallmanovu ideologii sdíleli a někteří jednoduše chtěli, aby existovalo velké množství dostupného svobodného kódu, začal projekt GNU vydávat svobodné náhrady pro mnoho z těch nejdůležitějších komponent operačního systému. Vzhledem k tomu, že standardizace počítačového hardwaru i softwaru v té době už značně pokročila, bylo možné tyto náhrady z projektu GNU používat i v systémech, jež jinak svobodné nebyly, což také mnoho lidí dělalo. Zvlášť úspěšnými výstupy projektu GNU byly jeho textový editor Emacs a kompilátor jazyka C (GCC), které si získaly velkou a oddanou skupinu následovníků ne na základě ideologie, ale pro jejich technické kvality. Kolem roku 1990 GNU vytvořil již většinu komponent svobodného operačního systému s výjimkou jádra – tedy té části, kterou počítač zavádí při startu (boot) a která je zodpovědná za správu paměti, diskového prostoru a dalších systémových zdrojů.

Naneštěstí si projekt GNU zvolil návrh jádra, který se z hlediska implementace ukázal obtížnější, než se očekávalo. Celý vývoj se začal zpožďovat, a kvůli tomu to tedy nakonec nebyla Free Software Foundation, kdo vytvořil první zcela svobodný operační systém. Poslední chybějící kus skládanky místo nich dodal Linus Torvalds, finský student informatiky, který s pomocí dobrovolníků z celého světa dokončil svobodné jádro vycházející z konzervativnějšího návrhu. Dal mu název Linux; poté, co bylo toto jádro zkombinováno s existujícími programy GNU, vznikl zcela svobodný operační systém. Poprvé v historii jste mohli nastartovat počítač a pracovat bez použití jakéhokoliv proprietárního softwaru.[5]

Velká část softwaru tohoto nového operačního systému nepocházela z projektu GNU. Ve skutečnosti dokonce GNU nebyla jedinou skupinou, která by se snažila vytvořit svobodný operační systém – ve stejné době už byl například vyvíjen kód, z nějž se časem staly systémy NetBSD a FreeBSD. Význam Free Software Foundation nespočívá jen v tom, že psali programy, ale také v jejich politické rétorice. Tím, že svobodný software brali jako svůj hlavní program a nejen jako něco, co může být celkem užitečné, bylo pro programátory obtížné si na celou věc neudělat nějaký názor. Dokonce i ti, kteří s FSF nesouhlasili, se celým problémem museli zabývat, už jen proto, aby mohli vyjádřit svůj odlišný názor. Úspěch FSF jako šiřitelů propagandy spočívá v tom, že svůj zdrojový kód svázali prostřednictvím GPL a dalších textů i s jasným politickým poselstvím. Jak se jejich programy šířily dál, šířilo se s nimi i toto poselství.

---

[5] Technicky vzato nebyl Linux úplně první. Nedlouho před Linuxem se objevil jiný svobodný operační systém pro počítače kompatibilní s IBM; nazýval se 386BSD. Tento systém ale bylo mnohem obtížnější spustit a udržet v provozu. Význam Linuxu spočíval nejen v tom, že byl svobodný, ale také v tom, že u něj byla celkem vysoká pravděpodobnost, že se na vašem počítači po instalaci skutečně i rozběhne.

Neúmyslné hnutí odporu

Na rodící se scéně svobodného softwaru se děla spousta dalších věcí, ale málo z nich bylo tak výrazně ideologických jako Stallmanův projekt GNU. Jedním z nejdůležitějších byl systém Berkeley Software Distribution (BSD) vytvořený programátory University of California v Berkeley jako postupná reimplementace operačního systému Unix, jenž byl až do konce 70. let víceméně proprietárním výzkumným projektem společnosti AT&T. Skupina BSD nečinila žádná otevřeně politická prohlášení o tom, že se programátoři musí sjednotit a vzájemně sdílet svou práci, ale v praxi tuto myšlenku realizovala, a to s velkým nadšením, při koordinaci celého masivního distribuovaného vývoje; v rámci projektu byly od základů přepsány unixové programy pro příkazový řádek, jeho knihovny a nakonec i samotné jádro operačního systému, a to převážně díky dobrovolníkům. Projekt BSD se stal ukázkovým příkladem neideologického vývoje svobodného softwaru a byl také místem, kde načerpala své první zkušenosti řada vývojářů, kteří pak zůstali v open source světě aktivní.

Další zatěžkávací zkouškou kooperativního vývoje byl X Window System, svobodné, síťově transparentní grafické výpočetní prostředí vyvinuté v polovině 80. let na MIT ve spolupráci s prodejci hardwaru, kteří měli společný zájem na tom, aby byli svým zákazníkům schopni nabídnout systém pracující s okny. Zdaleka nešlo o odpor vůči proprietárnímu softwaru – licence pro X dokonce záměrně dovolovala proprietární rozšiřování svobodného jádra systému, protože každý člen celého sdružení chtěl mít možnost základní distribuci X vylepšit, a získat tím konkurenční výhodu oproti těm ostatním. Samotné X Windows[6] byly sice svobodným softwarem, ale především proto, aby existovaly rovné podmínky pro vzájemný souboj obchodních zájmů, a ne ve snaze ukončit převahu proprietárního softwaru. Dalším příkladem, který projekt GNU o pár let předběhl, byl TeX Donalda Knuthe, svobodný systém pro sázení dokumentů, jenž kvalitou svých výstupů odpovídal požadavkům nakladatelství. TeX byl vydán s licencí, která komukoliv umožňovala kód upravovat a šířit, ale výsledek těchto úprav se nesměl nazývat „TeX“, pokud nesplnil velmi přísnou sadu testů kompatibility (je to tedy příklad svobodné licence „chránící obchodní značku“; tento typ probereme podrobněji v kapitole **9. Licence, autorská práva a patenty**). Knuth se vůbec nesnažil zaujmout nějaké stanovisko v debatě svobodný versus proprietární software. Potřeboval zkrátka lepší sázecí systém, aby mohl dokončit to, co bylo jeho primárním cílem, totiž kniha o počítačovém programování, a neviděl žádný důvod, proč by svůj systém neměl po dokončení zveřejnit.

Aniž bychom jmenovali každý projekt a každou licenci, můžeme bezpečně říct, že na konci 80. let již existovalo velké množství svobodného softwaru s celou řadou různých licencí. Rozmanitost těchto licencí odpovídala tomu, jak různorodé byly i motivace těchto projektů. Dokonce i někteří z programátorů, kteří si zvolili GNU GPL, byli ideologicky mnohem méně vyhranění než projekt GNU samotný. Třebaže mnoho z těchto vývojářů práce na svobodném softwaru bavila, nechápali proprietární software jako společenské zlo. Našli se sice tací, kteří považovali za svou morální povinnost zbavit svět „softwarového hamounění“ („software hoarding“ – Stallmanův pojem pro nesvobodný software), ale existovali

[6] Projekt sám má raději název „X Window System“, ale v praxi se obvykle používá „X Windows“, protože to je kratší.

i jiní, které pohánělo jejich technické zaujetí nebo radost ze spolupráce s podobně smýšlejícími lidmi, či dokonce obyčejná lidská touha po slávě. Přesto ale obvykle nedocházelo k tomu, že by se tyto nesourodé motivace nějakým způsobem tloukly. To je dáno částečně tím, že na rozdíl od ostatních tvůrčích forem, jako je literatura nebo výtvarné umění, musí software k tomu, aby mohl být považován za úspěšný, splnit několik částečně objektivních testů: musí fungovat a být do rozumné míry prostý chyb. To všem účastníkům projektu automaticky dává jakýsi společný základ, důvod a rámec pro vzájemnou spolupráci, aniž by se museli příliš starat o jinou než technickou způsobilost.

K tomu, aby vývojáři drželi pospolu, pak měli ještě další důvod – ukázalo se totiž, že ve světě svobodného softwaru vzniká někdy zdrojový kód velmi vysoké kvality. V některých případech byly tyto programy prokazatelně technicky dokonalejší než jejich nejbližší proprietární alternativa, zatímco v dalších byly přinejmenším srovnatelné; vždy ale byly samozřejmě levnější. Pro pár lidí mohla mít motivace k používání svobodného softwaru čistě filozofický základ, ale většina lidí jej používala prostě proto, že byl lepší. A z těch, kteří ho používali, bylo vždy určité procento ochotné věnovat svůj čas a schopnosti tomu, aby jej pomohli udržovat a zlepšovat.

Tendence psát dobré programy se rozhodně neprojevovala všude, ale u projektů svobodného softwaru celého světa se dala pozorovat stále častěji. Toho si postupně začaly všímat firmy, které byly na svém softwaru výrazně závislé. Mnohé z nich zjistily, že svobodný software už při své každodenní práci používají a neví o tom – protože vedení společnosti nemusí vždy tušit, co přesně jejich IT oddělení dělá. Různé společnosti začaly k projektům svobodného softwaru i na veřejnosti zaujímat stále aktivnější postoj. Dávaly jim k dispozici svůj čas a své vybavení a v některých případech dokonce vývoj svobodných programů přímo financovaly. Podobné investice se jim totiž v nejlepších případech mohou i několikanásobně vrátit. Sponzor platí pouze malé množství odborníků, kteří se díky tomu mohou projektu naplno věnovat; přitom ale sklízí plody práce všech zúčastněných, tedy včetně neplacených dobrovolníků a programátorů, kteří jsou placeni jinými organizacemi.

„Free“ versus „open source“

S tím, jak začaly společnosti světa svobodnému softwaru věnovat čím dál tím větší pozornost, vyvstaly před programátory nové problémy – tentokrát s prezentací. Jedním z nich bylo samotné slovo „free“, které v angličtině může znamenat „svobodný“, ale také „zdarma“. Když lidé slyší pojem „free software“ poprvé, mnoho z nich se mylně domnívá, že to znamená „software zadarmo“. Pravda je taková, že veškerý svobodný software je sice dostupný zdarma,[7] ale to, že je nějaký software zadarmo, ještě neznamená, že je svobodný. Například během války prohlížečů v 90. letech poskytly jak Netscape, tak Microsoft při rvačce o získání podílu na trhu své konkurenční webové prohlížeče veřejnosti zdarma. Ani jeden z nich však nebyl svobodný software. Nebylo možné získat jejich

[7] Za šíření kopií svobodného softwaru si sice můžete účtovat nějaký poplatek, ale protože nelze zabránit tomu, aby příjemce pokračoval v dalším šíření zdarma, klesá vaše cena prakticky okamžitě k nule.

zdrojový kód; i kdyby se vám to podařilo, neměli jste právo jej upravovat a šířit dál.[8] Jedinou věcí, kterou jste mohli udělat, bylo stáhnout a spustit zkompilovaný program. Tyto prohlížeče nebyly o nic víc svobodné než software, jenž jste si koupili v obchodě v krabici – jenom byly levnější.

Tento zmatek kolem slova „free" je způsoben výhradně jeho nešťastnou dvojznačností v anglickém jazyce. Většina jiných jazyků rozlišuje nízkou cenu od svobody (například mluvčím románských jazyků je ihned jasný rozdíl mezi *gratis* a *libre*). Ale protože je angličtina de facto hlavním komunikačním jazykem internetu, znamená to, že její problémy jsou do jisté míry problémy všech. Nedorozumění kolem slova „free" bylo tak časté, že programátoři svobodného softwaru časem vytvořili tuto standardní formulku: „Je to *free* jako *freedom* (svoboda) – tedy ve smyslu *free speech* (svoboda slova) a ne *free beer* (pivo zdarma)." To ale nic nemění na tom, že je poněkud unavující to pořád dokola vysvětlovat. Mnozí programátoři měli pocit – a to ne bezdůvodný –, že nejednoznačnost slova „free" brání tomu, aby veřejnost pochopila, o jaký software jde.

Ale ukázalo se, že problém je ještě závažnější. Slovo „free" totiž nese ještě další význam, tentokrát morální: pokud měla být cílem svoboda jako taková, pak nezáleželo na tom, zda byl shodou okolností tento svobodný software také lepší nebo pro nějaký podnik ekonomicky výhodnější. To byl jen příjemný vedlejší účinek; primární motivace zde ale nebyla v jádru ani technická, ani ekonomická, ale filozofická. Zastávání svobody softwaru navíc poukazovalo na to, že se často objevuje značný rozpor u společností, které sice v rámci své obchodní činnosti chtějí podporovat konkrétní svobodné programy, ale jinak dál pokračují v nabízení proprietárního softwaru.

A tak stálo před komunitou, která už tak mířila ke krizi identity, další dilema. Sami programátoři, kteří svobodný software píší, se nikdy neshodli na tom, jaký cíl tohoto hnutí vlastně je, pokud tedy vůbec nějaký existuje. Bylo by dokonce zavádějící tvrdit, že se zde objevuje celá škála názorů od jednoho extrémního postoje k druhému, protože taková lineární osa tu jednoduše neexistuje; místo toho panuje obrovská roztříštěnost názorů na mnoho dílčích aspektů věci. Pokud ale na chvíli odhlédneme od detailů, uvidíme zde v zásadě dvě velké skupiny uvažujících. Jedna z nich zastává Stallmanovo stanovisko, podle nějž je hlavním cílem svoboda sdílet a upravovat; pokud přestanete mluvit o svobodě, znamená to, že jste zapomněli na to nejdůležitější. Jiní mají pocit, že nejdůležitějším argumentem, který pro jakýkoliv software hovoří, je tento software samotný, a odmítají prohlásit, že veškerý proprietární software je ze své podstaty špatný. Někteří, ale ne všichni programátoři svobodného softwaru věří, že právo na to, určit podmínky pro distribuci softwaru, by měl mít jeho autor (nebo v případě placené práce jeho zaměstnavatel) a že volba licence nemusí nutně poukazovat na nějaké morální stanovisko.

