MIERA VPLYVU MONITORINGU NA SOFTVÉROVÝ PROJEKT

Nemôžeš kontrolovať to čo nemôžeš merať.

Marek Tuška

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

Abstrakt. Pre kontrolu dodržiavania plánu pri tvorbe softvérového projektu je veľmi dôležité jeho monitorovanie. Na sledovanie stavu projektu nám slúžia metriky, s rôznou mierou komplexnosti, spoľahlivosti ako aj výpovednej hodnoty. Výber vhodných metrík a citlivý prístup k ich vyhodnocovaniu môže mať kľúčový dopad na splnenie *časového harmonogramu a vyhnutiu sa rôznym iným problémom. Táto esej sa zaoberá problematikou do akej miery sa môžeme spoliehať na metriky a vyvodzovať z nich dôsledky a súčasne skúma do akej miery nám môžu tieto metriky odhaliť pravý dôvod vzniku problému. Esej sa snaží pomocou analýzy a porovnávania rôznych techník monitorovania softvérového projektu poukázať na dôležitosť ľudského faktoru pri výbere vhodných metrík a ich správnom vyhodnocovaní.*

Kľúčové slová: *metriky, projekt, monitorovanie, ľudský prístup*

Úvod

V procese vývoja softvérového projektu je správne naplánovanie vykonávania jednotlivých fáz jedným z kľúčových faktorov. Správne naplánovanie nám ale nezaručuje že sa nevyskytnú určité problémy, ktoré môžu narušiť aj dobre naplánovaný projekt. Na nečakané problémy, ako aj na problémy akéhokoľvek druhu, je treba reagovať čo možno najskôr, aby sa minimalizovala kolíznosť s ďalšími fázami vývoja a nevznikali ďalšie iné

komplikácie. Reakcia na zle sa vyvíjajúcu situáciu by nebol až taký problém, ale musíme vedieť čo konkrétne daný problém vyvoláva, teda potreba poznania konkrétnej príčiny problému. Potreba odhalenia správnej príčiny je nesporne veľmi dôležitý faktor pri riadení softvérového projektu, ale ďalším faktorom, a podľa mňa ešte dôležitejším, je včasné odhalenie tohto problému. Na monitorovanie projektu nám slúžia metriky sledujúce rôzne faktory vplývajúce na vývoja softvéru. Dôležité je ale rozlišovať tieto metriky a správne vyhodnocovať ich výstupné hodnoty. Netreba ale zabúdať ani na ľudský faktor pri sledovaní týchto metrík, pretože nemusia byť vždy zabezpečené správne vstupy pre tieto metriky. Preto si musíme byť vedomí aj týchto skutočností, ktoré nám v určitých prípadoch môžu pomôcť pri správnom vyhodnocovaní metrík.

Ako sa rozdeľujú softvérové merania

Pred uvedením konkrétnych metrík je dobré si uvedomiť čo sa dá na softvérovom projekte merať. Keďže softvérový projekt nemá fyzické atribúty, ako napríklad človek má výšku a váhu, nemôžeme používať na meranie konvenčné metódy. Preto boli zavedené skupiny meraní, nazývané metriky, ktoré vyjadrujú veci ako veľkosť, spoľahlivosť softvéru, komplexnosť ako aj iné aspekty softvérového produktu. Niektoré z týchto charakteristík môžu byť merané priamo, ale ďalšie musia byť merané nepriamo.
P.K.Pandey vo svojej práci [1] rozdeľuje tri oblasti záujmu softvérového merania

- Procesy
- Produkty
- Zdroje

Pod procesmi rozumieme softvérovo príbuzné aktivity, ktoré trvajú určitý čas. Produkty sú výtvory, ktoré sú výsledkom procesov. Zdroje sú položky, ktoré sú vstupmi pre procesy.

Základných rozdelenie atribútov softvérového projektu:

- Interné atribúty
- Externé atribúty

Interné atribúty sú tie ktoré sa týkajú priamo procesu, produktu alebo zdroju. Napríklad dĺžka je interným atribútom softvérového dokumentu a čas trvania je interným atribútom procesu. Externé atribúty procesov, produktov a zdrojov sú tie ktoré môžu byť merané iba po zohľadňovaním v akom sú vzťahu proces, produkt a zdroj k iným entitám prostredia. Napríklad spoľahlivosť programu nie je závislá iba na programe samotnom ale aj na kompilátory, počítači ako aj na používateľovi [1].

Nakoniec, samotné merania sú rozdelené na:

- Priame merania
- Nepriame merania

Priame merania sú tie, pri ktorých meranie určitého atribútu nezávisí od merania iného atribútu. Nepriame sú naopak tie, pri ktorých meranie určitého atribútu závisí od meraní iných atribútov.

