Metódy vývoja Open Source softvéru

JAROSLAV ZAJAC

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava

Abstrakt. Open Source je slovné spojenie v poslednej dobe často vystupujúce do popredia. V tejto eseji sa budem venovať spôsobom a metódam, akými je Open Source softvér vyvíjaný, pričom dôraz bude kladený na slovo metódy, nie Open Source. Tieto metódy sú s drobnými úpravami dobre využiteľné aj pri tzv. komerčnom vývoji softvéru, nakoľko nie sú kladené žiadne obmedzenia na účel a cenu vyvíjaného softvéru. Vyjadrím sa k aplikovateľnosti Open Source metód mimo Open Source komunity, pre ktorú sú typické isté odlišnosti od komunity bežných vývojárov, ako napríklad silná vnútorná motivácia prameniaca napríklad z použiteľnosti výsledného produktu pre samotného vývojára. Vyjadrím sa k zásadám Open Source, k vývoju a riadeniu Open Source projektov.

Úvod

Open Source je termín, ktorým boli pomenované tradície otvorených štandardov, zdieľaného zdrojového kódu a kolaboratívneho vývoja. V poslednej dobe získava veľkú pozornosť. Prečo? Asi najjednoduchšia odpoveď by mohla byť Internet. Internet, ako taký, umožňuje globálne distribuovaný vývoj produktov. Pozerá sa naň, ako na faktor umožňujúci vyvíjať softvér pri stiesňujúcich trhových podmienkach.

Vývoj softvéru cez Internet však nie je jednoduchý. Problém distribuovaného vývoja softvéru bol skúmaný. Väčšinou sa ale jednalo o adaptáciu konvenčných modelov. Open Source predstavuje alternatívny prístup vývoja softvéru, ktorý sa vyvinul priamo na Internete. Poskytuje informácie o bežných problémoch ako aj niekoľko možných riešení.

Problémom Open Source metód ale je, že neexistuje definitívna písomná formulácia týchto metód. Asi najlepšie je popísaná v práci E. Raymonda, The Cathedral and the Bazaar (všetky odkazy na túto prácu sú čerpané z [1], nakoľko pôvodnú prácu sa mi nepodarilo dostať do rúk). Tu sa ale nejedná o akademickú prácu, ale skôr o akýsi report z oblasti Open Source. Z hore spomenutého dôvodu môžu byť názory na to, čo vlastne Open Source metódy sú, odlišné, subjektívne. O tom, čo je to Open Source softvér, ako sa vyvíja, ako sa riadi a aké zásady vývoja Open Source softvéru boli sformulované, sa dočítate na nasledujúcich riadkoch.

.

Open Source softvér

Pojem Open Source môže reprezentovať filozofiu, spôsob vedenia obchodu, ako aj metodológiu vývoja softvéru. Ja sa budem zaoberať práve posledným spomenutým významom.

Open Source vývoj softvéru bol propagovaný ako spôsob vytvárania veľmi spoľahlivých produktov, ktoré lepšie plnia požiadavky používateľov.

Akúsi hrubú predstavu toho, ako vyzerá Open Source vývoj softvéru, poskytuje nasledujúca veta. Jednotlivec, alebo malá skupina ľudí vyvinie softvér, sprístupní ho na recenzovanie a úpravy na Internete, kontroluje zmeny kódu a s rastom projektu deleguje zodpovednosti. Tento prístup sa ukázal ako veľmi efektívny vo veľkom množstve projektov.

Bežné problémy, ktoré projekty riešia, sú distribuovaná koordinácia, spolupráca vo veľkom merítku a rýchly inkrementálny vývoj produktu.

Zásady Open Source

Hore spomenutá práca E. Raymonda bola kritizovaná, napríklad za to, že je príliš simplistická. Napriek tomu je považovaná za najlepšiu prácu o Open Source. Pán Raymond v nej mimo iné spísal akési body, podľa ktorých by sa Open Source vývoj softvéru mal riadiť. Voľný preklad a interpretáciu týchto bodov nájdete v tejto časti.