Tyto rozdíly nebylo dlouho nutné nijak zvlášť zkoumat nebo se k nim vyjadřovat, ale rostoucí úspěch svobodného softwaru v podnikatelské sféře si to časem vynutil. V roce 1998 byl jako alternativa k termínu „svobodný software" (free software) vytvořen pojem *open source*. Zasloužila se o to skupina

[8] Zdrojový kód Netscape Navigatoru byl nakonec v roce 1998 přece jen zveřejněn s open source licencí a stal se základem webového prohlížeče sdružení Mozilla. Viz **http://www.mozilla.org/**.

programátorů, kteří časem vytvořili The Open Source Initiative (Iniciativa open source, OSI).[9] Členové OSI měli pocit, že to není jen spojení „free software“, které způsobuje potíže, ale že zde existuje větší problém, jehož je slovo „free“ jako takové jen symptomem – totiž že celé hnutí potřebuje marketingový program, který by se dal představit světu obchodních společností, protože hovory o tom, jaké jsou morální a společenské výhody sdílení, na obchodních jednáních nikdo poslouchat nebude. Sdružení OSI celou věc vyjádřilo takto:

> *The Open Source Initiative je marketingový program pro svobodný software. Je to způsob, jak propagovat celou myšlenku svobodného softwaru postaveného na pevných pragmatických základech a ne na ideologickém zápalu. Podstata, a tedy i největší síla našeho softwaru se nemění; co se mění, je jeho symbolický rozměr a celý poraženecký přístup, který tato komunita má. …*
>
> *Většinu technicky založených lidí není nutné přesvědčovat o výhodách konceptu open source, ale o jeho jméně. Proč bychom neměli dál používat zavedený termín „svobodný software“?*
>
> *Jedním z důvodů je to, že pojem „free software“ lze snadno chápat nesprávně, což často vede ke konfliktům. …*
>
> *Ale tím hlavním, co nás k této změně označení vede, je marketing. Svou koncepci chceme nabídnout obchodnímu světu. Máme produkt, s nímž lze zvítězit, ale způsob jeho uvádění na trh byl dosud zcela špatný. Pojem „free software“ byl v podnikatelské sféře špatně pochopen; touha sdílet byla brána jako boj proti kapitalismu nebo ještě hůře jako krádež.*
>
> *Ředitelé velkých obchodních společností a jejich technologických oddělení na koncept „svobodného softwaru“ nikdy slyšet nebudou. Ale co když vezmeme stejné tradice, stejné lidi, stejné licence, které známe jako „svobodný software“, a změníme nálepku na „open source“? To už je něco, o čem se budou ochotni bavit.*
> *Někteří hackeři tomuhle odmítají uvěřit, ale to je dané tím, že jsou to technicky orientovaní lidé, kteří uvažují v konkrétních a reálných pojmech; nedokážou pochopit, jak je při prodeji čehokoliv důležitý celkový dojem, vaše image.*
>
> *Ve světě marketingu jste tím, čím vypadáte, že jste. Pokud vyvoláme dojem, že jsme ochotni slézt z barikád a spolupracovat se světem podnikání, bude to zrovna tak důležité jako naše skutečné chování, naše zásady a náš software.*

(citováno z **http://www.opensource.org/**. Přesněji řečeno z dřívější podoby těchto stránek. OSI je od té doby zřejmě stáhla, ale stále je lze najít na **http://web.archive.org/web/20021204155057/http://www.opensource.org/advocacy/faq.php** a **http://web.archive.org/web/20021204155022/http://www.opensource.org/advocacy/case_for_hackers.php#marketing.**)

---

[9] Domácí stránka OSI se nachází na **http://www.opensource.org/**.

V tomto textu se objevují náznaky mnoha sporných otázek. Zmiňuje se o „našich zásadách“, ale chytře se vyhýbá tomu říct, o jaké zásady přesně jde. Pro některé se může jednat o přesvědčení, že programy vyvinuté v rámci otevřeného procesu budou lepší; jiní těmito zásadami budou rozumět princip, že by se všechny informace měly sdílet. Používá se zde slovo „krádež“ (patrně) ve smyslu nelegálního kopírování – proti čemuž mnozí namítají, že pokud přitom původní vlastník o nic nepřišel, nemůže už z principu jít o krádež. Naznačuje se zde, že by hnutí za svobodný software mohlo být vnímáno jako antikapitalismus, ale tomu, zda je takové obvinění ve skutečnosti něčím podloženo, se pečlivě vyhýbá.

Tím ale nechci říct, že by byly stránky OSI nekonzistentní nebo úmyslně zavádějící. To rozhodně nejsou. Spíše jde o skvělý příklad toho, co podle OSI v hnutí za svobodný software chybělo, tedy dobrého marketingu; „dobrý“ zde znamená „životaschopný v obchodním světě“. Iniciativa open source tak dala řadě lidí přesně to, co hledali – způsob, jak mluvit o svobodném softwaru jako o metodologii vývoje a obchodní strategii a ne jako o filozofickém hnutí.

Vznik Iniciativy změnil celou situaci svobodného softwaru. OSI zde podala definici rozporu, který dlouho zůstával nepojmenován, čímž celé hnutí donutila připustit, že se nestačí zabývat politikou jen navenek, ale také uvnitř. Následkem toho je, že obě strany musely najít společnou řeč, protože většina projektů zahrnuje programátory z obou táborů i účastníky, které do žádné kategorie jednoznačně zařadit nelze. To neznamená, že by přestali mluvit o svých morálních zásadách – pokud se například někdo dopustí prohřešku vůči tradiční „hackerské etice“, bývá mu to někdy vyčteno. Jen málokdy ale nějaký vývojář svobodného nebo open source softwaru zpochybní motivaci ostatních účastníků projektu. Příspěvek je důležitější než jeho autor. Pokud někdo píše dobrý zdrojový kód, neptáte se ho, zda to dělá z morálních důvodů, zda jej za to někdo platí, zda si vylepšuje svůj profesní životopis nebo cokoliv jiného. Jeho příspěvek hodnotíte z technického hlediska a reagujete na technickém základě. Dokonce i explicitně politické organizace, jako je projekt Debian, jenž si dal za cíl nabídnout stoprocentně svobodné výpočetní prostředí, se k integraci nesvobodného kódu staví celkem otevřeně a spolupracují s programátory, kteří stejné cíle nesdílejí.

Současná situace

Při řízení projektu svobodného softwaru není potřeba řešit tyto závažné filozofické záležitosti každý den. Programátoři nebudou trvat na tom, aby všichni ostatní účastníci projektu plně souhlasili s jejich pohledem na věc – a ti, kteří na tom trvat budou, rychle zjistí, že pak nemůžou pracovat prakticky nikde. Musíte si ale být vědomi toho, že debata „svobodný“ versus „open source“ existuje, a to jak proto, abyste nechtěně neřekli něco, co by někteří účastníci mohli vnímat jako nepřátelské, tak proto, že pochopení motivace vašich vývojářů je tím nejlepším a do určité míry vlastně i jediným způsobem, jak projekt řídit.

Rozhodnutí se pro svobodný software je volbou každého jednotlivce. Abyste byli v této kultuře úspěšní, musíte pochopit, proč lidé tuto volbu učinili. Donucovací techniky zde nefungují. Pokud se vývojáři v jednom projektu necítí dobře, pak jednoduše odejdou jinam. Svobodný software je výjimečný i mezi jinými skupinami dobrovolníků, protože jejich osobní investice je zde jen velmi malá. Většina lidí zapo-

jených v projektu se s těmi ostatními nikdy osobně nesetká; projektu věnují kousky svého času zkrátka tehdy, když se jim chce. Běžné způsoby, kterými se lidé dávají dohromady a vytvářejí trvalé skupiny, jsou omezeny jen na uzounký kanál – psané slovo přenášené elektronicky. Z tohoto důvodu může trvat poměrně dlouho, než vznikne skupina, která opravdu drží pohromadě a jde za stejným cílem. A naopak je velmi snadné, aby projekt ztratil potenciálního dobrovolníka už během prvních pěti minut. Pokud nebude jeho první dojem z projektu dobrý, pak mu jen málokdy dá i druhou šanci.

Pomíjivost nebo přesněji řečeno potenciální pomíjivost vztahů je možná tím vůbec nejtěžším problémem, kterému musí nový projekt čelit. Jak všechny tyto lidi přesvědčit, aby zůstali pohromadě dost dlouho na to, aby vzniklo něco užitečného? Odpověď na tuto otázku je natolik složitá, že zabere prakticky celý zbytek této knihy. Pokud bychom ale měli odpovědět jednou větou, vypadala by asi takto:

> *Lidé by měli mít pocit, že míra jejich zapojení do projektu a jejich vliv na projekt jsou přímo úměrné jejich přínosu.*

Žádná skupina vývojářů (nebo potenciálních vývojářů) by neměla mít pocit méněcennosti nebo diskriminace z jiných než technických důvodů. Je jasné, že projekty sponzorované firmami nebo takové, v nichž existují i placení vývojáři, musí být v tomto ohledu zvlášť opatrné, jak podrobněji popíšeme v kapitole **5. Peníze**. To ale samozřejmě neznamená, že pokud není projekt sponzorovaný žádnou firmou, není se čeho bát. Peníze jsou jen jedním z mnoha faktorů, které mohou úspěšnost projektu ovlivnit. Jsou tu i další otázky: jaký programovací jazyk, jakou licenci a jaký vývojový proces bude nejlepší, jakou infrastrukturu je potřeba vytvořit, jak efektivně oznámit zahájení projektu a mnoho dalších. Tomu, jak při startu projektu vykročit tou správnou nohou, se věnuje následující kapitola.

2. Zahájení projektu

2. Zahájení projektu

Klasický model toho, jak projekty svobodného softwaru začínají, popsal Eric Raymond ve svém proslulém eseji o procesech používaných v open source, který má název *The Cathedral and the Bazaar* (Katedrála a bazar). Napsal:

> *Veškerý dobrý software začal tak, že se nějaký vývojář potřeboval podrbat tam, kde jej něco svědilo.*
>
> (citováno z **http://www.catb.org/~esr/writings/cathedral-bazaar/**)

Povšimněte si, že Raymond neřekl, že open source projekty vznikají jen tehdy, když někoho něco svědí. Řekl něco trochu jiného: že dobrý software vzniká tehdy, když má programátor osobní zájem na tom, aby byl nějaký problém vyřešen; ukázalo se, že právě to bývá skutečně i tou nejčastější motivací pro založení projektu svobodného softwaru.

Pro většinu z nich to platí dodnes, ale už jich není tolik jako v roce 1997, kdy Raymond článek napsal. V současnosti můžeme pozorovat fenomén, kdy řada organizací – včetně klasických společností orientovaných na zisk – spouští velké, centrálně řízené open source projekty vyvíjené od základů. Zdrojem značné části svobodného softwaru nadále zůstává osamělý programátor, který napíše kód, jenž řeší jeho vlastní problém, a pak si uvědomí, že se výsledek jeho práce dá použít i jinde, ale existují i jiné varianty.

Raymondův postřeh nicméně stále platí. Základní podmínkou je, aby tvůrci softwaru měli přímý zájem na jeho úspěchu, protože jej sami používají. Pokud software nedělá to, co by měl, pak osoba, popřípadě organizace, která jej vytváří, bude při své každodenní práci nespokojen. Vezměme si například projekt OpenAdapter (**http://www.openadapter.org/**), který spustila investiční banka Dresdner Kleinwort Wasserstein jako open source framework pro integraci několika různých finančních informačních systémů; jen těžko bychom mohli tvrdit, že tento projekt řešil problém nějakého jednotlivého programátora. Byla to instituce, která něco potřebovala vyřešit. Tento problém ale přímo vyplýval ze zkušeností této instituce samotné a jejích partnerů, takže pokud by projekt nesplnil svůj úkol, nepochybně si toho všimnou. V takovémto uspořádání vzniká dobrý software, protože zpětná vazba míří tím správným směrem. Není to program, který by byl napsán proto, aby mohl být prodáván někomu jinému a řešil cizí problémy. Byl napsán jako reakce na řešení něčího problému a pak sdílen s ostatními, jako kdyby tento problém byla nějaká nakažlivá nemoc a software byl lékem šířeným proto, aby zabránil epidemii.

Tato kapitola se zabývá tím, jak do světa uvést nový projekt svobodného softwaru, ale mnohá z jejích doporučení by zněla velmi povědomě zdravotnickým organizacím distribuujícím léky. Tyto dva cíle jsou totiž velmi podobné. Chcete-li, aby bylo jasné, k čemu je lék dobrý, dejte jej do rukou těch správných lidí a ujistěte se, že jej budou umět správně použít. Ale u softwaru chcete navíc přilákat některé z příjemců k tomu, aby se k probíhajícímu výzkumu, jenž má lék vylepšit, sami připojili.

Proces šíření svobodného softwaru má dva aspekty. Software potřebuje získat jak nové uživatele, tak vývojáře. Tyto dva požadavky sice nejsou protichůdné, ale zvyšují složitost počáteční prezentace projektu. Některé informace jsou užitečné pro obě skupiny, některé ale jen pro jednu nebo pro druhou. V obou případech by měly být sdělovány na základě principu odstupňované prezentace. To znamená, že v každé etapě by měla míra uváděných podrobností odpovídat množství času a úsilí, které musí čtenář vynaložit. Čím více úsilí vynaloží, tím více informací by měl získat. Pokud zmíněné dvě oblasti nebudou úzce sladěny, můžete u zájemců rychle ztratit důvěru, takže se raději přestanou snažit úplně.

