Monitorovanie projektu na základe dát zo systému na správu verzií

MARTIN ZAGORA

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
martin[.]zagora[zavináč]gratex[.]com

Abstrakt. S rastom rozsahov softvérových projektov rastú aj nároky na plánovanie. Kvalitné plánovanie a riadenie projektu si zase vyžaduje flexibilitu a prispôsobovanie sa aktuálnej situácii. Toto nie je možné bez aktívneho monitorovania projektu a sledovania vývoja. Existujú viaceré spôsoby ako sledovať projekt a vyhodnocovať získané poznatky, ktoré sa potom uplatňujú pri plánovaní ďalšieho postupu. Jedným z nich je monitorovanie takzvaného systému na správu verzií (z angl. Version control system, ďalej VCS). Systém sa stará o skladovanie a správu verzií zdrojových kódov projektu. Skúmaním týchto údajov systém dokáže poskytnúť veľké množstvo informácií ako napr. mieru prispievania programátora k realizácii projektu alebo vzťahy medzi projektmi, prípadne ich jednotlivými komponentmi. Manažéri na základe týchto údajov môžu korigovať smerovanie projektu alebo prehodnocovať zloženie jednotlivých tímov pracujúcich na projekte. Tento dokument popisuje, čo všetko sa dá z VCS zistiť a uvažuje nad možnosťami využitia týchto informácií.

Úvod

Proces vývoja softvéru ponúka mnoho otázok, ako napr. či sa projekt uberá správnym smerom, či bude dokončený v stanovenom termíne, či bude spĺňať požadovanú kvalitu a mnohé ďalšie.

Keďže na tieto otázky nie je možné pri väčšom projekte takmer nikdy jednoznačne odpovedať, vzniká potreba monitorovania projektu. Vhodne realizované monitorovanie nám umožní robiť rozhodnutia, ktoré nasmerujú realizáciu projektu lepším smerom.

Je mnoho spôsobov ako monitorovať projekt. V tomto príspevku sa budeme zaoberať monitorovaním na základe reálnych dát, ktoré je možno získať zo systému na správu verzií (z angl. Version control system, ďalej VCS). VCS je systém, ktorý umožňuje uloženie a spravovanie verzií súborov v databáze, typicky (ale nie výlučne) zdrojových kódov programu. Medzi najznámejšie patria napr. CVS, Subversion (SVN), Team Foundation Server, Visual Source Safe.

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-7.

Z monitorovania VCS môžeme získať informácie rozličného charakteru. Môžeme merať užitočnosť jednotlivých programátorov na projekte, zisťovať, ako ľudia spolupracujú a lepšie identifikovať pre nich vhodné roly vo vývoji daného projektu. Tiež môžeme získať informácie o vývoji projektu, využiteľné programátormi aj manažérmi pri plánovaní ďalšieho kroku v projekte. Môžeme teda získať veľké množstvo informácií. Niektoré z nich si popíšeme bližšie s ich pozitívami aj negatívami.

Problém kvalitného výskumu

Oblasť získavania informácií z VCS a výskum ich spracovania sa stretáva s dvomi typickými prekážkami [2]:

- *Obmedzený prístup k reálnym a dostatočne veľkým VCS.* Spoločnosti vyvíjajúce softvér nie sú vo väčšine prípadov ochotné povoliť vedcom prístup k ich detailným záznamom o projekte a zdrojovým kódom, ktoré sú súčasťou ich komerčného softvéru. V akademickej oblasti sa často nachádzajú len projekty s menším počtom programátorov, s neporovnateľne kratšou dobou životnosti a s nie príliš bohatou históriou, hlavne oproti dlhodobým komerčným projektom.
- *Komplexnosť spracovávania veľkých VCS automatizovaným spôsobom sťažuje osvojenie a integráciu výsledkov v inom type výskumu*. V mnohých prípadoch je získavaná len cieľová oblasť informácií a nie je záujem vykonávať hĺbkové analýzy všetkých dostupných údajov.

Prvý z týchto problémov bol čiastočne odstránený mohutným rozmachom projektov s otvoreným zdrojovým kódom, pretože niektoré z nich poskytujú bohaté VCS porovnateľné s komerčnými projektmi.

Nutnosť kvalitného vedenia

Predtým než sa rozhodneme zaviesť monitorovanie, musíme vedieť, čo vlastne od výsledkov monitorovania očakávame a čo si myslíme, že nám tieto výsledky prinesú. Monitorovanie nie je jednoduchý proces a neexistuje ideálne riešenie vhodné pre každý projekt. Určite nie je vhodné monitorovať všetko, čo monitorovať dokážeme, ale treba zvážiť, ktoré výsledky budú užitočné pre ďalšie rozhodovanie a plánovanie. Výsledky monitorovania totiž nie je možné odložiť a vrátiť sa k nim o mesiac, lebo v tej dobe už pravdepodobne nebudú postačujúco aktuálne. Získavanie údajov, ktoré nikdy nepoužijeme, je mrhanie časom a samozrejme aj peniazmi.

