ČO POVIE ÚLOŽISKO VERZIÍ PROJEKTOVÉMU MANAŽÉROVI?

Kiež by aj ľudia mali úložisko, z ktorého by som sa dozvedel, čo sú zač.

Ján Kvak

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

Abstrakt. *Pri dnešnej úrovni informačných technológií a technológií všeobecne sa zodpovedajúcim spôsobom stupňujú aj nároky na softvérový produkt. Dnes už je potrebné, aby na softvérových projektoch pracovali tímy ľudí, ktoré sú kvalitne riadené a koordinované. Kvalitné riadenie môže byť kľúčovým faktorom k úspešnej realizácii softvérového projektu a v konečnom odovzdaní kvalitného produktu zákazníkovi. Riadenie akéhokoľvek procesu bez toho, aby sme poznali, v akom stave sa nachádza a bez toho, aby sme mali tento stav možnosť objektívne posúdiť, nie je možné. Preto je dôležitou súčasťou riadenia použitie správnych metrík, ktoré nám dokážu vyjadriť stav projektu. Dôležitým zdrojom informácií pre tieto metriky môžu byť systémy na kontrolu verzií. Cieľom tejto eseje je ukázať, ktoré informácie môže projektový manažér získať zo systému na kontrolu verzií a aké metriky môže na ne aplikovať.*

Kľúčové slová: *systém na kontrolu verzií, úložisko, metriky*

Úvod

Pri tvorbe softvéru už dnes nestačí len zistiť požiadavky zákazníka a vytvoriť podľa nich daný softvérový produkt. Ako sa vyvíjajú technológie, míľovými krokmi pokračuje aj vývoj v oblasti softvéru. Zároveň sa stupňujú aj nároky zákazníkov na softvérový produkt. Požiadavky sú stále náročnejšie a komplexnejšie. A pretože úlohou softvérového

inžinierstva je uspokojiť tieto požiadavky, tak sa musí vyvíjať aj proces tvorby softvéru. Softvérové produkty sú čím ďalej, tým rozsahovo väčšie a komplexnejšie. Čím je softvérový produkt rozsahovo väčší, tým viac ľudí je treba na jeho úspešnú realizáciu. Pri väčších tímoch ľudí však začína byť problémom ich koordinácia a využitie ľudského potenciálu. Je nutné zabezpečiť, aby sa práca na softvérovom projekte v rôznych fázach životného cyklu dostávala k tým členom tímu, ktorí sú schopní ju vykonať najefektívnejšie. Zároveň sa musia jednotlivé činnosti koordinovať, aby bol projekt v každej fáze životného cyklu vnútorne konzistentný.

Na vykonanie všetkých potrebných činností v procese tvorby softvérového produktu už teda nestačia len odborníci z oblasti programovania. Ich činnosť by sa minula účinkom, keby bola vykonávaná nekoordinovane a nekontrolovane. Dôležitými súčasťami tohto procesu sa teda stávajú jeho plánovanie, riadenie a monitorovanie.

Plánovanie v projekte nemá význam, ak nedokážeme sledovať, či plán plníme. Efektívne riadiť proces tvorby softvérového produktu je nemožné, ak nedokážeme určiť, v akom stave sa nachádza. Nemôžeš kontrolovať to, čo nemôžeš zmerať [3]. Preto je nutné proces monitorovať a podľa toho riadiť. Ako však proces tvorby softvéru monitorovať? Ako môžeme zistiť, ako postupuje, koľko práce už vývojári spravili a koľko ešte spraviť musia? Ako je možné dostať sa k potrebným informáciám? Je projektový manažér nútený prísť za každým členom tímu, počítať riadky jeho kódu alebo inak žiadať od neho dôkaz, že postupuje podľa plánu? Požadovať od každého vývojára, aby nahlasoval každú zmenu, každú prácu, ktorú na projekte vykoná? Bol by to veľmi náročný a projekt narúšajúci prístup. Pre vývojára by bolo veľmi nepríjemné, keď by bol neustále pod drobnohľadom projektového manažéra, keby každú zmenu, každý postup musel nahlasovať hneď, ako ho vykoná. Tým by sa jeho práca samozrejme spomalila a určite by klesla aj jej kvalita. Ani pre projektového manažéra by to nebol najvhodnejší spôsob práce. Má však aj iné možnosti ako získať informácie o stave projektu?