Z toho logicky vyplývá, že na vzhledu záleží. To je něco, čemu zejména programátoři často nechtějí věřit. Povyšování obsahu nad formu je u nich bezmála profesionální ctí. Není žádná náhoda, že mnozí programátoři mají odpor k marketingu a k práci s veřejností; zrovna tak není náhoda, že profesionální grafičtí návrháři jsou často tím, co programátoři sami vytvoří, upřímně zděšeni.

Je to škoda, protože existují situace, kdy právě forma je podstatou, a prezentace projektu je jejich součástí. Například jednou z prvních věcí, kterou se návštěvníci o projektu dozví, je to, jak vypadají jeho webové stránky. Tato informace je vstřebána ještě předtím, než začnou vnímat jejich skutečný obsah – dříve, než si přečtou jakýkoliv text nebo kliknou na kterýkoliv odkaz. I když se vám to může zdát nespravedlivé, získávání bezprostředního prvního dojmu je něco, čemu nikdo nedokáže zabránit. Vzhled webových stránek jasně ukazuje, zda byla prezentaci projektu věnována nějaká péče. Na rozpoznání, kolik času někdo něčemu věnoval, jsou lidé neobyčejně citliví. Většina z nás dokáže už na první pohled říct, zda byly webové stránky spíchnuty narychlo, nebo zda nad nimi někdo vážně uvažoval. Jde o vůbec první informaci, kterou se váš projekt zviditelňuje, a dojem, jenž vytvoří, pak bude vztažen na všechno ostatní.

Takže i když se značná část této kapitoly bude zabývat obsahem, s nímž by váš projekt měl začít, nezapomínejte, že záleží i na vzhledu a na celkovém dojmu. Protože webové stránky projektu musí být přizpůsobeny dvěma různým typům návštěvníků, uživatelům a vývojářům, je potřeba věnovat zvláštní pozornost jejich srozumitelnosti a přehlednosti. Ačkoliv tato kniha není tím správným místem pro obecné pojednání o webdesignu, jeden princip je natolik důležitý, že si zaslouží, abychom se o něm zde zmínili, zejména pokud vaše stránky slouží několika různým skupinám čtenářů, které se mohou překrývat: návštěvníci by vždy měli mít přibližnou představu o tom, kam daný odkaz vede, ještě předtím, než na něj kliknou. Například odkazy, jež vedou do uživatelské dokumentace, by měly na první pohled nějak oznamovat, že vedou právě tam a ne řekněme do vývojářské dokumentace. Velká část práce ve vedení projektu spočívá v poskytování informací, ale patří sem i nutnost poskytovat je komfortním způsobem. Už pouhá přítomnost určité standardní nabídky na očekávaných místech bude působit dobrým dojmem na uživatele i vývojáře, kteří se rozhodují, zda se budou chtít zapojit. Říká jim totiž, že projekt má celkem jasnou představu, co chce dělat, že se zamyslel nad tím, jaké otázky budou lidé klást, a že se na ně pokusil zodpovědět způsobem, který ze strany tazatele vyžaduje co nejmenší úsilí. Projekt, který dává najevo, že nepodcenil svou přípravu, vysílá návštěvníkům signál: „Pokud se zapojíte, nebude to pro vás ztráta času," což je přesně to, co lidé chtějí slyšet.

Nejdříve se ale rozhlédněte

Než spustíte libovolný open source projekt, nezapomínejte udělat jednu velmi důležitou věc:

Vždy se rozhlédněte kolem sebe, jestli už neexistuje nějaký projekt, který dělá to, co chcete dělat i vy. Je celkem pravděpodobné, že problémem, který jste se rozhodli vyřešit, se už zabýval někdo před vámi. Pokud už jej vyřešil a svůj kód zveřejnil se svobodnou licencí, pak je z vaší strany zcela zbytečné zkoušet objevovat Ameriku. Existují samozřejmě výjimky – pokud chcete zahájit projekt proto, abyste se něco nového naučili, pak vám zdrojový kód někoho jiného nijak nepomůže; zrovna tak se může stát, že projekt, který nosíte v hlavě, je natolik specializovaný, že je šance, že by jej někdo řešil před vámi, zcela nulová. Obecně ale platí, že není žádný důvod se neporozhlédnout; může se vám to navíc velmi vyplatit. Pokud standardní internetové vyhledávače nic nenajdou, zkuste hledat na **http://freshmeat.net/** (server s novinkami o open source projektech, o němž si více povíme později), na **http://www.sourceforge.net/** a v adresáři svobodného softwaru, který spravuje Free Software Foundation na adrese **http://directory.fsf.org/**.

I kdybyste nenašli přesně to, co hledáte, můžete najít něco natolik podobného, že bude rozumnější se k tomuto projektu připojit a přidat požadovanou funkčnost, než začínat sami zcela od začátku.

Začněte od toho, co máte k dispozici

Porozhlédli jste se kolem, nenašli jste nic, co by splňovalo vaše požadavky, a rozhodli jste se založit nový projekt.

Co teď?

Nejtěžší částí spouštění projektu svobodného softwaru je najít způsob, jak své představy zveřejnit. Vy nebo vaše organizace můžete sice velmi dobře vědět, co přesně chcete, ale vyjádřit to tak, aby to bylo srozumitelné i ostatním, vyžaduje celkem dost práce. Přesto je velmi podstatné, abyste si na to našli čas. Vy a ostatní zakladatelé se musíte rozhodnout, co přesně bude cílem projektu – to znamená, že si musíte určit nějaké meze a stanovit nejen to, co váš software bude dělat, ale také co dělat nebude; své závěry pak sepište do jednoho dokumentu vašich závazně stanovených cílů. Tato část většinou není příliš obtížná, ačkoliv někdy může odhalit nevyřčené předpoklady a dokonce rozdílné názory na charakter projektu, což je ale naprosto v pořádku. Je lepší, když se tyto věci začnou řešit teď než později. Dalším krokem je zabalení projektu tak, aby byl stravitelný pro veřejnost, což je po pravdě řečeno čistá otročina.

Tato práce je tolik únavná proto, že spočívá zejména v uspořádávání a zdokumentování věci, které už dávno všichni vědí – tedy všichni, kteří už jsou v projektu zapojeni. Takže pro ty, kteří tuto práci dělají, z ní neplyne žádný bezprostřední užitek. Nepotřebují soubor README, který podává přehled projektu, nepotřebují ani dokument popisující jeho návrh nebo uživatelskou příručku. Nepotřebují mít

pečlivě uspořádaný strom zdrojového kódu, mají svou neformální strukturu, ale jejich zdrojový kód by měl odpovídat rozšířeným standardům distribuce zdrojového kódu. Stejně tak dobře jim poslouží i jakékoliv jiné uspořádání zdrojového kódu, protože už si na ně stejně zvykli; pokud kód lze spustit, vědí, jak jej používat. Dokonce je jim úplně jedno, že nejsou zdokumentovány základní principy architektury projektu, protože ty už také znají.

Jenže nově příchozí tyto věci naopak potřebují. Naštěstí je ale nepotřebují všechny najednou. Není nutné, abyste před zveřejněním projektu sepsali úplně všechno. V dokonalém světě by možná každý nový open source projekt zahájil svou existenci s pečlivě propracovaným dokumentem popisujícím jeho návrh, úplnou uživatelskou příručkou (kde bude zvlášť vyznačeno, které funkce se teprve plánují, ale ještě nejsou implementovány), bezvadně optimalizovaným a přenositelným kódem, který běží na všech možných platformách atd. V reálu by ale něco takového zabralo tak obrovské množství práce, že se do toho nikdo nepustí, zejména když se dá celkem rozumně očekávat, že vám s tím po rozběhnutí projektu pomohou dobrovolníci.

Prezentaci musíte zkrátka věnovat tolik času, aby se nově příchozí účastníci dokázali v projektu zorientovat. Berte to jako jakýsi první krok zaváděcího procesu, kdy je do projektu nutné vložit určitou minimální aktivační energii. Setkal jsem se u tohoto konceptu s označením *hacktivační energie* (hacktivation energy), tedy množství energie, které musí nováček vynaložit předtím, než začne získávat něco zpět. Čím je tato míra u vašeho projektu nižší, tím lépe. Vaším prvním úkolem je snížit tuto vyžadovanou vstupní energii na úroveň, která nebude ostatním bránit v tom, aby se zapojili.

Každý z následujících oddílů popisuje jeden důležitý aspekt zahájení nového projektu. Jsou uvedeny přibližně v tom pořadí, v němž se s nimi nový návštěvník bude setkávat; pořadí, v němž je budete uskutečňovat, může být samozřejmě jiné. Berte ho jako takový seznam úkolů. Při zahájení projektu projděte tyto body jeden za druhým a ujistěte se, že jsou všechny splněny, popřípadě že jste si vědomi toho, jaký dopad může mít, když nějaký přeskočíte.

Vyberte dobré jméno

Představte si, že jste v kůži někoho, kdo se o vašem projektu právě dozvěděl – třeba na něj narazil, když hledal software, jenž by vyřešil nějaký konkrétní problém. První věc, s kterou se setká, je jméno projektu.

Dobré jméno vašemu projektu úspěch samozřejmě nezajistí a asi těžko najdete projekt, který by selhal jenom proto, že měl špatný název – umím si sice představit jména, která by to patrně dokázala, ale vycházejme z předpokladu, že se nikdo nebude aktivně snažit svůj projekt potopit. Rozhodně ale platí, že špatné jméno může přijetí projektu výrazně zpomalit – buď proto, že jej lidé nebudou brát vážně, nebo proto, že budou mít problémy si ho vůbec zapamatovat.

Dobré jméno:

- Udělejte si představu o tom, co váš projekt přináší, nebo s čím nějak souvisí. Takže když víme co projekt přinese, jméno už si vybavíme mnohem snáz.
- Je snadno zapamatovatelné. Tady se nemůžete vyhnout skutečnosti, že standardním jazykem internetu je angličtina, takže „snadno zapamatovatelné“ zde znamená „snadno zapamatovatelné pro někoho, kdo čte anglicky“. Například jména tvořená slovní hříčkou opírající se o výslovnost rodilého mluvčího budou pro mnohé čtenáře, pro něž není angličtina mateřský jazyk, zcela nepochopitelná. Pokud je vaše hříčka opravdu zajímavá a zapamatovatelná, může samozřejmě stát za to ji použít, ale nezapomínejte přitom, že mnozí budou jméno projektu vyslovovat úplně jinak, než jak by jej četl rodilý mluvčí.
- Neshoduje se se jménem nějakého jiného projektu a neporušuje žádné obchodní známky. To patří k dobrým mravům, nemluvě o tom, že z právního hlediska je to rozumné. Navíc nechcete, aby se vaše jméno s něčím pletlo. Sledovat vše, co je na internetu k dispozici, je už tak dost náročné, natož kdyby se různé věci jmenovaly stejně.
- Zdroje, o kterých jsem se zmínil dříve v části **Nejdříve se ale rozhlédněte**, vám pomohou zjistit, zda už vámi zamýšlené jméno nepoužívá někdo jiný. Bezplatné hledání v obchodních ochranných známkách nabízejí servery **http://www.nameprotect.org/** a **http://www.uspto.gov/**.
- Pokud je to možné, je dostupné jako doménové jméno druhé úrovně v doménách `.com`, `.net` a `.org`. Z nich byste si měli vybrat jedno (pravděpodobně `.org`), které budete uvádět jako oficiální domácí místo projektu. Zbylá dvě doménová jména by sem měla přesměrovat – registrují se pouze proto, aby nemohly třetí strany vytvářet zmatek kolem jména projektu. Dokonce i když máte v úmyslu projekt hostovat na serveru někoho jiného (viz **Kompletní hosting**), můžete si stejně zaregistrovat domény odpovídající jménu projektu a přesměrovat je na hostitelský server. Pokud má projekt jednoduché URL, velmi to uživatelům usnadní jeho zapamatování.

Určete jasné cíle projektu

Když lidé najdou webové stránky projektu, začnou na nich hledat nějaký stručný popis, oznámení cílů projektu, aby se mohli (během ne víc než 30 vteřin) rozhodnout, zda mají zájem dozvědět se víc. Takový text by měl být umístěn na nějakém výrazném místě na hlavní stránce, ideálně hned pod jménem projektu.

Cíle projektu by měly být velmi konkrétní, přesně vymezené a především stručné. Tady máme pěkný příklad z **http://www.openoffice.org/:**

> *V rámci komunity vytvořit špičkový, mezinárodně použitelný balík kancelářských aplikací, který poběží na všech hlavních platformách a zpřístupní veškerou svou funkčnost a data prostřednictvím API založených na otevřených komponentech a s použitím formátu souborů založeného na XML.*

Několika slovy tady zachytili všechny hlavní body, přičemž do velké míry spoléhají na předchozí znalosti svých čtenářů. Tím, že napsali *„v rámci komunity“*, dávají najevo, že vývoj nebude ovládat žádná společnost; slovy *„mezinárodně použitelný“* říkají, že software lidem umožní pracovat ve více jazycích a v různých lokalizacích. Spojení *„na všech hlavních platformách“* pak znamená, že software bude spustitelný v systémech Unix, Macintosh a Windows. Zbytek naznačuje, že důležitou součást cílů tvoří použití otevřených rozhraní a snadno srozumitelných formátů souborů. Neříkají zde otevřeně, že chtějí být svobodnou alternativou k Microsoft Office, ale většina lidí to pravděpodobně vyčte mezi řádky. Ačkoliv tento popis cílů projektu vypadá na první pohled dost rozmáchle, ve skutečnosti je celkem dobře vymezený – slova *„balík kancelářských aplikací“* představují pro ty, kteří tento typ softwaru znají, něco velmi konkrétního. Zde se opět využívají předpokládané dřívější znalosti čtenáře (v tomto případě pravděpodobně získané v MS Office) k tomu, aby cíle projektu zůstaly co nejstručnější.