1. Každý dobrý softvér začína zaujatím vývojára.

Vývojári si môžu voliť projekty, na ktorých chcú pracovať. Toto je pre vývojárov motivácia, nakoľko si vyberajú to, čo ich baví, čo chcú robiť.

2. Dobrí programátori vedia, čo písať. Výborní vedia, čo prepísať (a znovupoužiť)

Dôležitosť znovupoužitia nie je pre Open Source vývoj softvéru jedinečná. Avšak Raymond poznamenáva, že zdieľanie zdrojových kódov uľahčuje znovupoužitie.

3. Plánuj niečo zahodiť. Aj tak to spravíš.

Túto myšlienku si Raymond požičal od Freda Brooksa z knihy The Mythical Man-Month. Vyzdvihuje to, že redizajn je často lepší ako prepracovanie zlého konceptu.

4. Ak máš správny prístup, zaujímavé problémy si ťa nájdu.

Implikáciou je, že projekt sa bude vyvíjať prirodzenejšie v prostredí, v ktorom sú zdieľané zdrojové kódy.

5. Keď stratíš záujem o program, tvojou poslednou povinnosťou je predať ho kompetentnému následníkovi.

Aby bolo znovupoužitie kódu efektívne, musí byť udržiavaný. Preto je potrebné, aby projekty mali niekoho zodpovedného za ne, ak pôvodný vlastník nemôže na ňom ďalej pracovať.

6. Správať sa k používateľom, ako ku spoluvývojárom je najjednoduchšia cesta k rýchlemu vylepšovaniu kódu a efektívnemu odstraňovaniu chýb.

Toto je jeden z kľúčových princípov Open Source vývoja softvéru. Projekty sa spoliehajú na používateľov, aby vyhodnocovali a vylepšovali softvér. Ďalšou výhodou tohto prístupu je, že používatelia ako spoluvývojári prinášajú do projektu vedomosti z cieľovej domény produktu.

7. Vydávaj skoro. Vydávaj často. Počúvaj svojich zákazníkov.

Ďalší kľúčový princíp. Pomáha zviditeľniť progres a umožní vytvoriť tesnú spätnú väzbu od používateľov.

8. Pri dostatočne veľkej skupine beta-testerov a spoluvývojárov bude akýkoľvek problém charakterizovaný rýchlo a oprava niekomu jasná.

Základná myšlienka je, že väčšia komunita nájde viac chýb. Tento prístup je jednoznačne rýchly. O jeho efektívnosti sa však dá polemizovať.

9. Rozumné dátové štruktúry a hlúpy kód fungujú oveľa lepšie, ako opačne.

Týmto chcel Raymond povedať, že dátové štruktúry pomáhajú pochopeniu pomocou abstrakcie.

10. Ak sa správaš k beta-testerom, ako k najdôležitejšiemu zdroju, budú reagovať tým, že sa stanú najdôležitejším zdrojom.

V podstate Raymond znovu zvýrazňuje význam používateľov ako spoluvývojárov, pričom poznamenáva, že úspech každého produktu závisí od záujmu a zaujatia jeho používateľov. Je potrebné, aby boli príspevky do projektu uznávané.

11. Druhá najlepšia vec hneď po tom mať vlastné nápady je rozoznanie dobrých nápadov od používateľov. Niekedy je to aj najlepšia vec.

Raymond vyzdvihuje význam spätnej väzby od kompetentnej skupiny používateľov. Veľká, pestrá komunita je dôležitá ako pre vývoj, tak aj pre odstraňovanie chýb. Viac ľudí ma viac dobrých nápadov a vyvíja lepšie riešenia.

12. Často najlepšie a najinovatívnejšie riešenia vznikajú z uvedomenia si, že koncept problému bol zlý.

