VERZIOVANIE V MALOM

Mýliť sa je ľudské, ukladať chybné verzie programátorské.

Jakub Kříž

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jacob.kriz[zavináč]gmail[.]com

Abstrakt. *Verziovacie systémy sú v praxi bežne používané na správu zdrojového kódu v tímoch s viacerými programátormi. Používanie týchto systémov v malých alebo dokonca jednočlenných tímoch však nie je samozrejmé, pretože programátor-jednotlivec v princípe nepotrebuje správu zdrojového kódu. Ich použitie však prináša viacero menej očividných výhod, ktoré pomôžu k vytvoreniu lepšieho softvérového produktu. Verziovacích systémov existuje veľké množstvo a delia sa na dve výrazne odlišné skupiny – centralizované a distribuované. Výber správneho systému sa môže líšiť od jedného projektu k druhému. V tejto eseji opisujem moje názory na výber správneho typu a použitie verziovacieho systému jednotlivcami a tímami s menším počtom programátorov.*

Kľúčové slová: *podpora vývoja, verziovacie systémy, jednočlenné tímy, malé tímy*

Úvod

Verziovacie systémy sú dlhodobo používané vo všetkých oblastiach vývoja softvéru. Výhody ich používania sú pre väčšinu tímov samozrejmé. Ak viacero programátorov pracuje na rovnakých častiach zdrojového kódu, potrebujú nástroj, ktorý im pomôže tento kód spravovať. Títo programátori môžu pritom byť od seba vzdialení tisíce kilometrov, ako je to dnes bežné v mnohých firmách alebo v prípade open source projektov. Jedinou inou možnosťou by bolo, aby každý programátor písal zdrojový kód do samostatných súborov, čo je ale v praxi nemožné.

Ak však programátor pracuje na projekte samostatne alebo iba v tíme niekoľkých ľudí, ktorí pracujú na oddelených častiach projektu, môže sa zdať, že nebude potrebovať

žiadne podporné prostriedky pre vývoj softvéru a konkrétne ani verziovací systém. Výberom a používaním správneho verziovacieho systému však podľa môjho názoru programátor získa značné výhody, ktoré mu napomôžu získať lepší prehľad nad projektom a teda k vytvoreniu lepšieho softvérového produktu.

Problém môže vzniknúť aj pri výbere verziovacieho systému. Tieto systémy sa zvyčajne delia na dva druhy: centralizované a distribuované. Oba štýly poskytujú svoje výhody aj nevýhody a ich výber nemusí byť vždy jednoznačný. Staršie a aj historicky viac používané boli centralizované systémy, no pre možné výhody distribuovaných systémov je tento typ v súčasnosti používaný v stále rastúcom počte projektov [1].

Verziovacie systémy sa delia aj na iné typy, napríklad na Lock-Modify-Unlock a Copy-Modify-Merge, no pre zjavné výhody druhého typu oproti prvému je ich výber v súčasnosti pomerne jasný [3]. V prípade použitia prvého typu systému by na jednom súbore so zdrojovým kódom mohol pracovať iba jeden programátor. Toto je prinajmenšom značne nepraktické a pri trochu väčších projektoch podľa môjho názoru priamo nefunkčné.

Sám sebe pánom

Najmenším možným tímom je tím jednočlenný. Na prvý pohľad sa môže zdať, že programátor, ktorý pracuje na projekte samostatne nepotrebuje nástroje na podporu vývoja. Tieto nástroje sú bežne chápané ako nástroje pre organizovanie viacerých programátorov a na zdieľanie ich práce. Tak, ako jednotlivec nemusí komunikovať sám so sebou pomocou e-mailov, Skype a podobných nástrojov, zvyčajne si myslí, že nepotrebuje ani zložitú správu zdrojového kódu.

Používanie verziovacieho systému však podľa môjho názoru prináša značné výhody aj pre programátora-jednotlivca. Zrejmou výhodou pre každého programátora je samotné rozdelenie softvéru na verzie. Pre vývoj softvéru sa nehodí si jednoducho ukladať všetky zmeny do rovnakého súboru bez možnosti obnovenia predchádzajúcej verzie. Pridanie novej funkcionality často znamená úpravu mnohých zdrojových súborov. V prípade, že tieto zmeny spôsobili nečakané chyby a potrebujeme ich rýchlo odstrániť alebo v prípade, že sa rozhodneme danú funkcionalitu do projektu nezahrnúť je prechádzanie týchto súborov nepraktické a pomalé.