Povaha takovýchto deklarací závisí částečně na tom, kdo je sepsal, a ne pouze na softwaru, který popisují. Například pro OpenOffice.org má smysl použít slova *„v rámci komunity“*, protože projekt byl zahájen a je stále do značné míry sponzorován firmou Sun Microsystems. Přidáním těchto slov firma Sun ukazuje, že si uvědomuje, že mezi ostatními mohou panovat obavy o to, zda se nebude pokoušet vývoj řídit. Ovšem tím, že tuto výtku předvídá už zde na samém začátku, podniká významný krok k tomu, aby se projekt nutnosti tento problém řešit úplně vyhnul. Na druhou stranu projekty, které nejsou sponzorovány žádnou společností, pravděpodobně nic takového říkat nepotřebují – vývoj probíhající komunitním způsobem je v open source světě normální, takže obvykle není žádný důvod jej zde výslovně zmiňovat.

Uveďte, že jde o svobodný projekt

Ti, u nichž zájem přetrvá i po přečtení cílů projektu, se budou chtít dozvědět další podrobnosti, třeba si prohlédnout uživatelskou nebo vývojářskou dokumentaci; možná si budou chtít i něco stáhnout. Ale ještě předtím se budou chtít ujistit, že jde o open source.

Hlavní stránka musí dávat jednoznačně a jasně najevo, že jde o open source projekt. Může vám to připadat jako samozřejmost, ale byli byste překvapeni, kolik projektů na to zapomíná. Viděl jsem webové prezentace projektů svobodného softwaru, na jejichž hlavní stránce nejen že jste nenašli vůbec nic o tom, jakou konkrétní svobodnou licencí se distribuce softwaru řídí, ale kde navíc nebylo ani zmínky o tom, že jde o svobodný software. V některých případech byly tyto zásadní informace vykázány na stránku s odkazy ke stažení či na stránku pro vývojáře, popřípadě na jiné místo, kam se dalo dostat až po dalším kliknutí myši. V extrémních případech nebyla licence na webu k nalezení vůbec – jediný způsob, jak se k ní dostat, bylo software stáhnout a podívat se dovnitř.

Tuto chybu nedělejte. Podobné opomenutí může vést ke ztrátě mnoha potenciálních vývojářů a uživatelů. Hned pod cílem projektu rovnou uveďte, že jde o „svobodný software“ nebo „open source software“ a jakou licenci přesně používá. Stručného průvodce výběrem licence najdete v sekci označené **„Výběr licence a její uplatnění“** dále v této kapitole. Problémy s licencováním se budeme podrobněji zabývat v kapitole **9. Licence, autorská práva a patenty**.

V tento moment už se náš hypotetický návštěvník rozhodl (a trvalo mu to zatím zhruba minutu nebo méně), že by měl zájem věnovat prozkoumání tohoto projektu řekněme dalších pět minut. Následující oddíl popíše, s čím by se během těchto pěti minut měl setkat.

Seznam funkcí a požadavků

Měli byste také uvést krátký seznam funkcí a vlastností, které váš software má (pokud něco ještě není dokončené, uveďte to také, ale připište k tomu *„plánuje se“* nebo *„ve vývoji“*), a jaké výpočetní prostředí ke svému běhu potřebuje. Tento seznam berte jako souhrn informací, které byste poskytli někomu, kdo vás požádá o stručnou charakteristiku vašeho softwaru. Často bude logicky vyplývat z cílů projektu. Řekněme, že je cíl projektu formulován například takto:

> *Vytvořit fulltextový indexovací a vyhledávací nástroj s bohatým API, který by mohli programátoři využít při poskytování vyhledávacích služeb ve větším množství textových souborů.*

Seznam funkcí a požadavků by pak uváděl podrobnosti, které tento cíl popíšou trochu podrobněji:

Vlastnosti:
- *Prohledávání souborů ve formátu prostého textu, HTML a XML*
- *Vyhledávání slov a frází*
- *(plánuje se) Vyhledávání částečných shod (Fuzzy matching)*
- *(plánuje se) Inkrementální aktualizace indexů*
- *(plánuje se) Indexování vzdálených webových serverů*

Požadavky:
- *Python 2.2 nebo vyšší*
- *Dostatečně velký diskový prostor pro uložení indexů (přibližně dvojnásobek velikosti zdrojových dat)*

Na základě těchto informací mohou čtenáři rychle zjistit, zda by jim váš software vyhovoval, nebo zda by se do něj nechtěli zapojit i jako vývojáři.

Stav vývoje

Lidé se vždy zajímají o to, jak si projekt vede. U nových projektů chtějí vědět, jaký je rozdíl mezi tím, co projekt slibuje do budoucna a co už v současnosti umí. U vyzrálých projektů pak chtějí vědět, jak aktivně je projekt udržován, jak často jsou zveřejňovány nové verze, jak pružně asi bude reagovat na hlášení chyb atd.

Jako odpověď na tyto otázky byste měli poskytnout webovou stránku popisující stav vývoje, na níž bude uveden seznam krátkodobých cílů a potřeb projektu (můžete například právě hledat vývojáře

s nějakou specializací). Stránka může obsahovat i historii minulých verzí se seznamy jejich funkcí, takže si návštěvníci mohou udělat představu o tom, jak si váš projekt pro sebe definuje „pokrok“ a jak rychle takové pokroky dělá.

Nebojte se toho, že váš projekt bude vypadat nehotově, a odolejte pokušení stav vývoje přechválit. Všichni vědí, že software se vyvíjí postupně, a není vůbec žádná ostuda říct „Toto je alfa verze a je v ní několik známých chyb. Většinu času sice běží a funguje dobře, ale používejte ji jen na vlastní riziko.“ Vývojáře, které v této fázi vývoje potřebujete, tím neodradíte. A co se týká uživatelů, jednou z nejhorších věcí, které můžete v projektu udělat, je přilákat nové uživatele ještě předtím, než na ně software bude připraven. Jak jednou váš software získá pověst, že je nestabilní a plný chyb, bude se jí jen těžko zbavovat. Z dlouhodobého hlediska se vyplatí být spíš opatrnější. Pro software je vždy lepší, když se ukáže jako stabilnější, než uživatelé čekali – pokud je váš produkt příjemně překvapí, budou jej tím spíše doporučovat i ostatním.

Alfa a beta

Pojmem *alfa* se obvykle označuje první verze, s níž už mohou uživatelé plnohodnotně pracovat a která už obsahuje všechny zamýšlené funkce, ale ve které jsou i známé chyby. Hlavním účelem verze alfa je získání zpětné vazby, aby vývojáři věděli, na co se mají zaměřit. Další etapa, *beta*, popisuje takový stav, kdy už byly všechny závažné chyby opraveny, ale kdy ještě nebyl software dostatečně otestován na to, aby mohl být schválen pro oficiální vydání. Účelem betaverze je buď to, aby se v případě, že se v ní už žádné chyby nenajdou, stala oficiálním vydáním (release), nebo aby mohli vývojáři získat podrobnou zpětnou vazbu a chyby rychleji opravit. Rozdíl mezi verzemi alfa a beta ale není stanoven příliš pevně a konkrétní projekty si jej mohou určit do značné míry samy.

Stahování

Software by mělo být možné stáhnout v podobě zdrojového kódu ve standardních formátech. Pokud byl projekt založen teprve nedávno, není nutné, aby byly k dispozici i binární (spustitelné) balíčky, s výjimkou případů, kdy má software tak složité požadavky na sestavení nebo tak komplikované závislosti, že by většině lidí dalo spoustu práce, než by jej do spustitelného stavu dostali. (Takový projekt ale bude mimochodem shánět nové vývojáře jen velmi těžko.)

Mechanismus distribuce by měl být pohodlný, standardní a co nejméně komplikovaný. Pokud byste se snažili vymýtit nějakou nemoc, také byste léky nedistribuovali tak, aby k jejich aplikaci byla potřeba řekněme neobvykle velká injekční stříkačka. Podobně by se měl i software podřídit standardním metodám sestavení a instalace. Čím víc se bude od standardů odchylovat, tím víc potenciálních uživatelů a vývojářů to po chvíli vzdá a raději odejde.

Může to znít jako samozřejmost, ale mnohé projekty se tím, že by instalační postup nějak standardizovaly, vůbec nezatěžují a celou práci odkládají až na poslední chvíli s tím, že to je přece něco, co se

dá dodělat kdykoliv: *„Všechny tyhle věci vyřešíme, až bude náš kód před dokončením."* Neuvědomují si ale, že když tuto nezáživnou práci s dokončováním sestavovacích a instalačních procedur odkládají, prodlužují tím i dobu, než bude kód dokončen, protože odrazují vývojáře, kteří by jinak k vývoji přispěli. Nejzáludnější na celé věci je, že si přitom vůbec neuvědomují, že o potenciální vývojáře přicházejí, protože celý tento proces sestává z událostí, které nejsou nikde zaznamenány: někdo navštíví webové stránky, stáhne si software, zkusí ho sestavit, nepovede se mu to, vzdá to a jde pryč. To se ale vůbec nikdo nedozví – kromě dotyčné osoby samotné. Nikdo z účastníků projektu nezjistí, že zde byl někdo, kdo měl o projekt zájem a dobrou vůli se přidat, ale kdo zase potichu zmizel.

Nudná práce, která se může hodně vyplatit, by se vždy měla udělat co nejdřív. Významné snížení vstupní bariéry projektu spočívající ve vytváření dobrých instalačních balíčků je přesně případ něčeho, co se vám oplatí velmi bohatě.

Pokud vydáte stažitelný balíček, je důležité, abyste mu přidělili unikátní číslo verze, aby mohl každý snadno srovnat dva různé releasy a poznat, který z nich je novější. Podrobnosti o číslování verzí naleznete v **Číslování verzí**; více o tom, jak vypadá standardní postup sestavování a instalace, najdete v části **Vytváření balíčků** – obojí viz kapitola **7. Vytváření balíčků, vydávání releasů a každodenní práce na vývoji.**

Zpřístupnění systémů správy verzí a sledování chyb

Možnost stažení zdrojových balíčků bude vyhovovat těm, kdo software chtějí jen nainstalovat a používat, ale nebude stačit těm, kteří by jej chtěli ladit nebo přidávat nové funkce. Zde sice mohou pomoci pravidelné noční snímky (Nightly source snapshots) zdrojového kódu, ale ty jsou pro rozvíjející se vývojářskou komunitu pořád ještě příliš hrubé. Vývojářům musíte poskytnout přístup k nejnovějšímu zdrojovému kódu v reálném čase, což se dá zajistit použitím systému pro správu verzí (version control system). Tím, že svůj kód budete spravovat systémem pro řízení verzí a zpřístupníte jej nejširší veřejnosti, vysíláte jak uživatelům, tak vývojářům jasný signál, že se snažíte dát jim vše, co je k účasti na projektu třeba. Pokud nemůžete systém pro správu verzí nabídnout hned, alespoň někde umístěte oznámení, že ho hodláte brzy zavést. Infrastrukturou správy verzí se podrobně zabývá **Správa verzí** v kapitole **3. Technická infrastruktura**.

Totéž platí pro systém sledování chyb projektu (bug tracker). Význam systému pro sledování chyb nespočívá jen v jeho užitečnosti pro vývojáře, ale také v tom, co naznačuje vnějším pozorovatelům projektu. Pro mnoho lidí je zpřístupnění databáze chyb jednou z nejvýraznějších známek toho, že by projekt měl být brán vážně. Kromě toho, čím větší počet chyb v databázi je, tím lépe projekt vypadá. Tohle tvrzení sice na první pohled nedává moc smysl, ale začne, pokud si uvědomíte, že počet zaznamenaných chyb závisí na třech faktorech: na absolutním počtu chyb, které se v softwaru nacházejí, na počtu jeho uživatelů a na tom, jak obtížné je zapsat do systému novou chybu. Z těchto tří faktorů jsou ty druhé dva významnější než ten první. Jakýkoliv software, který je dostatečně veliký a složitý, obsahuje určité, v zásadě libovolné množství chyb, jež čekají na to, až budou odhaleny. Otázkou tedy

spíše je, jak dobře projekt zvládá jejich zaznamenávání a jak přiděluje prioritu jejich řešení. Projekt, který má velkou a dobře udržovanou databázi chyb (tím rozumím to, že na nové chyby rychle reaguje, že duplicitně nahlášené chyby slučuje dohromady atd.), proto působí lepším dojmem než projekt, který žádnou databázi chyb nemá nebo ji má téměř prázdnou.

Pokud byl váš projekt založen teprve nedávno, pak bude samozřejmě databáze obsahovat jen velmi málo chyb, s čímž toho pochopitelně moc nenaděláte. Ale pokud na stránce o stavu vývoje zdůrazníte, že jde o velmi mladý projekt, a pokud lidé při pohledu do databáze chyb uvidí, že většina záznamů byla vložena teprve nedávno, mohou si z toho odvodit, že jsou nové chyby přesto hlášeny tempem, které ukazuje na zdravý růst – a tím, že je zaznamenaných chyb zatím jen málo, se nebudou rozrušovat.