Jedná sa v podstate o inú formuláciu 3. bodu. Vývojári by sa nemali báť odstrániť niektoré vlastnosti v záujme udržania čistého návrhu.

13. Perfekcia v návrhu sa nedosahuje tým, že už nie je čo pridať, ale skôr tým, že nie je čo odobrať.

Raymond sa odkazuje na dobre známy princíp navrhovania.

14. Ak chcete riešiť zaujímavé problémy, začnite tým, že nájdete problém, ktorý vás zaujíma.

V podstate ide o inú formuláciu 1. bodu. Vyzdvihuje myšlienku, že vývojár, ktorý si môže zvoliť, čo bude robiť, pracuje efektívnejšie, ako ten, ktorý si zvoliť nemôže.

15. Ak má koordinátor vývoja k dispozícii médium aspoň tak dobré, ako Internet a vie, ako viesť bez nútenia, viac hláv je jednoznačne lepších, ako jedna.

Bazárový model je závislý na Internete. Internet poskytuje prístup k veľkému množstvu potenciálnych prispievateľov, možnosť otvorenej komunikácie a decentralizovanú skupinovú štruktúru.

Raymond sa neobmedzil len na týchto 15 bodov. Zvyšné body však odchádzali od metodológie a boli formulované konkrétne pre fetchmail, projekt, ktorého vývoj

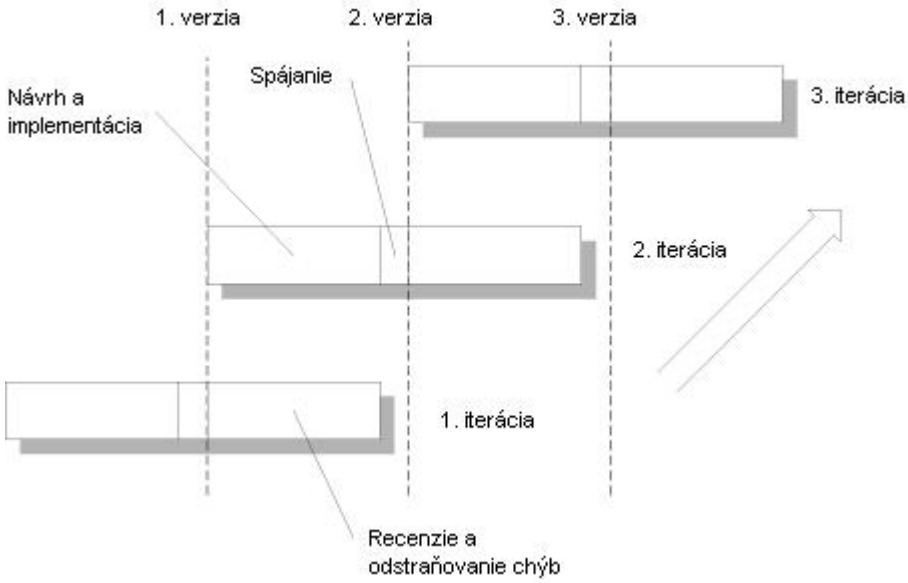
Raymond vo svojej práci popisoval. Z tohto dôvodu som ich do tejto časti nezaradil. Ich kompletný zoznam je možné nájsť v [1].

Čitateľovi isto neuniklo, že istá skupina týchto bodov sa opiera o veľkú komunitu. Preto je podľa mňa nevhodné sa pokúšať využiť tieto metódy pri malých tímoch, nakoľko sa strácajú niektoré z najväčších výhod tohto prístupu. Taktiež je niekedy ťažké niektoré tieto zásady aplikovať v uzavretom tíme. Napríklad problém vnútornej motivácie, na ktorú sa odvolávajú body 1 aj 14. Práca na Open Source projekte býva spravidla dobrovoľná, preto je vyriešenie problému motivácie základom úspechu projektu. Pokiaľ ale chceme efektívne využiť Open Source metódy pri komerčnej tvorbe softvéru, mal by každý člen tímu byť motivovaný. Je neefektívne, ak si polovica tímu povie, že ich daný projekt nezaujíma. Problém motivácie je v uzavretom tíme preto oveľa zložitejší. Tím si ťažko môže dovoliť podľa „nálady“ prijímať a prepúšťať členov, čo je v Open Source komunite v podstate nepodstatný a bežný jav.

