Komerčné verzus nekomerčné metódy vývoja

MARTIN JENČO

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

Abstrakt. Vývoj softvéru je dlhý a náročný proces. Pre komerčný vývoj softvéru boli vyvinuté rôzne metódy, ktoré majú za úlohu zvyšovanie efektivity vývojárov a manažérov a znižovanie nákladov potrebných na vývoj softvérového produktu. Postupom času sa vykryštalizovali rôzne metódy vývoja aj v nekomerčnej sfére. Prekvapením môže byť, že niektoré metódy používané Open-Source komunitami (nekomerčná sféra) sú časovo efektívnejšie, ľahšie manažovateľné a menej náchylné na chyby ako mnohé komerčne využívané metódy. Ich použitie však nie je vôbec limitované na nekomerčnú sféru. V tejto práci sú porovnané najpoužívanejšia metódu nekomerčnej sféry s metódami komerčnej sféry z hľadiska manažovateľnosti, efektivity a celkovej použiteľnosti.

Úvod

Vývoj rozsiahlych softvérových produktov nie je vôbec jednoduchá záležitosť. Hlavne vývoj väčších a zložitejších systémov prináša so sebou množstvo problémov, ktoré sa pri vývoji menších produktov nevyskytujú. Medzi takéto problémy patria hlavne problémy s manažovateľnosťou projektov, správa verzií (testovacia verzia, verzia pre zákazníka, ...) a mnohé iné.

Pri komerčnom vývoji sa využívajú metódy, ktoré sa snažia predísť, resp. minimalizovať spomínané problémy. Okrem komerčnej sféry je však softvér vyvíjaný aj nekomerčne, zdrojové kódy sú verejne dostupné a ktokoľvek môže prispieť pri vývoji. Takýto spôsob vývoja softvéru sa nazýva vývoj s otvoreným zdrojovým kódom programu (*Open-Source*). Už na prvý pohľad je jasné, že pokiaľ by ktokoľvek mohol bez akejkoľvek kontroly pridať ľubovoľný kód do zdrojových kódov, bol by projekt veľmi rýchlo odsúdený na zánik. Preto sa časom vyvinuli a zaužívali metódy vývoja Open-Source projektov, pri ktorých je každý pridávaný kód kontrolovaný, čím sa taktiež odhalí väčšina chýb ešte pred ich pridaním do zdrojových kódov.

Manažment v softvérovom inžinierstve, máj 2005, s. 1-8.

V tomto dokumente opíšem model vývoja softvéru používanom pri Open-Source projektoch, porovnať ho s metódami používanými v komerčnej sfére a zhodnotiť jeho kladné a záporné stránky.

Organizácia projektu

V komerčnej sfére sa najčastejšie používa hierarchický model organizácie projektu. Projekt je rozdelený na menšie podprojekty, kde za každý podprojekt je zodpovedná iná osoba. Väčšie podprojekty sa podobným spôsobom rozdelia na ešte menšie podprojekty, kde sa zodpovednosť taktiež rozdelí na skupinu ľudí. Keď už sú podprojekty rozumného rozsahu, začnú tímy vývojárov pracovať na projekte. Postupne ako pribúda funkcionalita jednotlivých podprojektov, stáva sa funkčným aj celkový projekt. Vývojári sú zodpovední za funkčnosť vyvinutých súčastí, zatiaľ čo vedúci tímov a manažéri sú zodpovední za správnosť a funkčnosť celého podprojektu, ktorý majú na starosti.

Pri použití takéhoto modelu je veľmi dobrá manažovateľnosť, nakoľko za každý celok, za každú súčasť je zodpovedný niekto konkrétny, kto priebežne informuje o tom, ako postupuje vývoj. Trochu problematickejšia je však komunikácia medzi jednotlivými tímami. Nie je zriedkavé, že tím vývojárov zodpovedný za vývoj jednej časti zistí chybu v inej časti projektu, za ktorú nie sú zodpovedný, ale spolupracujú s ňou. Alebo v prípade, že potrebujú, aby bola v inej časti projektu pridaná nová funkcionalita potrebná pre výsledný produkt resp. jeho časti. V takýchto prípadoch oznámia vývojári svoju požiadavku vedúcemu tímu, ktorý ju oznámi vedúcemu vyvíjaného celku, ten ju následne oznámi osobe zodpovednej za celý podprojekt a tak ďalej, až sa správa dostane k osobe, ktorá je zodpovedná za obidva celky. Tá ju následne odovzdá svojmu podriadenému, ktorý je zodpovedný za spomínaný celok, ten svojim podriadeným až sa správa dostane k druhému tímu vývojárov. Takýto spôsob komunikácie je už na prvý pohľad neefektívny a zdĺhavý. V praxi sa často tento spôsob komunikácie obchádza a tímy komunikujú priamo. Tým sa však znižuje manažovateľnosť projektu, nakoľko nadriadení, zodpovední za celok sa dozvedia o zmenách až dodatočne, prípadne vôbec. Tým sa taktiež ohrozuje funkčnosť celého projektu, nakoľko vývojársky tím nemusí vedieť podrobnosti o ostatných častiach a samoiniciatívnymi úpravami môžu napáchať viac škody ako úžitku.