Nezapomínejte, že systémy pro sledování chyb se často používají nejen k zaznamenávání softwarových chyb jako takových, ale také ke sledování požadavků na zlepšení, změn v dokumentaci, nevyřízených úkolů a k dalším účelům. Podrobnosti o tom, jak provozovat systém pro sledování chyb, naleznete v části **Systém pro sledování chyb** v kapitole **3. Technická infrastruktura**, takže zde se jimi dále zabývat nebudeme. Z hlediska prezentace je důležité, aby projekt nějaký systém pro sledování chyb vůbec měl a aby tato informace byla na jeho hlavní stránce uvedena.

Komunikační kanály

Návštěvníci obvykle chtějí vědět, jak se dostanou k lidem, kteří jsou do projektu zapojení. Poskytněte jim tedy adresy mailing listů, chatovacích místností, kanálů IRC a všech dalších fór, kde se mohou spojit s těmi, kteří se softwarem mají něco společného. Nezapomeňte zdůraznit, že vy osobně i všichni ostatní autoři projektu jste do těchto mailing listů přihlášeni, aby každý věděl, že zpětná vazba, kterou sem napíše, se k vývojářům vždy dostane. Z vaší přítomnosti v seznamu adresátů ale nevyplývá závazek odpovídat na všechny otázky nebo implementovat všechny požadované funkce. Z dlouhodobého hlediska se většina uživatelů do fóra stejně nikdy nezapojí, ale ocení, když budou vědět, že tuto možnost mají, pokud by ji někdy potřebovali.

V raných etapách projektu není nutné zřizovat oddělená fóra pro uživatele a pro vývojáře. Mnohem lepší je, když spolu budou všichni zúčastnění mluvit v jedné „místnosti“. Ze začátku je rozdíl mezi vývojáři a uživateli často dost nejasný; i pokud by se dala mezi těmito skupinami jednoznačně vést hranice, pak bývá v počátcích projektu poměr vývojářů k uživatelům obvykle mnohem vyšší než později. I když nemůžete předpokládat, že by byl každý raný účastník programátorem, který by se chtěl na vývoji softwaru podílet, můžete očekávat, že mají přinejmenším zájem sledovat diskuzi kolem vývoje a zjistit, kam bude projekt nasměrován.

Protože se tato kapitola zabývá pouze spuštěním projektu, řekneme si prozatím jen to, že by tato komunikační fóra měla existovat. Později (v části **Jak se vyrovnat s růstem** v kapitole **6. Komunikace**) se podíváme i na to, kde a jak taková fóra vytvořit a zda budou potřebovat moderátory nebo nějakou jinou formu správy. Probereme i to, jakým způsobem a kdy nejlépe oddělit uživatelská fóra od vývojářských, aniž bychom přitom vytvořili nepřekonatelnou propast.

Pokyny pro vývojáře

Pokud někdo bude uvažovat o tom, že by se mohl na projektu podílet, poohlédne se nejprve po pokynech pro vývojáře. Tyto pokyny nejsou ani tak technického jako spíš společenského rázu. Vysvětlují, jakým způsobem vývojáři komunikují mezi sebou a s uživateli a také jak vlastně celá práce probíhá.

Tímto tématem se podrobně zabývá **Jak to všechno zapsat** v kapitole **4. Společenská a politická infrastruktura**, ale už zde můžeme uvést, že tyto pokyny by měly obsahovat:

- odkazy na fóra pro komunikaci s ostatními vývojáři
- instrukce, jak ohlašovat chyby a zasílat záplaty (patche)
- alespoň rámcový popis toho, jakým způsobem probíhá vývoj – tedy zda je projekt řízen formou benevolentní diktatury, demokracie nebo něčeho jiného

Slovo „diktatura“ zde mimochodem není myšleno nijak hanlivě. Označuje se tím druh tyranie, kde má jeden konkrétní vývojář na všechny změny právo veta, což se považuje za naprosto korektní postup. Tímto způsobem funguje celá řada úspěšných projektů. Důležité ale je, aby to projekt oznámil hned na začátku a na rovinu. Diktatura, která předstírá, že je demokracií, lidi odradí; tyranie, která o sobě otevřeně přiznává, že je tyranií, bude fungovat, pokud je tyran schopný a důvěryhodný.

Jako příklad mimořádně pečlivých pokynů pro vývojáře si prohlédněte **http://subversion.apache.org/docs/community-guide/**; obecnější pokyny, které se zaměřují více na vedení projektu a na celkový charakter spoluúčasti než na technické věci, najdete na **http://www.openoffice.org/dev_docs/guidelines.html.**

To, jak software co nejlépe představit novým programátorům, probereme v části **Dokumentace pro vývojáře** dále v této kapitole.

Dokumentace

Dokumentace je naprosto nezbytná. Vždy musí existovat něco, co si lidé mohou přečíst, i kdyby to mělo být jen stručné a neúplné. Práce na dokumentaci je přesně takovou nepopulární činností, o jakých jsme mluvili dříve, a je to často ta první oblast, v níž nový open source projekt pohoří. Sepsání cílů projektu a seznamu jeho funkcí, výběr licence a poskytování informací o stavu vývoje jsou všechno poměrně jednoduché úkoly, které lze dostat do definitivního stavu, na nějž už obvykle není nutné dále sahat. Dokumentace je ale něco, co vlastně nikdy tak úplně hotové není, což může být jeden z důvodů, proč její sepisování lidé často odkládají.

Nejzákeřnější věcí je, že užitečnost dokumentace pro ty, kteří ji píší, je přesně opačná než pro ty, kteří ji budou číst. Pro uživatele-začátečníky jsou v dokumentaci nejdůležitější úplné základy: jak lze software rychle uvést do provozu, přehled toho, jak funguje, možná pár návodů, jak provést běžné úkoly. To jsou ale přesně ty věci, které tvůrci dokumentace znají až příliš dobře – natolik dobře, že pro ně může být

obtížné podívat se na ně očima čtenáře a pracně popisovat kroky, které jim (tedy autorům dokumentace) připadají natolik samozřejmé, že nemá cenu se o nich vůbec zmiňovat.

Pro tento problém neexistuje žádné kouzelné řešení – někdo si zkrátka musí sednout a dokumentaci napsat. Její kvalitu je pak ještě vhodné prověřit tím, že ji předložíte typickým novým uživatelům. Použijte jednoduchý formát, který se snadno upravuje, jako například HTML, prostý text, Texinfo nebo nějakou variantu XML – něco, co je možné jednoduše a rychle pozměnit, kdykoliv to bude potřeba. To proto, abyste zbytečně nekomplikovali život ani původním autorům textu, kteří jej budou chtít postupně vylepšovat, ani těm, kteří se k projektu připojí později a chtěli by na dokumentaci pracovat.

Jeden ze způsobů, jak zajistit, aby alespoň ta nejzákladnější dokumentace vznikla včas, je předem vymezit její rozsah. Díky tomu alespoň nebudou mít její autoři pocit, že je jejich práce nekonečná. Osvědčeným pravidlem je, že by dokumentace měla splňovat alespoň následující kritéria:

- Čtenářům jasně řekněte, jak velké odborné znalosti se u nich očekávají.
- Srozumitelně a podrobně popište, jak se software uvede do provozu; někde na začátku dokumentace uživatelům řekněte, jak provést diagnostické testy nebo spustit jednoduché příkazy, kterými by si mohli ověřit, že vše nastavili správně. Dokumentace týkající se uvedení softwaru do provozu je svým způsobem důležitější než ta, v níž se popisuje, jak software používat. Čím více úsilí někdo bude věnovat instalaci a zprovoznění softwaru, tím vytrvalejší bude při prozkoumávání pokročilých funkcí, které nejsou moc dobře zdokumentované. Pokud to vaši potenciální uživatelé vzdají, bude to někdy ze začátku; právě proto byste měli nejvíc podpory soustředit na počáteční fáze, jako je například instalace.
- Uveďte příklad, jak provést nějakou běžnou úlohu krok za krokem. Samozřejmě čím víc příkladů, tím lépe, ale pokud jste omezeni časem, vyberte si jen jednu věc a popište ji do posledních podrobností. Jakmile někdo zjistí, že se váš software dá použít pro jednu věc, začne sám zkoumat, co dalšího ještě zvládne. A pokud máte štěstí, začnou vaši uživatelé dokumentaci doplňovat sami. Což nás přivádí k dalšímu bodu…
- Označte části dokumentace, o kterých víte, že jsou neúplné. Když přiznáte, že jste si nedostatků dokumentace vědomi, ukážete čtenářům, že chápete jejich situaci. To je uklidní, protože pak budou vědět, že není potřeba někoho přesvědčovat o tom, co je v projektu důležité. V takto označených částech nemusíte nutně slíbit, že nedostatky k určitému datu napravíte – můžete je oprávněně považovat za žádost o pomoc ze strany dobrovolníků.

Tento poslední bod je mnohem důležitější, než na první pohled vypadá, protože se dá se uplatnit na celý projekt, nejen na dokumentaci. To, že uvedete přesný seznam známých nedostatků, je ve světě open source standardem. Na tyto nedokonalosti nemusíte nějak přehnaně poukazovat – prostě je na vhodném místě zaznamenejte (tedy v dokumentaci, v databázi chyb (bug trackeru), při diskuzi v mailing listu a podobně), svědomitě a bez jakýchkoliv emocí. Nikdo to nebude brát ani jako přiznání prohry, ani jako závazek vyřešit problém k určitému datu – pokud tedy takový závazek nebude explicitně vysloven. Protože každý, kdo software používá, přijde na tyto nedostatky časem sám, bude mnohem lepší, když na ně bude psychicky připravený. Projekt pak navíc působí dojmem, že si dobře uvědomuje, jak na tom přesně je.

Údržba FAQ

Dokument FAQ („Frequently Asked Questions“ čili často kladené otázky) může být pro šíření informací o projektu jednou z těch vůbec nejlepších investic. FAQ jsou přesně zacíleny právě na ty otázky, které uživatelé a vývojáři opravdu pokládají (tedy ne na otázky, jejichž výskyt byste možná očekávali). Dobře vedený dokument FAQ proto těm, kteří do něj nahlédnou, často poskytne přesně to, co hledají. Právě FAQ je často tím prvním místem, kam se uživatelé dívají, když narazí na nějaký problém, obvykle ještě dříve, než zkusí hledat v oficiálním manuálu; ve vašem projektu to pravděpodobně bude i tento dokument, na nějž budou jiné servery nejčastěji odkazovat.

FAQ ale bohužel nelze vytvořit hned na začátku projektu. Opravdu dobrý dokument FAQ nemůže být jen tak napsán, musí se vypěstovat. Už z definice jde o text, který na něco reaguje; vytváří se s postupem času jako odezva na každodenní používání softwaru. Protože není možné zcela přesně předpovědět, jaké otázky budou lidé skutečně pokládat, není ani možné si jednoduše sednout a napsat užitečný FAQ jen tak z ničeho.

Takové pokusy jsou jen ztrátou času. Může být užitečné, když ze začátku vytvoříte kostru pro FAQ, která bude na většině míst prázdná; zajistíte tím, že až se projekt rozběhne, bude existovat místo, kam lze připisovat otázky a odpovědi. V této etapě není nejdůležitější úplnost, ale jednoduchost. Pokud půjde do dokumentu FAQ snadno něco přidat, poroste časem sám. (Kvalitní údržba FAQ je úkolem nelehkým a značně spletitým; více se mu budeme věnovat v části **FAQ Manager (manažer sekce s nejčastěji kladenými otázkami)** v kapitole **8. Řízení dobrovolníků.**)

Dostupnost dokumentace

Dokumentace by měla být dostupná na dvou místech: on-line (přímo z webových stránek) a ve stažené distribuci softwaru (viz **Vytváření balíčků** v kapitole **7. Vytváření balíčků, vydávání releasů a každodenní práce na vývoji**). On-line a v podobě zobrazitelné webovým prohlížečem musí být dostupná proto, že dokumentaci uživatelé často čtou ještě předtím, než software poprvé stáhnou – až podle ní se rozhodnou, zda o něj vůbec stojí. Měla by ale být přibalena i k softwaru samotnému na základě principu, že stažením jediného balíčku byste měli získat (a mít lokálně přístupné) vše, co budete potřebovat.

On-line verze by měla obsahovat i odkaz na úplnou dokumentaci v podobě jediné HTML stránky (připojte k odkazu ke stažení i nějakou poznámku jako například „vše v jednom“ nebo „jedna velká stránka“, abyste upozornili na to, že její stahování může nějakou dobu trvat). Tato forma je užitečná proto, že uživatelé často chtějí v celé dokumentaci vyhledat konkrétní slovo nebo frázi. Obvykle je to proto, že už vědí, co přesně hledají, jenom si nemohou vzpomenout, v které části to je. Pro takové uživatele je velmi frustrující, když najdou jednu HTML stránku s obsahem, pak další, která obsahuje úvod, další s pokyny pro instalaci atd. Pokud je celá dokumentace takto rozkouskovaná, bude jim vyhledávací funkce jejich prohlížeče k ničemu. Rozdělení do samostatných stránek je užitečné pro ty, kdo už vědí, jakou část chtějí číst, nebo pro ty, kdo si chtějí celou dokumentaci přečíst od začátku do konce. To ale není ten nejběžnější způsob, jakým se dokumentace používá. Daleko častěji ji čtou ti, kteří už daný

software v podstatě znají, ale potřebují si něco konkrétního ověřit nebo najít. Pokud jim nedáte k dispozici jediný dokument, ve kterém lze jednoduše vyhledávat, zbytečně jim zkomplikujete život.