Softvérové prostriedky, ktoré napomáhajú pri samotnom vývoji softvéru môžu byť aj dôležitým zdrojom informácií o stave, v ktorom sa proces tvorby softvéru nachádza.
Prostriedky na manažment projektu sú nevyhnutné, ak na projekte pracuje väčší tím ľudí. Tí potrebujú svoju prácu zdieľať s ostatnými členmi tímu a zároveň pristupovať k ich práci. Celý tento proces musí byť vysoko koordinovaný, aby nedochádzalo ku konfliktom, nesúdržnostiam či vnútornej nekonzistencii softvérového produktu. Tieto problémy riešia systémy na kontrolu verzií.

Systém na kontrolu verzií(VCS)

Systémy na kontrolu verzií sa stali neoddeliteľnou súčasťou pri tvorbe softvérových projektov. Do veľkej miery tieto nástroje určujú, ako ľahko môžu ľudia prispieť do projektu, ako je koordinovaný vývoj novej súčasti, ako často sa spoja oddelené línie vývoja, ako je možné prezerať kód a ako je organizovaná podpora už hotového kódu [1]. Pre celý vývoj zložitého softvéroveho produktu počas celého jeho životného cyklu je dôležitý správny výber systému na kontrolu verzií. Správny výber môže ovplyvniť ako rýchlosť, s akou sme schopní softvérový produkt vytvoriť, tak aj jeho chybovosť, koordináciu práce v tíme a v konečnom dôsledku jeho cenu.

Vo všeobecnosti majú systémy na kontrolu verzií niekoľko spoločných znakov:

- Umožňujú tímu sledovať históriu súborov, na ktorých pracuje počas vývoja projektu. Členovia tímu môžu vidieť, kto vykonal zmenu, pochopiť, kedy a prečo ju vykonal, kontrolovať detaily zmeny a obnoviť stav projektu do času, keď bola zmena vykonaná.
- Členovia tímu môžu pracovať na subprojektoch bez toho, aby boli vyrušovaní zmenami svojich kolegov a bez toho, aby ovplyvňovali ich prácu. Tieto samostatné línie vývoja sa nazývajú vetvy. Používajú sa aj na manažment verzií, ktoré sa už aktívne nevyvíjajú.
- Keď sú práce na subprojektoch ukončené, tak sa integrujú späť do väčšieho projektu. Tento proces sa nazýva spájanie (*merging*) [2].

Systémy na kontrolu verzií teda okrem podpory práce v tíme, uľahčenia práce vývojárom plnia aj monitorovaciu funkciu. Manažér môže sledovať doterajší postup projektu, celú jeho históriu. Podľa zmien v systéme je možné kontrolovať metriky, ktoré o stave projektu vypovedajú. Systém na kontrolu verzií mu to umožní aj bez toho, aby narušil prácu vývojárov, bez toho, aby od nich vyžadoval správu o postupe. Získa však manažér týmto spôsobom všetky relevantné informácie na to, aby mohol posúdiť stav projektu? Aké sú vlastne informácie, ku ktorým má projektový manažér prístup cez systém na kontrolu verzií?

Závisí to aj od druhu systému na kontrolu verzií? Systémy na kontrolu verzií dnes môžeme rozdeliť do dvoch hlavných druhov, ktoré sú navzájom odlišné spôsobom uloženia spoločných dát a prístupu k nim. Z tohto pohľadu sa systémy delia na:

- centralizované - Centralized Version Control System (CVCS)
- decentralizované - Decectralized Version Control System (DVCS).

Centralizovaný systém na kontrolu verzií (CVCS)

Staršie systémy na podporu verzií používajú centralizovaný systém, kde sú dáta uložené v jednom úložisku. Je to samozrejme pokrok oproti predchádzajúcim prístupom, pretože umožňuje, aby na jednom veľkom projekte pracovalo väčšie množstvo vývojárov, ktorí nemusia nutne byť lokalizovaní na tom istom mieste. Systémy s týmto prístupom umožňujú prácu bez ohrozenia kvality kódu alebo dynamiky procesu tvorby softvéru. Tieto systémy fungujú na princípe klient – server. Srdcom celého systému je úložisko, ktoré je zodpovedné za uloženie všetkých verziovaných objektov. Verziovaný objekt je objekt (súbor zdrojového kódu, trieda, obrázok), ktorý je pod kontrolou verzií a každá jeho zmena je zaznamenaná pomocou VCS [7].