Vývoj

Ako sa vlastne taký typický Open Source projekt vyvíja? Začína motiváciou. Táto motivácia má ale trochu netradičnú formu. Na začiatku jednotlivec alebo malá skupina ľudí vyvíja uzavretý prototyp, niekedy nazývaný *build 0*. Cieľom prototypu je ukázať, že projekt má pred sebou sľubnú budúcnosť, že má zmysel na ňom pracovať. Teda skutočne ide o to motivovať vývojárov k práci na projekte. Uzavretý je preto, lebo dlhoročné skúsenosti ukázali, že to je efektívnejšie, ako začínať iteratívne inkrementálne. Problémom pri uzavretom prototypovaní je kedy prototyp ukončiť a začať s ďalšou fázou, iteratívnym a inkrementálnym vývojom. Ideálny čas na tento prechod je, keď je prototyp funkčný, ale stále je čo vyvíjať. Takisto je dôležité, aby prototyp predstavoval akýsi celok, nie len náhodné zoskupenie zaujímavých funkcií.

Po ukončení prototypovania projekt prechádza do fázy iteratívneho a inkrementálneho vývoja. Pre Open Source je tento prístup najvhodnejší, nakoľko je dostatočne agilný, aby bez väčších problémov bolo možné do projektu zapracovať nové nápady. Obzvlášť výhodné to je priamo v Open Source komunite, kde nie je dané trvalé zloženie tímu pracujúceho na projekte a s novými ľuďmi môžu prísť nové nápady. V tejto fáze sa snaží väčšina projektov spĺňať zásadu č. 7, aby bolo možné udržiavať silnú spätnú väzbu od používateľov, ktorí malé inkrementy recenzujú, hodnotia a pripomienkujú.



Obr. 1. Paralelný vývoj projektu

Ďalšou metódou, ktorá sa pri Open Source používa je paralelný vývoj. Je snaha vykonávať čo najviac návrhových, implementačných a testovacích úkonov, s cieľom urýchliť vývoj a zvýšiť flexibilitu projektu. Väčšinou sa pre projekt vytvorí tzv. *To Do List*, z ktorého si vývojári vyberajú problémy, ktoré je potrebné vyriešiť. Potenciálne riešenia vytvorené vývojármi sú posielané vo forme zmenového súboru na určené miesta, odkiaľ si ich môže ktokoľvek vyzdvihnúť a recenzovať ich. S cieľom projekt synchronizovať sa v istých pravidelných časových intervaloch všetky hotové inkrementy spoja do uceleného vydania novej verzie produktu. Priebeh takéhoto vývoja je vidieť na obr.1. Ako je vidieť, spájanie projektu predstavuje istú prácu navyše. Riešia sa problémy typu konflikty v kóde, distribúcia a podobne. Čas vynaložený na spájanie je ale možné skrátiť napríklad použitím verziovacieho nástroja, prípadne pravidelným zamrazením kódu, prípadne zamrazením nejakej vlastnosti. Zamrazenie znamená, že v zamrazenej časti projektu už nie sú prijímané žiadne zmeny, snáď s výnimkou opravy závažných chýb.