V praxi sa mi niekoľkokrát stalo, že som urobil a nasadil nevhodné zmeny. Na projekte som pracoval dlho do noci a keď som konečne dokončil funkcionalitu, na ktorej som pracoval, nechcelo sa mi skontrolovať ostatné časti aplikácie a zmeny som jednoducho nasadil. Na druhý deň som zistil, že táto nová funkcionalita znefunkčnila ostatné dôležité časti aplikácie. V tejto situácii som potreboval aplikáciu čo najrýchlejšie dostať do predchádzajúceho, plne funkčného stavu. Vtedy som žiaľ nepoužíval žiadny verziovací systém a tak bolo potrebné prejsť všetky upravované súbory a ručné prepísať zmeny. Toto bolo nielen veľmi pracné, ale aj náchylné na chyby.

Pomocou verziovacieho systému sa však vieme jednoducho vrátiť k predchádzajúcej verzii bez problematických zmien [2]. Zmeny z novšej verzie nám zostanú zachované a môžeme na nich neskôr znovu pracovať. Stále používanie verziovacieho systému by mi určite ušetrilo veľa času a nervov.

Menej očividnou výhodou neustáleho ukladania zmien je možnosť ich sledovania. Viacero verziovacích systémov núti používateľa zadať opis zmeny, ktorú práve vykonal. Toto programátora prinúti zamyslieť sa nad týmito zmenami. Boli tieto zmeny naozaj užitočné? Boli potrebné? Čo potrebujem vykonať ako ďalší krok? Odpovede na tieto otázky pomôžu získať lepší pohľad na celý projekt [4]. Pri práci na vlastnom projekte, ktorého vývoj nemá prográmator perfektne naplánovaný sa môže stať, že vykoná nezmyselné alebo nekompletné zmeny. Často sa mi stane, že teste pred zadaním vykonanej zmeny si uvedomím, že by do nej bolo potrebné ešte niečo dorobiť alebo, že som danú zmenu mohol vykonať inak. Výsledkom je, že programátor trávi svoj čas užitočnejšie a každý nasledujúci pohyb si dôkladne premyslí.

Ak sa teda programátor-jednotlivec rozhodne nejaký systém použiť, jeho výber je podľa môjho názoru jednoduchý. Takýto typ používateľa potrebuje distribuovaný systém, ktorého repozitár sa bude nachádzať na počítači, na ktorom bežne pracuje. Keďže nepotrebuje jeho prácu zdieľať s ostatnými ľuďmi, nepotrebuje vždy uchovávať najnovšiu verziu na centrálnom serveri. Osobne v praxi používam distribuovaný verziovací systém Git. V prípade potreby si prácu môžem preniesť na iný počítač aj pri používaní tohto distribuovaného systému.

Programátor jednotlivec môže zvážiť aj použitie centralizovaného systému. Výhody takéhoto systému vidím v stálom uchovávaní najnovšej verzie na serveri, teda ak programátor bude často meniť zariadenia, na ktorých pracuje, môže byť použitie takého systému pre neho výhodné. Takéto podmienky sa mi však nezdajú pravdepodobné a preto považujem distribuovaný systém za jednoznačnú voľbu. Výber je však na samotnom programátorovi. Každý musí zvážiť, či je takýto scenár pravdepodobný a rozhodnúť sa samostatne.

Dobrých ľudí sa všade veľa zmestí

Verziovacie systémy sú prakticky nevyhnutné pre všetky typy tímov, aj tie menšie. V prípade, že členovia tímu pracujú na nezávislých častiach projektu získajú rovnaké výhody, ako som popísal v predchádzajúcej časti eseje.

Ak potrebujú pracovať na rovnakých častiach zdrojového kódu, systém, ktorý im umožní spravovať tento kód je nutnosťou. Inak by sa mohlo stávať, že si budú progamátori zmeny navzájom prepisovať. Aby tomu zabránili, potrebovali by vynaložiť značné úsilie na réžiu zmien. Keď som začínal programovať nepoznal som žiadne verziovacie systémy. S kamarátom sme sa rozhodli robiť na menšom projekte. Napriek tomu, že sme boli maličký tím pozostávajúci z dvoch členov, zdieľanie zdrojového kódu na serveri vyvýjanej webstránky bolo veľmi náročné a často sme nezabránili konfliktom a chybám, ktoré sme následne museli pracne odstraňovať.

Čo sa týka typu systému, všeobecne sa distribuované systémy považujú za šikovnejšieho nástupcu centralizovaných, ktoré majú oproti nim viacero výhod [1]. Výber typu systému je však podľa mňa aj tak problém s nejednoznačnou odpoveďou a optimálny výber sa môže líšiť od projektu k projektu. Môj názor na výber záleží od typu projektu a kontextu, v ktorom tím pracuje.