K spoločnému dátovému úložisku sa pristupuje cez „*checkout*"[1]. Je tým zabezpečené, aby viac vývojárov nemenilo v tom istom čase jeden súbor, čo by mohlo viesť k nežiaducim zmenám, nezaznamenaniu žiaducich zmien alebo k nefunkčnosti celého systému. Vetvy (*branches*) sú využívané systémami kontroly verzií na paralelné úpravy obsahu úložiska.

Práve úložisko je miestom, ku ktorému má prístup projektový manažér. Má tak prístup k hlavnej vetve (*mainline branch*) [5] systému, ktorá reprezentuje hlavný vývoj

systému. Môže sledovať zmeny, ktoré vykonali jednotliví vývojári a aplikovať na tieto zmeny jednotlivé metriky.

Decentralizovaný systém na kontrolu verzií (DVCS)

Decentralizované systémy na kontrolu verzií sú novším prístupom. Namiesto jedného centrálneho úložiska využívajú niekoľko prepojených úložísk, každé samostatné pre každého vývojára. Synchronizácia týchto úložísk je samostatným krokom.[7] Týmto prístupom sme sa zbavili problému prístupu k centrálnemu úložisku dát, keďže každý vývojár má svoje vlastné úložisko s plným prístupom k vyvíjanému softvéru, kompletným záznamom doteraz vykonaných zmien a právom meniť, pridávať či mazať akúkoľvek časť programu. Toto umožňuje vyvíjať softvérové produkty aj bez pripojenia na sieť, takže je oveľa flexibilnejší ako centralizovaný systém, ale hlavnou výhodou je oveľa väčšia rýchlosť, keďže väčšina operácií sa vykonáva na strane klienta. Aj synchronizácia úložísk je rýchlejšia, pretože obidva stroje obsahujú metadáta o vykonaných zmenách. Typickými predstaviteľmi tohto typu systémov sú GIT, Mercurial, BZR a BitKeeper.

Okrem toho, že tento typ systému rieši problémy, ktoré riešil aj centralizovaný systém, umožňuje navyše pracovať všetkým členom tímu na projekte naraz aj bez nutnosti byť pripojený na centrálny server. Ďalšou výhodou je, že je rýchlejší a efektívnejší.

Tu môže projektový manažér naraziť však na problém, pretože tento prístup je síce pre vývojárov príjemnejší, ale môže spôsobiť, že manažér nemá v danom momente prístup ku všetkým relevantným dátam, pretože ich má vývojár vo svojej lokálnej kópii úložiska. Tento problém však vo väčšom časovom horizonte rieši práve spájanie oddelených úložísk jednotlivých vývojárov (*merging*). Vtedy je možné znovu identifikovať príspevky vývojárov ku konečnej verzii produktu.

Metriky merateľné v systéme na kontrolu verzií

Zistiť informácie o stave projektu nie je triviálna záležitosť. Ako sledovať objekt, ak nevieme odsledovať a vyjadriť, v akom stave sa objekt nachádza? Počas celej histórie ľudia posudzovali veci podľa vlastností, ktoré sa dali objektívne porovnávať a vyjadriť. Najjednoduchšie je posudzovať vlastnosť, ktorá sa dá vyjadriť číselne. Nie vždy sa vlastnosti objektu však dajú priamo vyjadriť nejakou metrikou. Vtedy prichádzajú do úvahy aj metriky, ktoré stav opisujú nepriamo. Nie vždy vyjadrujú skutočný stav objektu alebo procesu, ale môžu byť vodítkom, ako sa k opisu skutočného stavu priblížiť.