Odhliadnuc od náhodných faktorov, ani monitorovanie, ktoré poskytuje správne výsledky neznamená automaticky správne rozhodnutia. Každý manažér, ktorý sa rozhodne monitorovať svoj projekt spomenutými spôsobmi, musí vedieť aj správne vyhodnotiť výsledky monitorovania. Približne rovnaké výsledky monitorovania nemusia pri dvoch rôznych projektoch automaticky znamenať rovnaké správne

rozhodnutie. Ďalším problémom môže byť konflikt výsledkov rôznych monitorovaní jedného projektu. Ak by sme mali k dispozícii vždy len jedno monitorovanie, rozhodnutie by bolo pomerne jednoduché (nie zaručene správne). Ak máme k dispozícii viac pohľadov na jeden projekt, rozhodnutia vyvodené z jednotlivých monitorovaní môžu byť v rozpore a len kvalitný manažér dokáže posúdiť, ktoré výsledky sú relevantnejšie a ktoré zanedbateľné.

Vzhľadom na spomenuté okolnosti *nie je možné* ani pri dobre realizovanom a prepracovanom monitorovaní *opomenúť ľudské vedenie* a rozhodovať sa len na základe výsledkov monitorovania.

Znižovanie rizika a nákladov

Každý projekt znamená pre manažéra potenciálny zisk, ale tiež prináša aj riziká. Najtypickejším cieľom je ukončenie projektu v stanovenom termíne a spĺňať pritom požiadavky zákazníka (prejsť akceptačnými testami). Postihy za nedodržanie takéhoto termínu môžu byť rôzne. V niektorých prípadoch to môže znamenať, že zákazník produkt odmietne [3].

Na základe získaných dát zo softvérového úložiska je možné veľmi jednoducho identifikovať miesta v projekte, ktoré prinášajú potenciálne problémy:

- *Veľké zhluky zdrojového kódu* - súbory vyznačujúce sa nadpriemernou veľkosťou oproti priemeru v projekte. Pravdepodobne ide o nevhodné spojenie viacerých prvkov v jednom veľkom súbore, čo zvyšuje nároky na údržbu a zároveň zvyšuje riziko chyby v kóde. Riešením býva rozdelenie súboru do viacero menších logických zoskupení kódu.
- *Dlho upravované kódy* - súbory, ktorých priemerný čas medzi vyžiadaním (z angl. check-out[1]) a následným odovzdaním (z angl. check-in[2]) je príliš veľký, naznačujú, že modifikácie kódu trvajú programátorom príliš dlho. V tomto prípade je tiež vhodné, aby manažér analyzoval danú časť programu. (Táto metóda, založená na čase je použiteľná len za predpokladu, že programátori nemajú súbory vyžiadané vtedy, keď na nich nepracujú, čo sa v praxi často nedodržiava, a preto je výsledok tejto metódy ťažko použiteľný)
- *Často meniace sa kódy* - súbory, ktorých frekvencia zmien je príliš vysoká. Môže sa jednať o zle navrhnutú časť programu alebo o veľmi chybovú časť programu. V oboch prípadoch je namieste vykonať podrobnú analýzu kódu s následným prepísaním problémovej časti. Príklad grafického znázornenia často upravovaných súborov je na obr. 1.

[1] označenie súboru na editáciu, ostatný používatelia nemôžu meniť súbor
[2] opätovné sprístupnenie súboru

Obr. 1. Graficky znázornená intenzita upravovania súborov v projekte

Všetky tieto problémy si vyžadujú prepis menšej či väčšej časti kódu. Je na manažérovi aby sa rozhodol, či prepísanie kódu a s tým súvisiaca investícia času sa vráti späť v podobe menej problémového kódu, alebo by prepisovanie bolo príliš nákladné a nevyplatilo by sa, napr. keď sa projekt blíži ku koncu.