Aké poznáme softvérové metriky

Ideálna metrika by mala byť jednoduchá a jasne definovaná, aby bolo jasné čo bude vyjadrovať, objektívna, aby zohľadňovala všetky aspekty rovnako, validná, teda aby merala to na čo je určená a zároveň robustná, teda relatívne nezávislá od nedôležitých zmien v procese alebo v produkte. Každá zo súčasne používaných metrík spĺňa tieto požiadavky vo väčšej či menšej miere. Práve preto treba dobre poznať výhody a nevýhody danej metriky, čo nám môže pomôcť pri výbere správnych metrík, ktoré sa budú hodiť na meranie konkrétneho softvérového projektu.

Medzi najznámejšie a najpoužívanejšie metriky patrí napríklad LOC (Line of code), teda počet riadkov kódu. Táto metrika je síce jednoduchá, jasne definovaná a validná, ale o objektivite sa už veľmi hovoriť nedá. Keďže sa jednoducho vyhodnocuje len počet riadkov kódu je zrejmé že pri porovnávaní výkonnosti práce zamestnanca sa mohlo stať že zamestnanec A je produktívnejší ako zamestnanec B. Zamestnanec A síce mohol napísať viac riadkov kódu ako zamestnanec B, ale program zamestnanca B mohol napríklad byť viac optimalizovaný a rýchlejší alebo naprogramovaný v inom programovacom jazyku. [2]

Ďalšia známa metrika sa volá Halstead. Berie do úvahy dĺžku programu spolu s náročnosťou napísania, ako aj pochopenia takéhoto programu, ale na druhú stranu závisí na kompletnom kóde. Pred úplným dokončením kódu teda nie je možné sledovať produktivitu, čo sa mi zdá ako veľká nevýhoda tejto metriky. [2]

McCabeho metóda meria počet nezávislých ciest v programe, čo v skutočnosti znamená počet testovaných podmienok v nachádzajúcich sa v programe. Táto metrika sa môže použiť aj počas tvorby softvéru, na rozdiel od Halsteada, ale napríklad vnoreným podmienkam je prikladaná rovnaká váha, napriek väčšieho nároku na pochopenie takýchto štruktúr. Súčasne s veľkým počtom jednoduchých podmienok a rozhodnutí stráca táto metóda na výpovednej hodnote. Nevýhoda tejto metriky by mohlo byť ak by sa vývojári dozvedeli že sa používa práve táto metrika, potom by pre nich nebol problém používať vždy jednoduchších podmienky namiesto zložitejších, aby zlepšili svoje hodnotenia. [2]

K najviac akceptovateľným metrikám v súčasnosti patrí napríklad metrika funkčných bodov. Cieľom tejto metriky je spočítať absolútnu hodnotu funkčných bodov, ktoré sú určené počtom používateľských vstupov, výstupov, dopytov a hlavných programových súborov. Metriky funkčných bodov sú nezávislé na programovacom jazyku, takže pri porovnávaní dvoch programov, ktoré sú naprogramované v iných programovacích jazykoch, ale poskytujú rovnakú funkcionalitu, je vyhodnotený ako lepšií vyšší programovací jazyk. [3]

Metrika bodov prípadov použitia (use case points) je podobne ako metóda funkčných bodov nezávislá na použitom programovacom jazyku. V tejto metrike sa prikladajú rôzne váhy prípadom použitia ako aj hráčom, aby sa určil ich vplyv na nefunkcionálne faktory a faktory prostredia [3]. Metriky prípadov použitia sú určené na ohodnocovanie projektov vyvinutých objektovo orientovanými metódami. [3]

Aké môžu nastať ďalšie problémy

Po naštudovaní si týchto metód môžeme nadobudnúť pocit, že nám stačí si naštudovať jednotlivé výhody a nevýhody daných metrík a následne sa môžeme na ne spoľahnúť. Ale je to naozaj tak? Môžeme si síce byť vedomí, že metrika LOC nám nezohľadňuje programovanie v rôznych programovacích jazykoch, že Halstead nemôžeme použiť bez kompletného kódu a že McCabeho metriky majú určité výhody, ale existujú situácie kedy nie je efektívna. Stačí ale táto vedomosť k tomu aby sme si mohli byť istý tým, že keď sme vybrali správne metódy a vieme o ich nedostatkoch, tak sa nič iné, neočakávané, nemôže stať?