Pri Open-Source projektoch neexistuje žiadne hierarchické usporiadanie, ale taktiež nie všetci majú rovnaké možnosti ovplyvniť, akým smerom sa bude projekt uberať. Ktokoľvek sa môže pridať k tímu vývojárov a môže navrhovať zmeny zdrojových kódov. Zmeny však môže iba navrhovať, nemôže ich však presadiť. Každý návrh na zmenu je zverejnený spoločným komunikačným kanálom, napríklad mailing-listom. Počas istého časového intervalu sa môže ktokoľvek vyjadriť k funkčnosti, efektívnosti, bezpečnosti, programátorskému štýlu, prípadne čomukoľvek ohľadne navrhovanej zmeny. Navrhovateľ zmeny môže prijať konštruktívnu kritiku a upraviť návrh svojej zmeny tak, aby spĺňal všetky predpoklady na správny, funkčný, efektívny

a dobre napísaný kód. Ani keď nikto nevyjadrí negatívny postoj k návrhu, nie je isté, že zmena bude akceptovaná a pridaná do projektu. Rozhodujúce slovo má osoba zodpovedná za celý projekt, prípadne za časť projektu. Pokiaľ sa rozhodne potvrdiť úpravu, je návrh prijatý a vykoná sa úprava zdrojových kódov projektu. Pokiaľ sa však rozhodne neprijať úpravu, tak sa úprava nedostane do výsledného kódu projektu.

Princípy Open-Source metódy

V tejto kapitole opíšem základné princípy metód vývoja používaných v Open-Source projektoch podľa [1].

Malé úpravy

Základom metódy vývoja používanej pri vývoji nekomerčných projektov sú malé úpravy. Malé zmeny sú vykonávané vo forme malých úprav, veľké zmeny sú taktiež vykonávané vo forme viacerých malých úprav.

Výhodou malých úprav je dobrá prehľadnosť, dobrá kontrolovateľnosť pridávaných úprav a malá pravdepodobnosť spôsobenia veľkej, závažnej chyby. Malé úpravy sa ľahko kontrolujú a na ich pochopenie nie je väčšinou potrebný dlhý čas. Tento čas je ešte možné znížiť vhodne zvolenými komentármi, dokumentáciou a vhodne volenými názvami premenných, funkcií a podobne. Čo sa týka chybovosti, tak pri kontrole malých úprav je oveľa ľahšie všimnúť si chybu alebo nedostatok, než pri veľkých úpravách.

Rozdelenie úloh

Ako som už spomínal v časti *Organizácia*, i napriek tomu, že v Open-Source metódach sa nepoužíva štandardný hierarchický model, existuje tu isté rozdelenie úloh [2]. Najpočetnejšou a z hľadiska vývoja produktu nezanedbateľnou skupinou je tím vývojárov. Vzhľadom na povahu Open-Source projektov sa môže hocikto zaradiť do skupiny vývojárov a pomáhať pri vývoji. Vývojári sa neformálne delia na dve skupiny, z ktorých jedni vytvárajú a pridávajú nové funkčnosti, zatiaľ čo druhí pracujú na odstránení známych chýb.

Druhou skupinou, dôležitou hlavne z dôvodu zabránenia šírenia chýb v projekte je skupina kontrolórov. Táto skupina má za úlohu kontrolu navrhovaných zmien. V prípade, že vývojár považuje svoju úpravu za pripravenú na vloženie do produktu, zverejní ju prostredníctvom spoločného komunikačného kanála a vtedy prichádzajú na radu kontrolóri. Kontrolóri preskúmajú navrhovanú úpravu z hľadiska správnosti, funkčnosti, bezpečnosti, prehľadnosti a podobne. Svoje pripomienky zverejnia v už spomínanom spoločnom komunikačnom systéme. Vývojár môže uznať vhodnosť pripomienky a prepracovať svoj návrh na úpravu.