Meranie užitočnosti programátorov

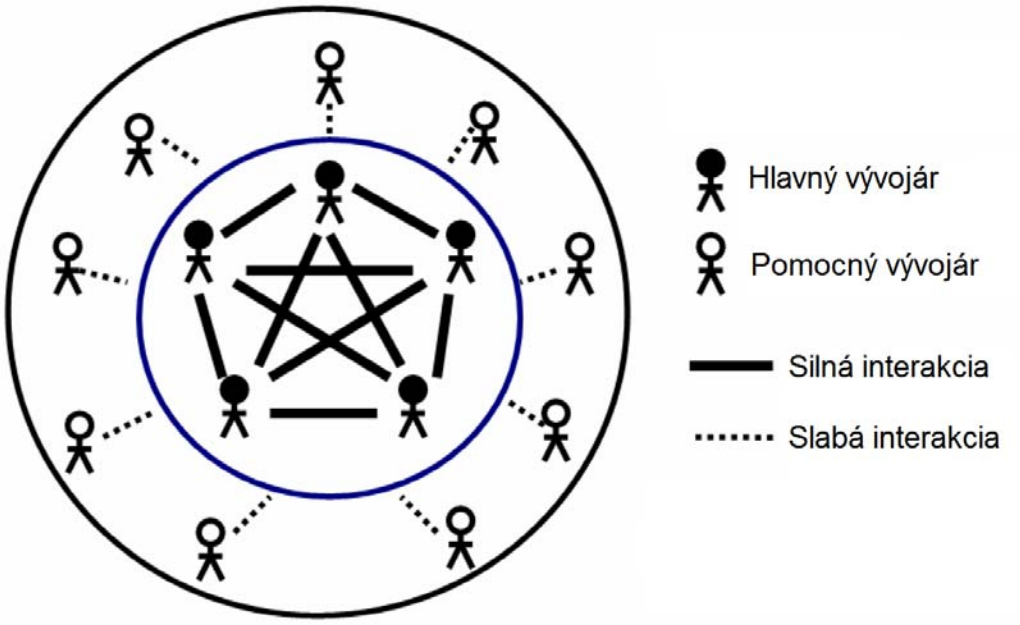
Každého projektového manažéra zaujíma, alebo by aspoň mala zaujímať efektivita a produktivita jeho podriadených, nakoľko sa od ich výkonov odvíja aj úspech či neúspech daného manažéra. Bežné metódy sledujú napr. počet riadkov napísaných daným programátorom. Tento údaj však môže byť veľmi zavádzajúci v prípadoch, keď lepší programátori dosiahnu rovnaký cieľ polovičným počtom kódu oproti ich kolegom.

Neexistuje žiadna jasná špecifikácia, ktorá by popisovala, čo je a čo nie je užitočný príspevok programátora pri vývoji softvéru. Programátori sa zaoberajú nielen písaním zdrojového kódu, ale aj tímovou komunikáciou, diskusiami s cieľom vyriešiť určité úskalia projektu. V neposlednom rade tiež opravujú chyby (angl. bugy), či už vlastné alebo ich kolegov. Z toho vyplýva, že samotné VCS nikdy nemôže byť dokonalým zdrojom informácií o zásluhách jednotlivých členov tímu. Keďže sa dá predpokladať, že každodenná komunikácia programátorov nie je žiadnym spôsobom zaznamenávaná, môžeme si dovoliť tvrdiť, že takto získané údaje môžu byť

informatívne, ale nemali by byť rozhodujúce pri hodnotení. Iný prípad je pri hodnotení programátorov pracujúcich na "diaľku", teda napr. vývoj open-source projektu po internete, pomocou mailing-listov a iných monitorovateľných komunikačných nástrojov. Tu je možné užitočnosť presnejšie hodnotiť kombinovaním informácií zo všetkých dostupných zdrojov [1].

Zisťovanie konzistencie tímu

Systémy VCS zaznamenávajú nielen každú zmenu zdrojového kódu, ale samozrejme aj autora tejto zmeny. Analýza a vizualizácia týchto zmien umožňuje odhaliť, na ktorých častiach projektu pracujú ľudia osamote a na ktorých pracuje pravidelne väčší počet programátorov. Obr. 2 znázorňuje grafickú reprezentáciu programátorov pridelených na projekt a ich vzájomnú interakciu v rámci určitej časti projektu.



Obr. 2. Organizácia vývojárskeho tímu [5]

Ani jeden z týchto prípadov neoznačuje zlý, ale ani dobrý prístup. Na určenie vhodnosti prístupu k jednotlivým častiam projektu je nutné tieto časti analyzovať, určiť ich náročnosť a odhadnúť správny prístup k realizácii. Ak sa napr. na projekte realizuje nejaká špecifická činnosť, povedzme príprava dokumentov na tlač, nie je vhodné, aby sa všetci programátori naučili orientovať v danej problematike, ale je lepšie poveriť touto úlohou jedného človeka. Tým sa zníži čas potrebný na učenie a programátori ho môžu využiť efektívnejšie vzhľadom na prebiehajúci projekt.