Dokumentace pro vývojáře

Vývojářská dokumentace je psaná proto, aby programátorům pomohla porozumět kódu, díky čemuž jej pak budou moci snadno opravovat a rozšiřovat. Je to něco jiného než pokyny pro vývojáře, o kterých jsme hovořili dříve – ty jsou spíš společenského než technického charakteru. Pokyny pro vývojáře programátorům říkají, jak se mají chovat k sobě navzájem; vývojářská dokumentace jim říká, jak se chovat ke zdrojovému kódu samotnému. Oba zmíněné dokumenty se pro zjednodušení často slučují do jednoho (jako tomu je v příkladu **http://subversion.apache.org/docs/community-guide/**, který jsme zmínili dříve), ale není to nutné.

Dokumentace pro vývojáře může být velmi užitečná, ale není žádný důvod k tomu, abychom odkládali zveřejnění softwaru jen kvůli ní. Dokud jsou původní autoři dostupní a ochotní odpovídat na otázky týkající se kódu, pak to pro začátek úplně stačí. Nejčastější motivací k sepsání dokumentace je ostatně právě to, že vývojáři musí neustále odpovídat na stejné dotazy. Přispěvatelé, kteří jsou odhodláni se k projektu přidat, si ale obvykle najdou způsob, jak si s kódem poradit i bez dokumentace. Důvod, proč lidé tráví svůj čas studiem zdrojového kódu, je to, že tento kód dělá něco, co je pro ně užitečné. Pokud budou věřit, že je k něčemu dobrý, najdou si čas na to, aby zjistili, jak funguje; pokud jim tato víra schází, pak je žádná vývojářská dokumentace nepřesvědčí ani neudrží.

Takže pokud máte čas na to, sepsat dokumentaci jen pro jednu skupinu, napište ji raději pro uživatele. Koneckonců veškerá uživatelská dokumentace je zároveň i dokumentací pro vývojáře. Každý programátor, který má na softwaru začít pracovat, musí vědět, jak jej používat. Když později uvidíte, že vaši programátoři kladou pořád dokola stejné otázky, vyhraďte si čas na to, abyste napsali zvláštní dokumentaci i pro ně.

Některé projekty používají pro počáteční dokumentaci stránky spravované systémem wiki; někdy je stejným způsobem vedena veškerá dokumentace projektu. Podle mých zkušeností tento způsob opravdu funguje jen tehdy, když tyto wiki stránky vytváří pár lidí, kteří se shodnou na tom, jak by měla být dokumentace organizována a jaký styl by měla používat. Další informace najdete v části **Wiki** v kapitole **3. Technická infrastruktura**.

Vzorový výstup a snímky obrazovky

Pokud má projekt grafické uživatelské rozhraní nebo pokud produkuje grafický nebo jinak výrazný výstup, umístěte na webové stránky pár příkladů. V případě rozhraní to budou snímky obrazovky (screenshoty), v případě výstupu to mohou být rovněž snímky obrazovky nebo prostě soubory. Obojí nasytí lidskou potřebu po okamžitém uspokojení. Jediný snímek obrazovky může být přesvědčivější než celý odstavec textového popisu nebo klábosení v mailing listu, a to čistě proto, že snímek obrazovky je

nezvratným důkazem, že váš software funguje. Může obsahovat chyby, může být obtížné jej nainstalovat, může mít neúplnou dokumentaci, ale snímek obrazovky pořád ukazuje, že pokud uživatel vyvine patřičné úsilí, je možné software zprovoznit.

Snímky obrazovky

Těm, kteří to nikdy nedělali, může pořizování snímků obrazovky připadat složité, takže uvedu stručný návod. Při použití programu Gimp (**http://www.gimp.org/**) otevřete Soubor->Vytvořit->Snímek obrazovky, vyberte Zachytit jedno okno nebo Zachytit celou obrazovku a klikněte na OK. Podle vašeho výběru se buď zachytí celá obrazovka nebo zvolené okno a výsledek se v Gimpu objeví jako obrázek. Podle potřeby obrázek ořežte a upravte jeho velikost podle pokynů, které najdete na **http://www.gimp.org/tutorials/Lite_Quickies/#crop.**

Na webový server projektu můžete umístit i celou řadu dalších věcí, pokud na to máte čas nebo pokud je to z nějakého důvodu zvlášť vhodné: stránku s novinkami, stránku s historií projektu, stránku se souvisejícími odkazy, funkci pro vyhledávání na celém serveru vašeho projektu, odkazy pro dárce atd. V době zahájení projektu nejsou tyto věci nezbytné, ale v budoucnosti se vám můžou hodit.

Kompletní hosting

Existuje několik serverů, které zdarma nabízejí kompletní hosting a infrastrukturu pro open source projekty: webový prostor, systém pro správu verzí, systém pro sledování chyb, prostor pro stahování, diskuzní fóra, pravidelné zálohování atd. V podrobnostech se od sebe liší, ale základní služby poskytují všechny zhruba stejné. Když jednoho z těchto serverů využijete, získáte bez práce spoustu věcí; to, čeho se ale musíte vzdát, je plný vliv na to, jak budete působit na své uživatele. Je to správce hostingové služby, kdo rozhoduje, jaký software na serveru poběží, což může zásadním způsobem ovlivnit celkový dojem z webových stránek projektu.

Podrobnější diskuzi o výhodách a nevýhodách kompletního hostingu a seznam serverů, které jej nabízejí, najdete v části **Kompletní hosting** v kapitole **3. Technická infrastruktura**.

Výběr licence a její uplatnění

Tato podkapitola má sloužit jako velmi rychlý orientační návod pro výběr licence. Pokud chcete porozumět podrobným právním důsledkům různých licencí a tomu, jak může výběr licence ovlivnit možnost mísení vašeho softwaru s jiným svobodným softwarem, přečtěte si kapitole **9. Licence, autorská práva a patenty.**

Existuje velké množství licencí pro svobodný software, z nichž si můžete vybírat. Většinou z nich se zde nemusíme zabývat, protože byly napsány tak, aby vyhověly právním potřebám konkrétních společností nebo osob, a pro váš projekt by nebyly vhodné. Omezíme se pouze na ty nejběžněji používané licence, protože ve většině případů pro vás bude nejlepší vybrat si jednu z nich.

Licence typu „vše je povoleno“

Pokud vám nevadí, že by kód vašeho projektu mohl být případně využit v proprietárních programech, pak použijte licenci ve stylu *MIT/X*. Je to ta vůbec nejjednodušší z řady minimalistických licencí a nedělá prakticky nic jiného, než že jmenuje držitele autorských práv, aniž by nějak omezovala možnost kopírování; výslovně uvádí, že na kód není poskytována žádná záruka. Podrobnosti najdete v **MIT / X Window System License**.

GPL

Pokud nechcete, aby byl váš kód použit v proprietárních programech, použijte GNU General Public License (**http://www.gnu.org/licenses/gpl.html**). GPL je v současnosti pravděpodobně nejznámější licencí pro svobodný software na světě. To už samo o sobě znamená velkou výhodu, protože ji už mnoho vašich potenciálních uživatelů a přispěvatelů bude znát a nebudou potřebovat věnovat další čas tomu, aby si vaši licenci přečetli a porozuměli jí. Další informace najdete v **GNU General Public License** v kapitole **9. Licence, autorská práva a patenty.**

Pokud uživatelé s vašimi programy pracují především prostřednictvím sítě, a váš software je tedy obvykle součástí hostované služby, pak místo GPL zvažte použití *GNU Affero GPL*. Podrobnosti najdete v části **GNU Affero GPL: Verze GNU GLP pro kód na straně serveru** v kapitole **9. Licence, autorská práva a patenty.**

Jak licenci aplikovat

Až si licenci vyberete, nezapomeňte ji uvést na hlavní stránce projektu. Nemusíte tam vkládat celý text licence – stačí jen její jméno a odkaz na plné znění nacházející se na jiné stránce.

Tím veřejnosti řeknete, s jakou licencí máte v úmyslu software zveřejnit, ale z právního hlediska tento krok nestačí. Je potřeba, aby tuto licenci obsahoval i software samotný. Obvykle se to dělá tak, že licenci v plném znění přiložíte do souboru nazvaného `COPYING` (nebo `LICENSE`) a na začátek každého zdrojového souboru připojíte krátké oznámení, v němž uvedete datum copyrightu, jeho držitele a typ licence, včetně odkazu na její plné znění.

To se dá dělat celou řadou různých způsobů, takže si uvedeme jen jeden příklad. GNU GPL doporučuje na začátek každého souboru uvést následující oznámení:

```
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>
```

Není zde výslovně řečeno, že se kopie licence obdržené spolu s programem nachází v souboru `COPYING`, ale obvykle se vkládá právě tam. (Citované upozornění samozřejmě můžete upravit tak, aby to uvádělo přímo.) Tato šablona uvádí i odkaz na webovou stránku, která obsahuje text licence. Další běžnou metodou je uvedení poštovní adresy, na které si můžete kopii licence objednat. Podle svého uvážení uveďte odkaz na jakékoliv místo, kde si myslíte, že je k nalezení nejtrvanlivější kopie licence – což může být jednoduše například na vašich webových stránkách. Obecně se dá říct, že poznámka přidávaná na začátek každého souboru nemusí nutně vypadat úplně stejně jako ta, kterou jsme si zde ukázali; důležité je, aby měla na začátku oznámení o držiteli copyrightu a datum, uvedla jméno licence a jednoznačný odkaz na to, kde nalézt její plné znění.

Udávání tónu

Zatím jsme se zabývali jednorázovými úkony, které se při zahájení projektu provádějí: výběr licence, uspořádání webových stránek atd. Ty nejdůležitější aspekty zahájení nového projektu jsou ale dynamické. Zvolit adresu mailing listu je snadné, ale zajistit, aby se v něm konverzace držely tématu a byly produktivní, to už je něco úplně jiného. Pokud svůj projekt otevíráte po několika letech uzavřeného, interního vývoje, bude potřeba radikálně změnit podobu jeho dosavadních vývojových procesů, na což musíte stávající členy týmu připravit.

První kroky jsou nejtěžší, protože ještě nejsou ustanoveny precedenty a nikdo zatím přesně neví, jak bude vše probíhat. Stabilita projektu nemá svůj původ v nějaké sadě předpisů, ale ve sdílené kolektivní moudrosti, která se sama časem vyvíjí a již lze jen těžko nějak zachytit. Často existují i psaná pravidla, ale ta obvykle bývají spíše jakýmsi zjednodušením těch nepsaných a neustále se proměňujících dohod, které projekt skutečně řídí. Psané zásady kulturu projektu ani tak nedefinují, jako spíš popisují – a i to jen přibližně.

Pro to, proč věci fungují právě takto, existuje několik důvodů. Růst a vysoká fluktuace zúčastněných nejsou pro vytváření společných norem zas tak velkou překážkou, jak bychom se mohli domnívat. Pokud se změny nedějí až příliš rychle, mají nově příchozí vždy dost času se seznámit s tím, jak věci fungují; až se vše naučí, budou sami dodržování stejných pravidel prosazovat. Funguje to trochu podobně jako způsob, kterým se šíří dětské říkanky. Dnešní děti používají stále zhruba stejná říkadla jako před stovkami let, ačkoliv tehdejší děti už dávno nežijí. Mladší děti je znají z podání starších dětí; až vyrostou, naučí je zase ty mladší. Tohle samozřejmě děti nedělají cíleně, aby zajistily jejich předání dalším generacím, ale ve skutečnosti dělají právě to: celý pravidelně se opakující mechanismus znamená, že říkadla přežívají dál. Časová rozpětí projektů svobodného softwaru se možná nebudou pohybovat v řádu staletí (na dělání takových závěrů je ještě trochu brzo), ale dynamika přenosu je do značné míry stejná. Výměna účastníků je zde ovšem podstatně rychlejší, což je potřeba kompenzovat tím, že celý přenos bude probíhat aktivněji a cílevědoměji.

Tomu pomáhá fakt, že když většina lidí někam přijde, bude předpokládat, že tam existují nějaké společenské normy, a začne se po nich pídit. Lidé jsou zkrátka takoví. V jakékoliv skupině, kterou sdružuje společný cíl, noví účastníci instinktivně hledají způsoby chování, jež by sdělovaly, že sem patří. Cílem vytvoření precedentů co nejdříve je zajistit, aby se tímto standardním chováním stalo něco, co projektu prospívá, protože jak se normy jednou podaří vytvořit, budou už se do značné míry udržovat samy.

V dalších oddílech ukážu pár příkladů konkrétních věcí, které můžete pro vytvoření dobrých precedentů udělat. Jejich seznam není vyčerpávající, má spíše ukázat, jak může projektu ohromně pomoct, když se v něm už na začátku vytvoří atmosféra spolupráce. V reálném světě může sice každý vývojář pracovat sám a zcela odděleně, ale můžete v něm zkusit vyvolat pocit, jako kdyby pracoval společně s ostatními v jedné veliké místnosti. Čím silněji se takto bude cítit, tím víc času bude chtít projektu věnovat. Tyto konkrétní příklady jsem si vybral, protože se objevily v projektu Subversion (**http://subversion.tigris.org/**), kterého jsem se účastnil a který jsem sledoval od úplného začátku. Rozhodně ale nejsou jedinečné pro Subversion – s podobnými situacemi se setkáte u většiny open source projektů a měli byste je vnímat jako příležitosti pro vykročení tou správnou nohou.