Vyššie som spomenul recenzovanie inkrementov. Vzájomné recenzie inkrementov vývojármi predstavujú v Open Source nástroj zaručenia kvality produktu. Pomáhajú nachádzať a odstraňovať chyby z kódu. V Open Source komunite sú recenzie implicitné, nakoľko zdrojové kódy sú voľne prístupné a hlásenie chýb je verejné. Tento prístup okrem odstraňovania chýb má ešte aj motivujúci faktor, kedy vývojár v záujme získania, prípadne uchovania statusu kvalitného vývojára má snahu vydávať kvalitný kód. Recenziu môže napísať ktokoľvek na ktorýkoľvek zatiaľ neprijatý inkrement. Pri Open Source projektoch je ale recenzovanie v podstate jedinou používanou metódou kontroly kvality, čo je v istom kruhu odborníkov považované za

negatívum. Takisto je diskutabilné, či takáto neriadená forma recenzovania je efektívna. Rýchlosť metódy je nepopierateľná, ale čím väčší počet osôb recenzuje daný inkrement, tým viac človekohodín sa naň minie.

V [2] pán Lussier popísal modifikáciu pôvodného systému recenzií. V tejto modifikácii vývojár posielajúci časť kódu na recenziu určí jednu osobu, ktorá povinne musí recenziu napísať. Pre ostatných členov tímu sa jedná o nepovinnú činnosť. Týmto vývojár za predpokladu, že dobre pozná tím môže určiť najviac kompetentnú osobu na recenziu. Ďalšou zaujímavou úpravou bolo to, že recenzovanie malo prednosť pred písaním kódu. A poslednou úpravou pridali viac formalizmu do systému hlásenia chýb, ktoré bolo v podstate súčasťou písania recenzií, ale v pôvodnej metóde neexistovala žiadna dohoda o forme hlásenia chýb. Obsah hlásenia o chybe bol stanovený na presnú pozíciu chyby a jej detailný popis.

Riadenie

Riadenie Open Source projektov rieši hlavne problémy organizácie distribuovaných projektov. Pritom sa spolieha na decentralizovanú spoluprácu, vedenie na základe dôvery, internú motiváciu a asynchrónnu komunikáciu, neformálne plánovanie.

Pod decentralizovanou spoluprácou sa rozumie rozdelenie tímu, prípadne celej komunity, na menšie tímy. S Open Source projektmi sa často spája takzvaný prístup demokratického decentralizovaného tímu, kde v tíme nie je žiadny trvalý vedúci. Namiesto toho sa volia dočasní vedúci podľa vhodnosti na koordináciu rôznych úloh. V skutočnosti sa ale v praxi viac používa prístup kontrolovanej decentralizovanej skupiny, kde vedúci tímu koordinuje isté úlohy a existujú tzv. sekundárni vedúci, zodpovední za podúlohy. Aj keď ostáva vývojárom projektu istá voľnosť a samoriadenie, zároveň sa zachovávajú základné riadiace štruktúry. S rastom projektu a počtu vývojárov rastie potreba riadenia. Takisto s rastom projektu vzniká potreba obsadenia role integrátora riešení, aby sa odstránili konflikty, duplicitné riešenia a rozhodol, čo sa dostane do ďalšej verzie produktu. Preto sú Open Source projekty popisované, ako *benevolentné diktátorstvo,* kde vlastník projektu (jeho iniciátor) má právo robiť záväzné rozhodnutia, s tým, že majú odrážať záujmy komunity, prípadne tímu.

Vedúci pri Open Source projektoch nefungujú konvenčnými spôsobmi. Niekto sa môže stať vedúcim, ak ukáže, že k danému projektu veľa a kvalitne prispieva. Vplyv vedúcich sa vlastne získava dôverou stavanou na kompetencii. V zásade totiž platí, že programátori si vážia viac ľudí, u ktorých uznajú, že sú dobrí v tom, čo robia. Preto je ľahšie viesť tím programátorov ako výrečný programovací génius, než ako najrýchlejšie rozprávajúci predavač na svete. Bežnou praxou sa stáva, že pôvodca daného projektu sa stáva jeho hlavným vedúcim, benevolentným diktátorom a štruktúra sekundárnych vedúcich sa rozprestiera pod ním.