Treťou a poslednou skupinou je skupina potvrdzovateľov. Potvrdzovateľ je osoba zodpovedná za celý projekt, prípadne jeho časť a má rozhodujúce slovo, či bude úprava prijatá do projektu, alebo nie.

Jednotlivé skupiny úloh nie sú navzájom sa vylučujúce. To znamená, že vývojár môže byť aj kontrolór, kontrolór môže byť vývojár ba dokonca jedna osoba môže byť vo všetkých troch skupinách súčasne.

V Open-Source projektoch je štandardné rozdelenie úloh také, že osoba zodpovedná za projekt (ktorá projekt vymyslela, založila, stará sa o projekt a pod.) prípadne osoby, ktoré tým poverila sú v pozícií potvrdzovateľov. Skupiny vývojárov a kontrolórov sú verejné, čo znamená, že hocikto má možnosť pridať sa do ľubovoľnej z nich, prípadne obidvoch.

Kontrola kódu

Veľmi dobrým a spoľahlivým spôsobom hľadania a odstraňovania chýb je kontrola kódu. Pri kontrole kódu sa podľa štatistických výskumov odhalí 60-90% všetkých chýb, čo je určite nezanedbateľné množstvo.

Kontrola kódu znamená, že niekto iný, ako autor kódu sa pokúsi pochopiť, čo a ako má kontrolovaný kód robiť, pri čom sa snaží zobrať do úvahy všetky prípady, ktoré môžu nastať a kontroluje, či sa v kóde nevyskytne chyba.

Čím viac ľudí kontroluje kód, tým je väčšia pravdepodobnosť odhalenia chýb. V Open-Source projektoch v závislosti od rozsiahlosti projektu kontroluje kód rádovo pár desiatok až pár desiatok tisíc ľudí. Všetci majú možnosť vyjadriť sa ku kontrolovanému kódu, prípadne navrhnúť jeho zmeny, úpravy a modifikácie.

Potvrdzovanie úprav

Úpravy sa do výsledného produktu dostávajú až potvrdením. Potvrdenie je proces, kde osoba zodpovedná za projekt (potvrdzovateľ) potvrdí vhodnosť a správnosť úpravy a pridá ju do výsledného produktu.

Čo sa týka vykonávania zmien výsledného produktu, má vždy posledné slovo potvrdzovateľ. I v prípade, že navrhovaná úprava prejde kontrolou kódu bez vážnejších pripomienok, má potvrdzovateľ možnosť rozhodnúť sa nezaradiť úpravu do výsledného produktu.

Vždy funkčná verzia

Dôležitým faktorom je, aby výsledný produkt bol vždy funkčný. Toto je možné dosiahnuť vďaka už spomínaným malým úpravám. Všetky úpravy musia mať taký charakter, aby neobmedzili celkovú funkčnosť výsledného produktu.

Tým sa zabráni tomu, aby bola posledná aktuálna verzia produktu nepoužiteľná, lebo je potrebné pridať ešte ďalšie nedorobené súčasti. Je lepšie mať funkčnú verziu, ktorá ešte neponúka všetky možnosti, ako mať verziu, ktorá ponúka všetky možnosti, ale je z nejakého dôvodu nepoužiteľná.

Podrobná dokumentácia

Dokumentácia je neoddeliteľnou súčasťou vývoja a bez dobrej dokumentácie je projekt pravdepodobne odsúdený na rýchly zánik.

Každá vykonaná úprava musí byť riadne okomentovaná a zdokumentovaná. Každá úprava v závislosti od závažnosti úpravy ovplyvní subverziu, prípadne verziu komponentu, resp. celého produktu. V prípade uvolnenia vydanej verzie je potrebné podrobne zaznamenať verzie všetkých súčastí a komponent, z ktorých sa výsledný produkt skladá.

Paralelný vývoj

Čas sú peniaze. Túto myšlienku možno s menšou obmenou aplikovať aj na nekomerčné Open-Source projekty. V Open-Source projektoch nejde o peniaze, ale projekty, ktoré sú vyvíjané tak dlho, že keď konečne uzrú svetlo sveta, sú už k ničomu, sú zbytočne vynaloženým úsilím.

Hlavným faktorom ovplyvňujúcim rýchlosť vývoja projektu je počet ľudí, ktorí na projekte pracujú. Pokiaľ však nemôžu pracovať paralelne, tak projekt postupuje veľmi pomaly. Práve vďaka už spomínaným malým zmenám je možné, aby na projektoch využívajúcich Open-Source metódy vývoja pracovalo viacero ľudí súčastne.