Vyhněte se soukromým diskuzím

I po zveřejnění projektu se spolu s jeho dalšími zakladateli často přistihnete při tom, že byste raději obtížné otázky vyřešili při soukromých rozhovorech v úzkém kruhu. Platí to zvlášť na začátku projektu, kdy se musí učinit řada důležitých rozhodnutí a kdy obvykle máte jen hrstku dobrovolníků, kteří by k tomu byli způsobilí. V takových situacích budete mít před očima všechny nevýhody, které diskuze ve veřejných mailing listech mají: zpoždění spojené s konverzacemi prostřednictvím e-mailu, nutnost ponechat dostatek času na zformování konsenzu, občas velmi únavné jednání s naivními dobrovolníky, kteří si myslí, že všem problémům rozumí, ale ve skutečnosti nerozumí (takoví se vyskytují v každém projektu; někdy se z nich časem stanou vaši nejlepší přispěvatelé, někdy zůstanou naivními napořád), nebo těmi, kdo zkrátka nedokážou pochopit, proč chcete řešit jen problém X, když se zjevně jedná o podmnožinu většího problému Y atd. Pokušení rozhodnout za zavřenými dveřmi a výsledek pak předložit jako hotovou věc nebo přinejmenším jako silná doporučení jednotné a vlivné skupiny lidí bude ve skutečnosti značné.

Ale nedělejte to.

I když mohou být veřejné diskuze někdy velmi pomalé a neohrabané, z dlouhodobého hlediska jsou téměř vždy tím nejlepším řešením. Pokud budete vést důležité rozhovory pouze v soukromí, mnoho přispěvatelů tím odradíte. Žádný normální dobrovolník nezůstane dlouho někde, kde všechna velká rozhodnutí dělá jakýsi tajný výbor. Kromě toho mají veřejné diskuze i pozitivní vedlejší účinky, které jsou z dlouhodobého hlediska významnější než ten konkrétní problém, který se zrovna řešil:

- Diskuze pomáhá školit a vychovávat nové vývojáře. Nikdy nevíte, kolik očí konverzaci sleduje – většina lidí se jí vůbec nemusí účastnit, jen ji tiše sledovat, a získávat tak informace o softwaru.
- Diskuze školí i vás samotné – učí vás vysvětlovat technické problémy lidem, kteří se softwarem nejsou tak obeznámení jako vy. Tato dovednost vyžaduje praxi a tu nezískáte, když budete mluvit s lidmi, kteří už ví všechno, co víte i vy.
- Diskuze a její závěry zůstanou už napořád uchovány ve veřejných archivech projektu, díky čemuž se budoucí debaty budou moct vyhnout prošlapávání stejných cest. Viz **Nápadné využívání archivů** v kapitole **6. Komunikace.**

Konečně existuje i možnost, že někdo z účastníků přispěje do konverzace myšlenkou, která vás vůbec nenapadla. Je obtížné říci, jaká je pravděpodobnost, že toto nastane – závisí to na složitosti kódu a na stupni požadované specializace. Ale ze svých zkušeností bych si dovolil tvrdit, že se to stává častěji, než by si většina lidí myslela. V projektu Subversion jsme (my zakladatelé) věřili, že před námi stojí několik zásadních a složitých problémů, o kterých jsme usilovně přemýšleli několik měsíců; zcela upřímně jsme pochybovali, že by do této diskuze byl někdo z čerstvě založeného mailing listu schopen nějak smysluplně přispět. Takže jsme si vybrali tu snadnější cestu a vyměňovali jsme si své technické nápady přes soukromé maily. Brzy si ale jeden pozorovatel projektu[10] domyslel, co se děje, a požádal nás, abychom diskuzi přenesli do veřejného mailing listu. Pomysleli jsme si své, ale udělali jsme to. To množství promyšlených, věcných komentářů a návrhů, které se záhy začaly objevovat, nás zcela ohromilo. Hned několikrát přišel někdo s návrhem, který nás vůbec nenapadl. Ukázalo se, že v mailing listu bylo celou dobu několik velmi chytrých lidí, kteří jen čekali na tu správnou návnadu. Je tedy pravda, že následná debata zabrala delší čas, než kdybychom zůstali u soukromé konverzace, ale byla tak produktivní, že za ten čas navíc rozhodně stála.

Nechci tady vyslovovat obecná moudra jako „skupina je vždy chytřejší než jednotlivec“ (každý z nás už se v životě setkal se skupinami, u nichž by se o tom dalo vážně pochybovat), ale určitě platí, že existují činnosti, které skupiny dělají mnohem lépe. Jednou z nich je vzájemná kontrola nových příspěvků; rychlé generování velkého počtu nových nápadů je další. Kvalita těchto nápadů samozřejmě závisí na těch, kdo s nimi přišli; na druhou stranu ale nikdy nezjistíte, jak chytré vaše publikum vlastně je, dokud mu nepředložíte nějaký zajímavý problém.

---

[10] K části věnované oceňování zásluh jsme se zatím nedostali, ale stejně bych měl dodržet to, co budu později kázat: tímto pozorovatelem byl Brian Behlendorf a byl to právě on, kdo poukázal na to, jak je důležité vést všechny debaty na veřejnosti, pokud neexistuje nějaký dobrý důvod držet je v soukromí.

Existují samozřejmě i debaty, které musí být vedeny soukromě, a v této knize si několik takových případů popíšeme. Základním principem by ale vždy mělo být: *Pokud není důvod k soukromé debatě, měla by být veřejná.*

Pro to je ale potřeba něco udělat. Nestačí jen zajistit, aby se všechny vaše příspěvky objevily na veřejném mailing listu. Musíte do něj dostat i ostatní, kteří dál vedou soukromé rozhovory, aniž by k tomu měli důvod. Pokud se někdo pokusí zahájit soukromou diskuzi na téma, které to podle vašeho názoru nevyžaduje, považujte za svou povinnost na to okamžitě upozornit. K původnímu tématu se vůbec nevyjadřujte, dokud se vám buď nepovede konverzaci úspěšně přesměrovat na veřejnost, nebo dokud vás ostatní nepřesvědčí, že tato diskuze opravdu musí zůstat soukromá. Pokud v tom budete důslední, ostatní se toho rychle chytí a začnou používat veřejné fórum přednostně.

Nezdvořilé chování potlačte hned v zárodku

Od prvních chvil veřejné existence projektu byste měli na jeho fórech dodržovat zásadu nulové tolerance vůči hrubému a urážlivému chování. Nulová tolerance nemusí nutně znamenat použití technických prostředků. Pokud se někdo naváží do jiných účastníků, nemusíte jej hned odstraňovat z mailing listu; zrovna tak nemusíte odebírat právo přispívat do projektu těm, kdo píší hanlivé poznámky. (Teoreticky můžete být časem k něčemu takovému přinuceni, ale vždy až poté, co všechny ostatní způsoby selhaly, což se už z definice na začátku projektu stát nemůže.) Nulovou tolerancí se jednoduše rozumí to, že špatné chování nikdy neprojde bez povšimnutí. Pokud například někdo napíše technický komentář smíchaný s útokem *ad hominem* na jiného vývojáře projektu, pak musíte nejprve zareagovat na tento útok jako na zcela samostatný problém a až pak přejít k technickému obsahu.

Bohužel se velmi snadno může stát (a není to nijak neobvyklé), že se konstruktivní diskuze zvrhnou ve zbytečné válčení. Do e-mailu někdy lidé napíší věci, které by někomu do očí nikdy neřekli. Tento efekt často zesiluje téma, které se právě probírá. Při řešení technických problémů se často objevuje pocit, že na většinu otázek existuje jediná správná odpověď a že nesouhlas s touto odpovědí lze vysvětlit jen jako výraz nevědomosti nebo hlouposti. Od toho, že někdo prohlásí předložené technické řešení za pitomé, už je jen krůček k tomu, aby byl za pitomce prohlášen jeho autor. Ve skutečnosti je někdy velmi těžké určit, kde končí technická debata a začíná osobní útok, což je také jeden z důvodů, proč není rozumné reagovat drastickými prostředky nebo tresty. Když něco podobného zpozorujete, je vhodné místo toho zareagovat příspěvkem, který zdůrazní důležitost zachování přátelského tónu diskuze, aniž by kohokoliv obviňoval z pokusů ji rozvrátit. Tyhle příspěvky psané ve stylu „hodný policajt“ naneštěstí často znějí, jako když učitelka v mateřské školce dětem vysvětluje, jak se chovat slušně:

> *Za prvé prosím omezte komentáře, které lze vztáhnout k nějaké osobě. Neříkejte tedy například, že návrh bezpečnostní vrstvy, který předložil J., je „naivní a prokazuje základní neznalost toho, jak počítačová bezpečnost funguje“. Možná to pravda je, možná není, ale rozhodně to není způsob, jak vést diskuzi. J. svůj návrh napsal v dobré víře. Pokud má nějaké nedostatky, upozorněte na ně a společně je buď opravíme, nebo vymyslíme něco jiného. M. určitě neměl v úmyslu*

J. urazit, ale jeho formulace byla nešťastná a my se tady pokoušíme zůstat konstruktivní. Teď tedy k tomu návrhu: myslím, že M. má pravdu v tom, že…

I když takové odpovědi mohou znít trochu prkenně, jsou celkem účinné. Pokud budete na špatné chování soustavně poukazovat, ale přitom nebudete po nikom vyžadovat omluvu nebo přiznání vlastní chyby, dáváte diskutujícím čas trochu vychladnout a pro příště ukázat svou lepší stránku v podobě uhlazenějšího chování – což se také obvykle stane. Jedno z tajemství úspěchu této strategie spočívá v tom, že z této vedlejší diskuze nikdy neuděláte hlavní téma. Vždy by to měla být jen krátká poznámka bokem, která předchází hlavní části vaší odpovědi. Upozorněte jaksi mimochodem na to, že „tohle se tady nedělá“, ale pak se věnujte skutečnému obsahu, abyste lidem dali něco, co se týká tématu a na co by mohli reagovat. Pokud někdo začne protestovat, že si vaše pokárání nezasloužil, rozhodně se odmítněte nechat zatáhnout do dalšího sporu. Buď na to vůbec nereagujte (pokud usoudíte, že si prostě potřeboval ulevit a žádná odpověď není třeba), nebo napište, že se za svou přehnanou reakci omlouváte a že se v e-mailech některé jemnější významové odstíny rozlišují jen obtížně. Pak se vraťte k hlavnímu tématu. V žádném případě netrvejte na tom, aby dotyčný uznal, ať už veřejně nebo soukromě, že se choval nevhodně. Pokud se sám od sebe omluví, bude to jen dobře, ale pokud byste to po něm vyžadovali, akorát vyvoláte odpor.

Vaším hlavním cílem je to, aby bylo dodržování pravidel slušného chování vnímáno jako jedna z vnitřních zásad skupiny. Projektu to pomůže, protože podobné hašteření může vést až k odchodu některých vývojářů (a to dokonce i z projektů, které se jim líbí a které chtějí podporovat). To je opět něco, co se nemusíte vůbec dozvědět – může to být někdo, kdo do mailing listu nikdy nepřispívá, ale pečlivě jej sleduje. Pokud dojde k závěru, že účast na projektu vyžaduje hroší kůži, může se rozhodnout, že mu to za to nestojí. Udržování přátelského charakteru fóra je z dlouhodobého hlediska důležité pro přežití projektu a je to mnohem snazší, dokud v něm není mnoho lidí. Jakmile se to stane součástí kultury projektu, nebudete už tím jediným, kdo to bude prosazovat. Pomůžou vám s tím všichni ostatní.

Kontrolujte kód veřejně

Jeden z nejlepších způsobů, jak zajistit, že vaše komunita vývojářů bude výkonná, je zavést pravidlo, že si svůj kód všichni navzájem prohlížejí. Aby se to dalo provádět efektivním způsobem, je potřeba zajistit příslušnou technickou infrastrukturu, především zapnout rozesílání informací o změnách kódu e-mailem (podrobnosti najdete v části **E-maily oznamující commit**). Pokaždé když někdo do zdrojového kódu promítne změnu (commit), rozešle se e-mail, v němž se objeví příslušná zpráva z protokolu a rozdíly, v nichž tato změna spočívá (viz **diff**, v části **Slovníček pojmů správy verzí**). *Revizí kódu* se rozumí praxe, kdy se každý z těchto e-mailů při přijetí pečlivě kontroluje, přičemž se v něm hledají chyby a zkoumá se, zda a jak by se dal zlepšit.[11]

[11] Tak se to alespoň dělá u většiny open source projektů. U centrálněji řízených projektů se „revize kódu“ provádí i tak, že se několik lidí společně sejde, projde vytištěný zdrojový kód a pokusí se v něm najít konkrétní problémy a identifikovat časté nedostatky.

Revize kódu slouží několika účelům současně. Je to dobrý příklad toho, jak si v open source světě všichni kontrolují svou práci navzájem, což pomáhá udržet kvalitu softwaru. Každá chyba, která se ve vydané verzi vašeho programu objeví, se tam dostala tak, že byla při zapsání kódu přehlédnuta. Z toho vyplývá, že čím víc očí nově zapsané změny prohlédne, tím méně chyb se dostane na veřejnost. Revize kódu ale zároveň dělá ještě něco jiného – pro účastníky projektu znamená potvrzení, že někomu na jejich práci a na tom, jaký bude mít výsledek, záleží, protože jinak by jen těžko věnovali svůj čas tomu, aby ji kontrolovali. Když lidé vědí, že si ostatní najdou čas na to jejich práci zhodnotit, odvádějí ji, jak nejlépe dokážou.