Zoberme si len jednoduchý príklad obyčajného programátora. Programátor sa učí nový jazyk a dosahuje horšie výsledky ako keď programoval v starom programovacom jazyku, v ktorom mal bohaté skúsenosti overené rokmi praxe. Metrika tvrdí že programátor sa oproti minulému obdobiu zhoršil, môžeme ale povedať že tento programátor sa zhoršil len na základe vedomosti ktorú nám vyjadruje daná metrika? Podobne sa môže programátor snažiť použiť nové, modernejšie metódy a postupy pri ktorých samozrejme naráža na neočakávané problémy, ktorých riešením môže stráviť viac času ako keby použil staré, zaužívané postupy. Výsledky metrík potom môžu vyhodnotiť jeho výkon horšie ako sa očakávalo a znechutiť daného programátora v progrese a donútia ho k stagnácii, čiže jeho snaha o zlepšenie kvality bude potrestaná.

Podobným problémom môže byť nepochopenie významu metrík z pohľadu ľudí za to zodpovedných. Ak nebudú títo ľudia dostatočne vycvičení a oboznámení s významom jednotlivých metrík tak nebudú zberať relevantné údaje. Odhliadnuc od prípadu, že by takýto ľudia z výsledkov určitých metrík neboli schopní vyvodiť žiadne konkrétne dôvody problémov, či by sa dokonca vyvodili zlé dôvody, tak by im mohli uniknúť širšie súvislosti medzi rôznymi metrikami. Napríklad ak by určitá metrika vykazovala náznak problému, tak slabo vycvičený človek si nebude vedieť overiť či tam takýto problém naozaj je, alebo nie je, pomocou inej metriky. Takýmto spôsobom môže prísť k zisteniu nedostatkov až v neskoršej fáze projektu, kedy sú už hodnoty metrík pri pokračovaní problémov výraznejšie.

Nedostatočnú komunikáciu považujem tiež za jeden z možných problémov. Aplikovaním na prechádzajúci príklad, ak by metriky vykazovali náznak určitého problému a daný človek by z toho nevedel vyvodiť konkrétny dôvod problému, mohol by upozorniť kolegu, ktorý by si s týmto problémom mohol vedieť poradiť. Ak by si ale nebol istý dôvodom, ktorý spôsobil dané výsledky metrík, a s nikým by tento problém nekonzultoval, nemuselo by nastať včasné odhalenie problému.

Ďalším problémom by mohlo byť napríklad zlé zadefinovanie metrík. To by mohlo viesť k zlému ohodnoteniu určitého problému rôznymi ľuďmi. Napríklad odhalenie chyby pri testovaní by mohlo byť jedným človekom ohodnotené ako chyba pri testovaní, ďalším ako chyba pri programovaní. To samozrejme môže nastať len ak neexistujú metodiky určujúce postup v takýchto situáciách. Takto by sa k jednému objektu pristupovalo ako keby to boli dva rôzne objekty. Takáto zlá definícia by mohla robiť problémy pri použití

metrík určených na vyhodnocovanie projektov vyvinutých pomocou objektovo-orientovaných metód, ako sú napríklad metriky prípadov použitia. [1]

Vo väčšine organizácií je použitý určitý automatický nástroj na zbieranie a vyhodnocovanie metrík. Hlavne pri zložitejších metrikách, ako sú napríklad metriky funkčných bodov alebo metriky bodov prípadov použitia, je takýto nástroj nevyhnutný. Každý nástroj, respektíve každý výrobca, implementoval rovnaké metriky vo svojom nástroji trochu iným spôsobom. Preto sa môžu hodnoty ktoré vykazujú rovnaké druhy metrík v rôznych nástrojoch, ale vykonané nad rovnakým projektom, trochu líšiť. Treba si byť vedomí aj tejto skutočnosti a preskúmať ako ktorý nástroj vyhodnocuje rôzne metriky, hlavne pri používaní alebo prechode na iný nástroj. Automatické nástroje a techniky zberu dát sú výhodné, hlavne z hľadiska odbremenenia jednotlivých vývojárov od extra práce pri zaručovaní platnosti a integrity dát a pri pomáhaní prezentácie a zhodnocovania údajov z metrík. V [4] je opísaná skutočnosť že „nie vždy je možný zber presných dát bez aktívneho zapojenia vývojára“ . Z toho vypláva, že spoliehať sa len na automatické nástroje tiež nemôžeme.