Ak sa jeden vývojár, resp. tím vývojárov rozhodne zásadne prepracovať časť projektu, tak do okamihu ako vytvoria použiteľný kód, nemôžu ostatní pracovať na tej istej časti. Avšak vzhľadom na to, že všetky úpravy sú vykonávané vo forme malých a častých úprav, je čas zdržania pri paralelnom vývoji zanedbateľný.

Iteratívny a inkrementálny vývoj

Ako už bolo spomínané v časti rozdelenie úloh, vývoj sa skladá z pridávania novej funkčnosti (inkrementálny vývoj) a z opravy známych chýb (iteratívny vývoj).

Iteratívnym vývojom sa vývojári snažia dosiahnuť bezchybnú a plne funkčnú verziu produktu. O tom, či je možné, alebo nie je možné dosiahnuť 100% bezchybnú verziu sa dá polemizovať. Podstatné je však, že je možné sa iteratívne (postupným približovaním) približovať bezchybnej verzii. Vývojári pracujúci na odstraňovaní chýb využívajú zoznam chýb, nazývaný *bug-list*. V tomto zozname sú uvedené všetky známe chyby a môže tu byť uvedená aj závažnosť chyby. Je logické, že najskôr sa pracuje na odstránení závažných chýb a až potom sa venuje čas odstraňovaniu chýb menej závažných.

Inkrementálny vývoj slúži na pridávanie novej funkčnosti do projektu. Podobne ako pri iteratívnom vývoji, aj v inkrementálnom vývoji existuje zoznam úloh, na ktorých by bolo vhodné pracovať. Tento zoznam sa nazýva *Todo-list*. Todo-list je vytváraný prevažne na základe ohlasu spotrebiteľa produktu, prípadne na základe ohlasu vývojárov.

Výhody

V predchádzajúcej kapitole som naznačil isté výhody modelu vývoja používaného pri Open-Source projektoch. V tejto kapitole sú zosumarizované najdôležitejšie výhody sú popísané, do akej miery sa môžu odraziť na úspešnosti softvérového projektu [2].

Časová úspora

Čas je jedným z rozhodujúcich faktorov úspešnosti projektu. Vo väčšine prípadov platí, že čím skôr je projekt hotový, tým lepšie. Taktiež v komerčnej sfére v zásade platí, že čím dlhšie sa pracuje na projekte, tým je projekt drahší.

Časová úspora opisovanej metódy spočíva prevažne v tom, že väčšina chýb sa objaví vo veľmi skorom štádiu. V závislosti od počtu kontrolórov kódu sa pri kontrole odhalí 60 – 90% všetkých chýb. A pri vývoji softvéru platí, že čím skôr sa chyba objaví, tým menej času a úsilia treba na jej nápravu. Podľa prieskumu uvedenom v [1] vyplýva, že i v prípade, že každú úpravu kontrolujú ďalší traja ľudia, celkový vývoj vyjde lacnejšie, ako v prípade, keď sa kód nekontroluje.

Rýchle odozvy na požiadavky

Ďalším dôležitým faktorom je pružnosť projektu. Pružnosť závisí od toho, ako rýchlo je projekt schopný reagovať na požiadavky.

Keďže pri spomínanej metóde je stále udržiavaná funkčná posledná verzia a sú vykonávané iba malé, časovo nenáročné úpravy, je možné v prípade potreby veľmi pružne zareagovať na požiadavku s vysokou prioritou. Príkladom môže byť napr. oprava závažnej bezpečnostnej chyby, ktorá má bezpochyby vysokú prioritu. V prípade, že by neboli vykonávané drobné úpravy, ale veľké úpravy a chyba by sa vyskytla práve v časti, ktorá sa prerába, bolo by potrebné s novou, opravenou počkať na dorobenie celej časti. Pri drobných úpravách však nie je potrebné čakať na dokončenie celého celku a je možné chybu opraviť takmer okamžite.

Chyby sa nedostanú do vydanej verzie

Chyby sú veľmi zlou vizitkou celého projektu a spôsobujú veľmi veľa problémov. Ideálny stav = nerobiť žiadne chyby sa prakticky nedá dosiahnuť. Keďže sa chybám nedá vyhnúť, otázkou je iba to, kedy a ako chyby zistíme. Pre dobré meno projektu je určite lepšie, keď chybu odhalia vývojári, prípadne kontrolóri, ako keby mal chybu nájsť zákazník / používateľ produktu.