Podľa typu, by sa dali entity a atribúty, ktoré chceme merať pomocou metriky rozdeliť do troch skupín [6]:

1. **Procesy** – sú to v softvérovom inžinierstve v čase prebiehajúce aktivity, ktoré sú dôležité pre dosiahnutie daného cieľa, ktorým je kvalitný softvérový produkt. Charakterizované sú práve časovým vymedzením, či už explicitne daným, kedy proces musí skončiť v určitom čase, alebo implicitným, kedy ďalší proces začína vtedy, keď predchádzajúci skončí.[6] Tu sa hneď ako prvá metrika ponúka čas, ktorý daný proces trvá. Pomocou systémov na kontrolu verzií môžeme sledovať, ako dlho danému vývojárovi trvalo, kým skončil úlohu, ktorou bol poverený. Je

možné merať aj úsilie, ktoré bolo na dokončenie danej časti systému treba vynaložiť. Systém nám poskytne históriu, kde môžeme zistiť koľko človeko-hodín, človeko-dní či dokonca človeko-mesiacov sme vynaložili pri tvorbe danej časti produktu či v konečnom dôsledku pri tvorbe celého produktu. Toto môže byť informácia veľmi dôležitá pre riadenie a plánovanie.

2. **Produkty** – nie sú to len konečné produkty, ktoré doručíme zákazníkovi, ale aj všetky súbory, objekty či dokumenty vytvorené počas životného cyklu projektu [6]. U produktov môžeme merať ich veľkosť, zložitosť, chybovosť a ďalšie vlastnosti. Ak chceme zistiť veľkosť softvérového produktu, jednou z najpoužívanejších metrík je LOC (*Lines of code*) – počet riadkov kódu. Na jednej strane nám dosť jasne dáva najavo, koľko daný vývojár pridal kódu, ale nemusí až tak pravdivo vypovedať o jeho skutočnom príspevku k vývoju softvérového produktu. Slabinou tohto typu metriky je, že sa dá bezpečne využiť až na konci celého projektu [4]. Mnohokrát je možné tú istú funkcionalitu dosiahnuť efektívnejšie použitím kratšieho kódu. Na odstránenie tejto nevýhody sa používa ďalší typ metriky FP (*Functional points*) – body funkcionality. Postup projektu sa touto metrikou určuje podľa toho, koľko funkčných bodov v programe pribudlo. Ak však programátor upravil funkcionalitu nejakého modulu tak, že napríklad implementoval vhodnejší algoritmus, nemuseli nutne pribudnúť ani riadky kódu alebo nové funkčné body. Programátor v našom prípade však celú jeho prácu zachytil a opísal v dokumentácii. Preto aj posun v dokumentácii indikuje posun celého projektu vpred. Aj vyhľadávanie a oprava chýb existujúcich vo vytváranom produkte je dôležitá činnosť, to znamená, že počet chýb môže byť dôležitou metrikou postupu projektu. Všetky tieto vyššie spomínané metriky sú použiteľné, ak vytvárame softvérový produkt pomocou systému na kontrolu verzií. Softvérový manažér má prístup ku kódu a súborom jednotlivých vývojárov a preto môže jednoduchým spôsobom zistiť, koľko pribudlo riadkov kódu a funkčných bodov, koľko chýb odbudlo, či ako pokračuje projekt podľa záznamov v dokumentácii.
3. **Zdroje** – sú to entity potrebné a dôležité pre proces vývoja softvéru. Zaraďujeme sem ľudské zdroje, materiál, nástroje a vybavenie aj metódy práce [6]. Niektoré charakteristiky ľudských zdrojov však sú pomocou dát, dostupných zo systému na kontrolu verzií, ťažko odpozorovateľné. Dá sa síce podľa toho, ako často vývojár pridáva príspevok k lokálnemu úložisku (pri centralizovaných systémoch na kontrolu verzií) alebo ako často spája svoje lokálne úložisko s centrálnym (pri distribuovaných systémoch) približne usudzovať, aká je jeho produktivita, ale nemusí to byť vždy objektívne posúdenie. Hlavne pri distribuovaných systémoch na kontrolu verzií, kde môže vývojár pracovať na svojej vlastnej kópii systému bez pripojenia k sieti, nemusí byť frekvencia spájania bernou mincou. Až v súčinnosti s inými metrikami môže čo-to vypovedať o produktivite daného zamestnanca. Ak si zoberieme ako príklad zamestnanca, ktorý dlhší čas (povedzme týždeň) nespojil svoje lokálne úložisko s hlavnou vetvou programu, nemusí to znamenať, že produktívne nepracoval. V prípade, že pridal potom svoj príspevok, ktorý bol naozaj kvalitný a ošetrený od chýb, tak má jeho práca väčšiu hodnotu ako práca zamestnanca, ktorý síce často svoje príspevky pridával, ale