Ak na nejakom súbore pracovalo viac ľudí, dá sa predpokladať, že každý z nich nejako prispel k cieľovej funkčnosti danej časti programu. Pomocou analýzy informácií z VCS dokážeme pre každý súbor graficky znázorniť zastúpenia kódov od rôznych ľudí, ako to je napr. na obr. 3. Jednotlivé políčka znázorňujú zdrojové súbory projektu a intenzita farby znázorňuje, ako často autor modifikoval daný súbor v porovnaní so zvyškom programátorov.

Obr. 3. Príklad grafického znázornenia autorstva v projekte [4]

Extrakcia mapy projektu

Pomocou analýzy zdrojových súborov je možno veľmi jednoducho vytvoriť mapu projektu, teda graf znázorňujúci prepojenie jednotlivých súborov alebo komponentov v projekte. Vizualizácia tejto mapy nám ukáže, ktoré miesta sú izolované a ktoré miesta sú naopak odkazované vo veľkom počte súborov. Pri zmene takéhoto miesta treba venovať zvýšenú pozornosť dopadu na ostatné odkazované súbory, aby sme neznefunkčnili určitú časť projektu. Používanie takejto mapy skracuje čas nutný na hľadanie odkazov v súboroch a zároveň znižuje pravdepodobnosť vytvorenia chyby (bugu) v projekte.

Doplňujúce zdroje

Doteraz sme spomínali, aké informácie nám dokáže poskytnúť VCS. Aby boli tieto informácie ešte lepšie a konkrétnejšie, je vhodné ich dopĺňať ďalšími informáciami získanými z iných zdrojov ako napr. [4]:

- *Archívy e-mailov*. Väčšina projektov používa mailing-list, na ktorom programátori koordinujú svoju prácu a diskutujú nápady a zmeny v projekte. V týchto archívoch je možné nájsť veľké množstvo informácií o práci programátorov.
- *Softvér pre manažment tímu* (GanttProject, Open Workbench, NetOffice, dotProject, Microsoft Project, ...). Obsahuje evidenciu úloh, chýb (bugov) a zmien v projektoch, pričom tieto väčšinou obsahujú popis v komentároch. Z týchto komentárov môžeme tiež získať veľmi užitočné informácie.

Záver

V tomto článku som sa snažil o priblíženie monitorovania VCS systémov bežnému, v tejto problematike nezainteresovanému čitateľovi. Nie sú tu rozoberané konkrétne postupy a technológie, ale skôr všeobecný popis toho, čo sa môžeme z VCS dozvedieť, ako je to možné použiť v praxi a tiež aj môj osobný názor, prečo to vždy nemusí byť až také jednoznačné. Osobne si myslím, že monitorovanie je veľmi dobrá vec, ale oveľa dôležitejšie ako monitorovanie je osoba, ktorá výsledky monitorovania vyhodnocuje, teda kvalitný projektový manažér. Bez kvalitného vedenia ani najlepšie monitorovanie projekt nezachráni.

Použitá literatúra

1. Gousios, G., Kalliamvakou, E., Spinellis, D.: Measuring Developer Contribution from Software Repository Data. *Int. Proceedings of the 2008 international working conference on Mining software repositories*, May 2008, 129-132.
2. Hassan, A. E.: Mining Software Repositories to Assist Developers and Support Managers. *Int. 22nd IEEE International Conference on Software Maintenance (ICSM'06),* 2006.
3. Masticola, S.: Lightweight Risk Mitigation for Software Development Projects Using Repository Mining. *Int. Fourth International Workshop on Mining Software Repositories,* 2007.
4. Weißgerber, P., Pohl, M., Burch, M.: Visual Data Mining in Software Archives To Detect How Developers Work Together *Int. Fourth International Workshop on Mining Software Repositories,* 2007.
5. Yu, L., Ramaswamy, S.: Mining CVS Repositories to Understand Open-Source Project Developer Roles *Int. Fourth International Workshop on Mining Software Repositories*, 2007

Annotation

Project monitoring based on data from software repositories

With growth of software projects, requirements for planning are also growing. Good planning and project managing requires flexibility and adapting to current situation. This is not possible without active project monitoring and development tracking. There are many approaches to project monitoring and evaluating of results, which are later used in planning of next project step. One of them is monitoring data in Version Control System (VCS). It is commonly used for warehousing and managing versions of project source codes. By examining of this results, system can give us lot of informations like degree of developer's contribution to the project or relationships between projects or their components. Managers can make decisions based on these results and change course of project before it is too late. This document discusses what type of information we can get from CVS, how we can use it and also when it is appropriate to use them.