Školský projekt

Pri školskom projekte, ako je napríklad projekt, na ktorom pracujeme v rámci predmetu Tímový projekt, je vhodnejší distribuovaný systém, pretože študenti pracujú na softvéri samostatne a na serveri nepotrebuje byť neustále najnovšia verzia.

Na projekte pracujeme agilne, metódou Scrum, ktorá má jasne definované šprinty čiže obdobia, po ktorých sa skontroluje stav softvéru. Študenti môžu na ich úlohách pracovať samostatne vo svojom vlastnom repozitári a pred termínom ukončenia šprintu odovzdajú svoju prácu do centrálneho repozitára. Distribuovaný systém umožňuje oddeliť ukladanie zmien vo vyvýjanom softvéri od publikovania zmien pre ostatných programátorov, čo v centralizovaných systémoch nie je samozrejmé.

Náš tím pozostáva zo siedmych ľudí, no na projekte pracujeme súčasne na rôznych častiach. Na rovnakom zdrojovom kóde pracujú súčasne iba niekoľkí členovia tímu, takže ide vlastne o veľmi malý tím. Napriek tomu je používanie verziovacieho nástroja veľmi užitočné. Jednotlivým programátorom poskytuje výhody opísané v predchádzajúcich kapitolách. Ostatní členovia môžu sledovať jeho činnosť, čo je výhodné aj pre monitorovanie projektu.

Metóda vývoja Scrum tiež vyžaduje detailné sledovanie spĺňania jednotlivých cieľov. Tieto údaje sú používané na tvorbu tzv. burndown grafov, pomocou ktorých sledujeme priebeh vývoja. Verziovací systém je veľmi nápomocný aj v tomto smere pretože spĺňanie jednotlivých cieľov jednoducho sledujeme vďaka rozdeleniu na jednotlivé verzie.

Firemný tím

Centralizované systémy sú konceptuálne jednoduchšie a, podľa môjho názoru, aj jednoduchšie na používanie. Distribuované systémy však prinášajú viacero praktických výhod [1], z ktorých niektoré som už opísal, ale na ktoré sa znovu pozriem v kontexte firemných projektov.

Možnosť práce na projekte offline je jedna z hlavných výhod, ktoré poskytujú distribuované veziovacie systémy. V bežnej firme, kde sú všetci zamestnanci neustále prítomní v kancelárii však programátori pracujú na firemných počítačoch a sú neustále online, čo znamená, že môžu často nahrávať ich prácu na centrálny server. Offline na projekte zvyčajne nepracujú. Táto výhoda teda nie je vo firemnom kontexte zvyčajne aplikovateľná.

Ďalšou všeobecnou výhodou distribuovaných systémov je jednoduché vytvárenie a spájanie vetiev, čo podporuje vývoj experimentálnych funkcionalít, pretože ich je možné vyvýjať v samostatných vetvách, ktoré počas vývoja neovplyvňujú hlavnú funkcionalitu. Ak sa danú funkcionalitu podarí vyvynúť do dostatočne dobrého stavu, pridá sa k ostatným. Vo firemnom projekte je však málokedy potrebné vyvýjať experimentálne časti. Požiadavky na výsledný softvérový produkt sú tradične dopredu špecifikované zákazníkom. Aj pri použití agilnejších metód vývoja požiadavky na funkcionality špecifikuje zákazník a tieto požiadavky musia byť splnené.

Výhodou distribuovaných systémov je poskytnutie plného prístupu všetkým potencionálnym vývojárom. Toto môže byť veľmi užitočné pri iných typoch projektov, no pri firemnom, kde je jasne definované, kto má prístup k projektu a kto nie, je táto funkcionalita zbytočná.

Opísané výhody distribuovaných systémov sa podľa môjoho názoru ukazujú ako neveľmi užitočné v kontexte firemných projektov. Ako som spomínal, centralizované systémy sú podľa môjho názoru jednoduchšie na používanie. Na výber systému pre firmy by som použil parafrázu princípu Occamovej britvy: ak pre problém existujú dve rovnako dobré riešenia, použijeme to jednoduchšie.

Použitím centralizovaného systému bude navyše na serveri neustále najnovšia verzia, ktorú si potom môže jednoducho prezrieť vedúci tímu alebo, v prípade záujmu, aj samotný zákazník. Preto je podľa mňa centralizovaný verziovací systém jednoznačná voľba pre klasický firemný projekt.

Open source projekt

Na projekty s otvoreným zdrojovým kódom je ideálne použiť distribuovaný verziovací systém. V minulosti sa používali systémy centralizované, čo zo sebou prinášalo mnoho nevýhod.