chybovosť bola u neho oveľa vyššia. Samozrejme, že ak niektorý vývojár neprimerane dlhý čas neprispel k hlavnému programu, tak sa dá usudzovať, že má nejaký problém. Na to by mal kvalitný projektový manažér pružne reagovať a problém riešiť. Charakteristiky systémov, na ktorých boli dané časti systému vyvíjané, nie sú však zo systému na kontrolu verzií poznať. Toto sú charakteristiky, ktoré musí projektový manažér získavať inými prostriedkami.

Záver

Ktoré metriky teda môžeme sledovať, ak máme ako projektoví manažéri prístup k úložisku systému na kontrolu verzií? Systémy na kontrolu verzií sú softvérové podporné prostriedky pre vývoj softvérových produktov. Umožňujú tímom vývojárov vyvíjať väčší produkt spoločne tak, aby nevznikali vnútorné nesúdržnosti a chyby v projekte. Okrem toho distribuované systémy na kontrolu verzií umožňujú pracovať vývojárovi na lokálnej kópii celej doterajšej histórie softvérového produktu aj bez nutnosti prístupu na sieť. Vývojárom teda poskytujú priaznivé prostredie pre vývoj a projektovým manažérom poskytujú cenný zdroj informácií o stave projektu. Systém na kontrolu verzií poskytuje manažérovi prístup k reálnemu postupu projektu. Získava prístup k samotnému kódu, informáciám typu kto, kedy a aký kód pridal. Použitím správnych metrík môže z týchto informácií zistiť, ako postupuje samotný kód projektu, či postupuje podľa plánu. Ďalej má možnosť zistiť, aké sú príspevky jednotlivých vývojárov, aká je produktivita ich práce, aký kvalitný kód poskytujú, koľko je v ňom chýb a aká je jeho zložitosť. Vďaka týmto informáciám sa môže ďalej fundovane rozhodnúť, ako postupovať ďalej v plánovaní projektu, ako postupovať v riadení, môže včas odhaliť chyby, či problémy, ktoré nastali. Tým môže predísť problémom, ktoré pri väčších softvérových projektoch nastávajú a včas odovzdať produkt zodpovedajúcej kvality zákazníkovi.

Použitá literatúra

1. Brian de Alwis, Jonathan Sillito: Why Are Software Projects Moving From Centralized to Decentralized Version Control Systems? Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering ,(2009), 36-39
2. Brian O Sullivan Making Sense of Revision-control Systems, ACM Queue, Volume 7 , Issue 7 (August 2009)
3. DeMarco, Tom. Controlling Software Projects: Management, Measurement and Estimation. ISBN 0-13-171711-1.
4. Gousios G., Kalliamvakou E., Spinellis D.: Measuring Developer Contribution from Software Repository Data. Proceedings of the 2008 international working conference on Mining software repositories,129-132
5. L. Wingerd and C. Seiwald. High-level SCM best practices. In System Configuration Management, volume 1439 of LNCS, pages 57–66. Springer, 1998.
6. Scotto M., Silitti A., Succi G.,Vernazza T.: A relational approach to software metrics. In:ACM Symposium on Applied Computing (2004),1536-1540

7. Vladimir Jotov: An investigation on the approaches for version control systems, International Conference on Computer Systems and Technologies - CompSysTech.08, (2008)

Annotation

What is repository telling to a project manager?

Nowadays there is the development of information technology, technology generally and it demands for software product become more important. It is necessary to manage and coordinate project software teamwork. To be succesful we need good management and we can provide an excellent product to a customer. We need to know process, its state and objective judgement to be able to manage it. The important part of the management is using the right metrics, which can express the state of the project. The objective of the essay is to indicate which information can be won by the project manager from version control system and metrics, which can be applied.