Čím neskôr zistíme chybu, tým viac času a úsilia musíme vynaložiť na to, aby sme ju odstránili. Open-Source metóda sa v značnej miere sústredí na skoré odhaľovanie chýb. Práve vďaka kontrole kódu, ktorá je vykonávaná nad každou navrhovanou zmenou sa prevažná väčšina chýb zistí a odstráni ešte pred tým, než je pridaná do vydanej verzie.

Kontrola šírenia chýb

Ak sa predsa len voľajaká chyba dostane do vydanej verzie, je veľmi užitočné vedieť, ktorých verzií sa chyba týka a čo všetko mohla chyba ovplyvniť.

Pokiaľ máme zdokumentovanú každú vykonanú úpravu, je relatívne jednoduché zistiť, do ktorých verzií sa chyba rozšírila. Následnou kontrolou úprav vykonaných po vzniku chyby je možné zistiť, či sa vplyv chyby nerozšíril aj do iných častí projektu.

Nevýhody

Okrem výhod má spomínaná metóda samozrejme aj nevýhody. V tejto kapitole sa zameriam na najdôležitejšie z nich [2].

Prototyp / vydanie

Keďže vždy je k dispozícií funkčná verzia, je potrebné rozhodnúť, kedy bude verzia označená za vydanú verziu, označiť ju príslušným číslom verzie a dať ju k dispozícií používateľom.

Nie vždy je ľahké zhodnotiť a rozhodnúť, kedy nastal pravý čas na označenie verzie za vydanú. Je potrebné zhodnotiť závažnosť a množstvo vykonaných úprav.

Nepodarený kód

Pri každom väčšom projekte je potrebné veľmi dôsledne sledovať a kontrolovať vykonávané úpravy a zmeny kódu, prípadne nový pridávaný kód.

Okrem funkčnosti a efektívnosti je dôležitý aj programátorský štýl použitý pri vývoji. Pokiaľ na vývoji pracuje veľké množstvo vývojárov, a keďže každý rozmýšľa mierne odlišne, môže sa veľmi ľahko stať, že kód sa postupne stane neprehľadný, neohrabaný a veľmi ťažko modifikovateľný. A práve potvrdzovateľ má za úlohu zabrániť vzniku takéhoto nepodareného kódu. Aby sa zabránilo vzniku nepodareného kódu má potvrdzovateľ možnosť rozhodnúť, že zmena sa do projektu nepridá i napriek tomu, že funguje.

Veľké projekty

Opisovaná metóda je takmer ideálna pre malé projekty, pri väčších však môžu nastať problémy s manažovateľnosťou celého projektu.

Pokiaľ je potvrdzovateľ iba jeden, prípadne malá skupina spolupracujúcich osôb, nie je s manažovateľnosťou projektu veľký problém. Pokiaľ je však projekt takého rozsahu, že malá skupina ľudí nezvláda potvrdzovať všetky vykonávané úpravy, je potrebné vytvoriť v skupine potvrdzovateľov hierarchickú štruktúru.

Záver

I napriek tomu, že Open-Source je veľmi často vyvíjaný zadarmo, nie je vyvíjaný iba nadšencami, amatérmi a študentmi. Veľké množstvo vývojárov pracujúcich na Open-Source projektoch sú profesionálni vývojári vyvíjajúci softvér aj v komerčnej sfére.

Spomínaná metóda však nie je obmedzená iba pre použitie v Open-Source projektoch, ale s menšími úpravami je použiteľná i v komerčných projektoch.

Použitá literatúra

1. Stephane Lussier, Macadamian technologies: *New Tricks: How Open Source Changed the Way My Team Works.* Focus: developing with open source software, IEEE software.
2. MSc Thesis Kim Johnson: *A Descriptive Process Model for Open-Source Software Development*. University of Calgary, June 2001, http://sern.ucalgary.ca/students/theses/KimJohnson/toc.htm (anglicky).

Annotation

Commerce versus non-commerce software development methods

Software development is long and difficult process. There are many methods for commerce software development. They helps to increase developers and managers efficiency and decreasing software development costs. By the time, some methods were developed also in non-commerce sphere. Some methods used by Open-Source communities are more efficient, easier to manage and more error proof than some methods used in commerce sphere. Usage of this methods is not limited only for non-commerce projects. This paper try to compare mostly used non-commerce method with other, commerce methods.