Pri open source projektoch na rozdiel od ostatných nie je vždy jasné, kto má mať právo prispievať do projektu a kto nie. Práva na upravovanie kódu bolo potrebné udeľovať jednotlivým programátorom, ktorí sa zdali dostatočne schopní prispievať do projektu. Toto so sebou prinášalo potrebu testovať nových programátorov a trošku politiky, ktorá je pre projekt neužitočná a dokonca mu môže uškodiť.

Nie je tajomstvom, že Linus Torvalds vytvoril distribuovaný verziovací systém git pretože bol nespokojný s kvalitou predošlých systémov [5]. Jeho systém je teda priamo zameraný na open source projekty ako jeho vlastný Linux. Výhodou pre open source projekty je, že každý si môže skopírovať celý repozitár projektu a plnohodnotne na ňom pracovať. Hlavní vývojári potom skontrolujú zmeny, ktoré prispievatelia vykonali a v projekte použijú iba tie, ktoré považujú za potrebné a správne [1]. V tomto procese je veľmi dôležitá schopnosť systému vytvárať, spravovať a spájať vetvy vývoja, v čom, vďaka ich fundamentálnemu návrhu majú navrch distribuované systémy.

Na open source projektoch sa zvyčajne podieľa veľká skupina vývojárov, no z ich podstaty vyplíva, že počet programátorov môže byť aj veľmi malý. V každom prípade nevyhnutne potrebujú verziovací systém.

S prácou na open source projektoch priame skúsenosti nemám, no na projektoch, ktoré sa venujú open source sociálnemu programovaniu, ako napríklad GitHub vidím, ako návrh distribuovaného systému prospieva open source riešeniam.

Na tomto portáli sa nachádza aj množstvo menších projektov, na ktorých pracuje iba jeden programátor. Vďaka verziovaciemu systému potencionálni používatelia týchto projektov jednoducho a prehľadne vidia, ako často na nich autori pracujú a aké zmeny vykonávajú. Navyše ak je niekto nespokojný s vývojom projektu alebo jeho pôvodný autor na ňom nepracuje môže si jednoducho vytvoriť svoju kópiu projektu a pokračovať v práci podľa svojej chuti.

Záver

Verziovacie systémy sú pre mnohé tímy nevyhnutnosťou, no zídu sa aj tým, ktorí by sa bez nich v najhoršom zaobišli. Touto skupinou používateľov sú napríklad programátori-

jednotlivci alebo veľmi malé tímy, ktorým nebude manažovanie zdrojového kódu činiť také ťažkosti, ako v prípade veľkých projektov.

Na základe vlastných skúseností viem, ako dokáže používanie verziovacieho systému pomôcť pri mnohých problémoch spojených s vývojom alebo priamo pri vývoji softvérového produktu. Tieto výhody podľa môjho názoru robia používanie verziovacích systémov prospešné aj za cenu malého úsilia navyše.

Dôležité je aj samotné používanie verziovacieho systému a jeho výber. Na rôzne typy projektov je možné použiť rôzne systémy a ich výber je nie vždy jednoduchý. Preto je dôležité sa nad výberom dôkladne zamyslieť a zvážiť kontext, v ktorom bude tím na projekte pracovať. Hoci distribuované tímy sú modernejšie a majú viacero výhod, v prípade firemných projektov sa podľa môjho názoru viac hodí použiť systém centralizovaný.

Použitá literatúra

1. de Alwis, B., Sillito, J.: *Why are software projects moving from centralized to decentralized version control systems?* Cooperative and Human Aspects on Software Engineering, 2009. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5071408
2. Clifton, C., Kaczmarczyk L., Mrozek, M.: *Subverting the Fundamentals Sequence: Using Version Control to Enhance Course Management.* Proceedings of the 38th SIGCSE technical symposium on Computer science education. 2007. http://doi.acm.org/10.1145/1227504.1227344
3. Jotov, V.: *An investigation on the approaches for version control systems.* Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing - CompSysTech '08, 2008. http://portal.acm.org/citation.cfm?doid=1500879.1500959
4. Kagan, J.: *Version Control.* http://jonahkagan.me/projects/writing/version-control.html
5. Torvalds, L.: Linus Torvalds on git. Transcript from Google Tech Talk, May 2007. http://git.or.cz/gitwiki/LinusTalk200705Transcript

Annotation

Small-time Versioning

Version control systems are in practice widely used to manage the source code in teams with many programmers. Using these systems in small or even one member teams is not so certain, because a single programmer does not need complex source code management. However, using these systems brings a number of advantages which help the programmer to create a better software product. There are many version control systems in use. They are usually divided into two quite different groups – centralized and distributed. Choosing the right system can be a difficult task which differs from one project to the other. In this work I describe my opinion on choosing the right type of system and usage of the system by single programmers and teams with a small number of members.