Asi najzaujímavejšou charakteristikou Open Source projektov je motivácia vývojárov. Vývojári Open Source projektov bývajú veľmi motivovaní. Táto motivácia vyplýva z toho, že Open Source projekty sa snažia vývojárov zaujať, ako je písané v zásadách č. 1 a 14 spomenutých vyššie. Sekundárne motivátory, ako peniaze, tu

nikdy nehrali dôležitú rolu. Väčšinou sa vývojári do projektov zapájali, lebo im to bolo umožnené, prípadne kvôli komunite, alebo statusu v komunite. Mimo Open Source komunity nájsť pre tím vnútorné motivátory podľa mňa pre manažérov a vedúcich predstavuje najväčší problém. Motivovať tím ale nie je dôležité len v prípade využitia Open Source metód. Jedná sa o všeobecný problém, ktorý podľa istých odborníkov pravdepodobne najviac ovplyvňuje kvality výsledného produktu.

Komunikácia prebieha asynchrónne. Dôvod takejto komunikácie je zrejmý. Keďže Open Source projekty sú vyvíjané na báze dobrovoľnosti, aby bola zabezpečená čo najväčšia účasť vývojárov na projekte, nesmú sa klásť geografické obmedzenia. Asynchrónnou elektronickou komunikáciou vo forme *e-mailu* a *mailing listov* je znížená organizačná záťaž.

Zaujímavosťou Open Source vývoja je, že plánovanie je neformálne. Pre projekt sa stanoví jeden veľký, abstraktný cieľ: zlepšiť produkt. Prinajlepšom sa určia míľniky určujúce hrubý časový odhad vývoja. Výhodou je, že „neexistujúci" plán nie je potrebné meniť, ak počas vývoja príde niekto s novým nápadom. Tým získavajú Open Source projekty istú flexibilitu. Toto je v konsenze s ostatnými metódami, ktoré sú tiež využívané s dôrazom na dynamičnosť, agílnosť a flexibilitu. Nevýhodou je problém s odhadom trvania vývoja produktu, problematický výpočet nákladov. Toto ale v Open Source komunite nepredstavuje problém.

Záver

Open Source metódy vývoja softvéru vznikli a používajú sa v pomerne špecifickom prostredí komunity, ktorá projekty vyvíja na báze dobrovoľnosti. Preto je podľa mňa nemožné využiť pri projekte mimo Open Source komunity pôvodné Open Source metódy bez istých úprav.

Problémy, ktoré je potrebné riešiť sú náhrada za vnútornú motiváciu prameniacu z istých zdrojov neprítomných mimo Open Source komunity, ako napr. status v komunite, osobné využitie vyvíjaného softvéru a podobne. Taktiež je pre projekt, obzvlášť komerčný, problematické neformálne plánovanie, sťažujúce až znemožňujúce časový a nákladový odhad projektu.

Väčšina ostatných metód a zásad je ale veľmi dobre aplikovateľná aj mimo Open Source komunity. Výhody sa prejavia napr. v znížení času vynaloženého na riadenie projektu, vo vhodnejšom delení práce skrz *To Do List,* z ktorého si každý vyberie to, čo mu najviac vyhovuje.

Použitá literatúra

1. Johnson, K.: *A Descriptive Model for Open-Source Software Development*. University of Calgary, Department of Computer Science, 2001.
2. Lussier, S.: New Tricks: How Open Source Changed the Way My Team Works. *IEEE Software*, Vol. 21, No. 1 (January/February 2004) 68-72.

Annotation

Open Source software development methods

Open Source is a collocation getting more and more attention lately. This essay will be dedicate to the methods, with which Open Source software is being developed. The emphasis is on the word method, not Open Source. These methods can be used with some small addjustments even in so called commercial software development, because there are no limitations on the purpose and cost of the developed software. I will deal with the tenets of Open Source, the development and management of Open Source projects. I will deal with the applicability of Open Source methods outside of the Open Source comunity, which differences in some points from the typical developer comunity, like strong internal motivation originating for instance from the usability of the final product to the developer.