Revize by měly být veřejné. Dokonce i v případech, kdy jsme seděli s vývojáři v jedné místnosti a někdo z nás zapsal změnu, jsme si dávali pozor na to, abychom revizi neprováděli jen ústně, ale poslali ji do vývojářského mailing listu. Z provádění revizí před očima všech plynou různé výhody. Ti, kteří revizní komentáře čtou, v nich můžou najít nedostatky; v případech, že žádné nenajdou, jim tato činnost bude alespoň připomínat, že revize kódu je něco běžného, co se musí dělat pravidelně, stejně jako třeba umývání nádobí nebo sekání trávníku.

V projektu Subversion jsme ze začátku kód nijak pravidelně nekontrolovali. Neexistovala žádná záruka, že každou změnu kódu někdo zkontroluje; pokud se někdo o příslušnou oblast programu zvlášť zajímal, mohl si občas změny prohlédnout. Tak se stávalo, že se do zdrojového kódu dostaly i celkem zbytečné chyby, které opravdu měly být zachyceny dříve. Vývojář jménem Greg Stein, který už z dřívějška věděl, jak cenné revize mohou být, se rozhodl, že půjde všem příkladem a bude recenzovat každičký řádek každého nového commitu, který se v úložišti kódu objeví. Na každou změnu, kterou někdo zapsal, brzy Greg zareagoval komentářem ve vývojářském mailing listu, v němž ji rozpitval, analyzoval možné problémy a někdy i vyjádřil své uznání nad zvlášť rafinovaným programátorským kouskem. Ihned zachytil chyby a nestandardní postupy, které by jinak prošly bez povšimnutí. Ani jednou si nepostěžoval, že je tím jediným, kdo každou změnu kontroluje, ačkoliv mu to muselo zabrat spoustu času; při každé vhodné příležitosti ale zdůrazňoval, jak je revize kódu důležitá. Poměrně brzy ji začali pravidelně provádět i ostatní, mne nevyjímaje. Jakou jsme měli motivaci? Nebylo to proto, že bychom se před Gregem začali stydět. Dokázal nám ale, že čas strávený revizí kódu opravdu není ztracený a že touto činností můžete projektu prospět zrovna tak jako psaním nového kódu. Když jsme si to všichni uvědomili, stalo se revidování kódu standardním postupem – až do té míry, že pokud nějaká změna prošla bez reakce, začal si její autor dělat starosti a dokonce se pak v mailing listu vyptával, jestli se na ni opravdu zatím nikdo nepodíval. Greg později dostal jinou práci, která znamenala, že projektu Subversion už nemohl věnovat tolik času a musel s pravidelnou kontrolou kódu přestat. Tou dobou už ale byl tento zvyk natolik zakořeněný, že se nám zdálo, že tu byl už od nepaměti.

Začněte revidovat kód hned u prvních příspěvků. Mezi problémy, které se při revizi rozdílů nejsnadněji zachytí, patří bezpečnostní nedostatky, úniky paměti, nedostatečné komentování nebo dokumentace API, cykly provedené s jedním opakováním navíc/méně, nedodržení standardní komunikace mezi volajícím a volaným (caller/calee) a další problémy, které se dají zpozorovat i v minimálním kontextu. Může ale zachytit i jiné nedostatky, které jsou mnohem rozsáhlejší, jako například možnost sloučit podobné části kódu na jediné místo, protože pokud děláte revize pravidelně, pak si při prohlížení jedné části snadno vybavíte, co jste kontrolovali předtím.

Nedělejte si starosti s tím, že možná nenajdete nic, co by se dalo okomentovat, nebo že toho o některých částech kódu víte jen málo. Skoro ke každé změně je co říct. Když v ní nenajdete žádný problém, možná najdete alespoň něco, co si zaslouží pochvalu. Důležité je dát každému přispěvateli najevo, že si jeho příspěvku někdo všiml a že mu porozuměl. Revize kódu samozřejmě nezbavují programátory zodpovědnosti za to, aby si své změny před zapsáním prohlédli a otestovali sami. Nikdo by neměl spoléhat na to, že revize jeho kódu odhalí věci, kterých si měl všimnout sám.

Když otvíráte dosud uzavřený projekt, uvědomte si rozsah změn

Pokud otvíráte existující projekt, který má své aktivní vývojáře zvyklé na práci v uzavřené skupině, ujistěte se, že všichni rozumí tomu, že to přinese velké změny – a také že se na věc dokážete podívat jejich očima.

Zkuste si představit, jak se celá situace jeví jim: dříve psala kód a o projektu rozhodovala skupina programátorů, kteří daný software znali víceméně stejně dobře, kteří byli pod stejným tlakem stejného vedení a kteří vzájemně znali své silné a slabé stránky. Teď po nich chcete, aby svůj kód vydali napospas zkoumavému pohledu úplně cizích lidí, kteří je budou soudit výhradně na základě kódu, aniž by tušili, že některá rozhodnutí mohla být výsledkem tlaku shora. Tito noví lidé budou pokládat řadu otázek, díky nimž si původní vývojáři uvědomí, že dokumentace, se kterou se tolik nadřeli, na svůj úkol nestačí (to je něco, čemu se nedá vyhnout). A aby toho nebylo málo, všichni tito nováčci jsou zcela neznámí jedinci bez tváře. Pokud si některý z vašich vývojářů už předtím nebyl úplně jist svými schopnostmi, představte si, jak se jej dotkne, až někdo přicházející zvenku poukáže na chyby v jeho kódu, a co hůř, ještě před jeho kolegy. Pokud váš tým nesestává výhradně z dokonalých programátorů, tak se něčemu takovému nevyhnete; pravděpodobně se to ze začátku stane každému z nich. Ne proto, že by jejich práce byla špatná, ale proto, že každý dostatečně složitý program chyby zkrátka obsahuje a kontrola kódu někým jiným než původním autorem některé z nich může objevit (viz **Kontrolujte kód veřejně** výše v této kapitole). Zároveň bude platit, že tyto nováčky samotné nebude kritizovat prakticky nikdo, protože se ještě neseznámili s projektem natolik, aby do něj mohli sami přispět. Vaši vývojáři tak mohou snadno získat pocit, že veškerá kritika směřuje zvenku na ně a nikdy ne opačným směrem; hrozí, že si začnou připadat jako v obležení a budou se i bránit.

Nejlepší způsob, jak tomu předejít, je každého upozornit, co nastane, vysvětlit to, sdělit jim, že je naprosto normální, když to na začátku bude poněkud nepříjemné, a ujistit je, že se to zlepší. Některá z těchto varování by měla proběhnout soukromě ještě předtím, než je projekt otevřen. Může ale být užitečné všem účastníkům ve veřejných mailing listech čas od času připomenout, že vývoj projektu teď probíhá novým způsobem a že nějakou dobu potrvá, než se vše přizpůsobí. To nejlepší, co můžete udělat, je jít sám příkladem. Pokud uvidíte, že vaši vývojáři neodpovídají dostatečně na otázky nováčků, pak moc nepomůže, když jim řeknete, aby odpovídali víc. Možná zatím nemají vyvinutý smysl pro to, co si odpověď zaslouží a co ne; možná si ještě neumí pořádně uspořádat priority mezi programováním samotným a novou zátěží v podobě komunikace s okolím. K účasti je přimějete tak, že se zúčastníte sami. Sledujte veřejný mailing list a na otázky, které se v něm objeví, odpovídejte. Pokud nějaký dotaz přesahuje vaše odborné znalosti, pak jej viditelně přihrajte jinému vývojáři, který dané problematice

rozumí – a ujistěte se, že na něj pak skutečně odpoví nebo alespoň zareaguje. Dlouholetí vývojáři budou v pokušení důležité otázky řešit v soukromých diskuzích, protože jsou na to zvyklí. Nezapomeňte tedy ani sledovat dění v interních mailing listech, na kterých k tomu může docházet, abyste mohli případně požádat o přenesení celé diskuze na nějaké veřejné fórum.

S otevřením dříve uzavřených projektů jsou spojeny i další, dlouhodobé problémy. V kapitole **5. Peníze** se podíváme na to, jak úspěšně řídit spolupráci placených a neplacených vývojářů, a v kapitole **9. Licence, autorská práva a patenty** si řekneme, proč je třeba postupovat opatrně při zpřístupňování soukromých zdrojových kódů, které mohou obsahovat software napsaný nebo „vlastněný“ dalšími stranami.

Oznamování

Až bude projekt ve stavu, který se dá předvést na veřejnosti (nemusí být dokonalý, stačí, když bude přijatelný), je čas jeho existenci oznámit. To je až překvapivě jednoduchá záležitost: jděte na **http://freshmeat.net/**, klikněte na tlačítko Submit v horní navigační liště a vyplňte oznamovací formulář. Freshmeat všichni sledují jako to místo, kde se nové projekty ohlašují. Stačí, aby si oznámení vašeho projektu všimlo pár lidí; pak už se tato informace začne šířit sama.

Pokud víte o mailing listech nebo diskuzních skupinách, kam by oznámení vašeho projektu tematicky zapadalo a kde by mohlo vyvolat zájem, napište tam; dejte ale pozor na to, abyste do každého fóra poslali jen jedno oznámení a abyste případnou následující diskuzi přesměrovali do fóra vyhrazeného vašemu projektu (nastavením hlavičky Reply-to). Zpráva by měla být krátká a měla by jít rovnou k věci:

```
Komu: diskuzni@list.priklad.org
Předmět: [ANN] Projekt fulltextového indexovacího nástroje Scanley
Reply-to: dev@scanley.org

Toto je jednorázové oznámení vzniku projektu Scanley, což je
open source fulltextový indexovací a vyhledávací nástroj s bohatým
API, jenž by mohli využívat programátoři při poskytování vyhledávacích
služeb ve větším množství textových souborů.  Scanley má v současnosti
podobu funkčního kódu, je aktivně vyvíjen a hledá nové vývojáře
i testery.

Domovská stránka: http://www.scanley.org/
```

```
Vlastnosti:
   - Prohledávání souborů ve formátu prostého textu, HTML a XML
   - Vyhledávání slov a frází
   - (plánuje se) Vyhledávání částečných shod (Fuzzy matching)
   - (plánuje se) Inkrementální aktualizace indexů
   - (plánuje se) Indexování vzdálených webových serverů

Požadavky:
   - Python 2.2 nebo vyšší
   - Dostatečně velký diskový prostor pro uložení indexů (přibližně
     dvojnásobek velikosti zdrojových dat)

Více informací naleznete na scanley.org.

Děkuji,
-J. Novák
```

(Rady, jak oznamovat nové verze softwaru a jiné události projektu, naleznete v sekci **Publicita** v kapitole **6. Komunikace.**)

Ve světě svobodného softwaru už dlouho probíhá debata, zda je nutné začínat fungujícím programem, nebo zda může být pro projekt výhodné, když je otevřený už během etapy návrhu a diskuzí. Dříve jsem si myslel, že začít s fungujícím programem je ten vůbec nejdůležitější faktor, který odděluje úspěšné projekty od pouhých hraček; měl jsem za to, že vážné zájemce z řad vývojářů přitáhne jen software, který už něco konkrétního dělá.

Ukázalo se ale, že to není tak úplně pravda. V projektu Subversion jsme začali s dokumentem popisujícím návrh, s několika vývojáři, kteří měli o věc zájem a kteří spolu dobře komunikovali, s velkolepou reklamou a bez kousku běžícího kódu. K mému velkému překvapení projekt získal aktivní účastníky hned od samého začátku a v době, kdy už jsme měli alespoň něco, co by se dalo spustit, už s ním úzce spolupracovalo poměrně velké množství dobrovolníků. A Subversion není jediným příkladem – například projekt Mozilla byl také ohlášen, aniž by měl jakýkoliv spustitelný kód, a dnes je z něj úspěšný a populární webový prohlížeč.

Ve světle těchto důkazů jsem musel svůj předpoklad, že pro spuštění projektu je nutné mít běžící program, poněkud přehodnotit. Stále platí, že funkční program je tím nejlepším základem úspěchu a že je v zásadě dobrý nápad počkat s ohlášením projektu až do doby, než ho budete mít. Nicméně existují situace, kdy má smysl zveřejnit oznámení už dřív. Pořád si myslím, že přinejmenším dobře propracovaný dokument s návrhem nebo alespoň nějaká kostra celého programu jsou nezbytné. Samozřejmě je možné jej na základě zpětné vazby od veřejnosti poněkud upravit, ale vždy musíte mít něco konkrétního, něco hmatatelnějšího než jen pár dobrých nápadů, aby se toho někdo mohl chytit.

Ať už ale vydáte oznámení kdykoliv, nečekejte, že se k projektu ihned připojí hromada dobrovolníků. Výsledkem oznámení většinou je, že dostanete několik nezávazných dotazů, k mailing listu se připojí pár lidí a kromě toho všechno pokračuje v zásadě tak jako předtím. Časem si ale všimnete, že se objevuje čím dál více nových přispěvatelů i uživatelů. Oznámení je jen jakési zasazení semínka. Než se zpráva rozšíří, může to trvat dlouho. Pokud projekt bude důsledně odměňovat ty, co se do něj zapojí, šířit se bude, protože když někdo najde něco dobrého, chce se o to obvykle také podělit. Pokud vše půjde dobře, promění časem dynamika sítí s exponenciálně narůstající komunikací váš projekt v komplexní komunitu, ve které už nebudete znát každého člena jménem a nebudete schopni sledovat každou konverzaci. Následující kapitoly se zabývají tím, jak v takovém prostředí pracovat.