Ďalšou témou môže byť nedôvera vývojárov k metrikám. Môže nastať jav ktorý v [5] nazvali ako „gaming“, teda vyhýbanie sa plnej zodpovednosti metrikám programu pri súčasnom tvárení sa že poskytujú všetky požadované dáta. Tento dôvod môže nastať zo strachu z porovnávania výsledkov metrík medzi rôznymi projektmi alebo organizačnými štruktúrami, o ktorých si vývojári myslia, že by sa nemali porovnávať. Tento dôsledok vyplýva z presvedčenia, že každý projekt je unikátny a preto nie je možné jeho porovnávanie s inými projektmi. Iným dôvodom gamingu môže byť strach, že sa namiesto monitorovania procesov a produktov sledujú práve vývojári a že metriky môžu byť použité proti nim.

Extrémnejšia forma gamingu je vedomé falšovanie metrík. Takéto falšovanie vstupných údajov pre metriky môže nastať pomocou využitia dier v metrikách, napríklad uploadovanie prázdnych dokumentov ak si metrika neoveruje či je odovzdaný relevantný dokument. Zisťovanie, či sú vstupné dáta pre metriky falšované, je veľmi ťažké na odhalenie. Odhalenie je možné po preverení určitých podozrivých údajov. Napríklad vývojový tým by mal mať úspešnosť 90% z plánovanej úspešnosti. Ale vývojári sú si vedomí tohto čísla a preto umelo zdvihnú priemernú úspešnosť na 80%. Keď ale v predchádzajúcich obdobiach dosahoval rovnaký tým úspešnosť 70%, tak je táto zmena podozrivým údajom. [5]

Možným problémom gamingu môže byť aj nedôvera k nemennosti a neférovosti metrík. Vývojári môžu byť presvedčení, že výsledky metrík sú aj tak menené manažérmi, aby vykazovali lepšie výsledky ako sú v skutočnosti. Z tohto dôvodu sa plne nezapájajú do aktivít súvisiacich s metrikami. Neférovosť môže byť vnímaná napríklad pri nájdení chyby vzhľadom na jeho opravu. Napísanie 2000 riadkov kódu môže byť napríklad rovnako hodnotené ako opravenie jednej chyby v tomto programe zmenou iba jedného riadku. [6]

Niektoré problémy nemusia byť reportované vývojármi z dôvodu kamarátstva medzi danými vývojármi. Takýto vývojári môžu napríklad reportovať len problémy s vysokou a strednou prioritou a niektoré problémy, ktoré vznikli ako výsledok práce spolupracovníka nie sú vôbec reportované. Takéto správanie môže nastať z dôvodu faktu, že počet chýb vo vlastnom kóde, ako aj v inej práci, je brané ľuďmi ako veľmi citlivý údaj.

Ďalším problémom súvisiacim so spolupracovníkmi môže byť potreba zachovania dobrého dojmu u spolupracovníkov. Nikto nie je rád ak na neho sú reportované problémy. Ak metriky vyhodnotia zlú úspešnosť jednotlivca, môžu byť tieto metriky spochybňované, aj keď reflektujú reálnu situáciu. Pre každého je úplne prirodzenou vecou, že svoju vykonanú prácu hodnotia ako dobrú, preto sú pre nich metriky hodnotiace ich prácu ako chybnú, považované za zavádzajúce. Ak príde k podobnému odhaleniu, tak ak nabudúce vývojár odhalí veľké množstvo chýb, môže ich nereportovať, ale bude sa snažiť ich vyriešiť a odovzdá až kód o ktorom si je stopercentne istý, že má potrebnú kvalitu. To ale môže nepriaznivo ovplyvniť dobu potrebnú na spustenie ďalších fáz softvérového projektu, ktoré sú naviazané na tú časť ktorú daný vývojár vylepšuje.

Z predchádzajúceho problému môže vyplávať podobný problém. Keď napríklad poviete vývojárovi že v týchto 4 veciach má veľký počet chýb, tak tento vývojár môže zamerať všetku svoju pozornosť na tieto 4 oblasti problémov, aj napriek tomu že tieto oblasti nie sú také dôležité, ako tie na ktorých by mal pracovať. Samozrejme, je ťažké presvedčiť takéhoto vývojára, že práca v oblastiach, ktoré majú väčší strategický význam z pohľadu celkového projektu, je dôležitejšia ako zlepšenie si vlastných čísel. [5]

Ako by sa dal zlepšiť prístup k monitorovaniu

Základom zlepšenia prístupu ľudí k monitorovaniu je správne pochopenie dôvodov monitoringu, ako aj z pohľadu manažérov tak aj z pohľadu ľudí, ktorých práca je monitorovaná. Toto ale stále nezaručuje, že sa na metriky bude pozerať ako na nástroj, ktorý nám pomáha, namiesto ako na niečo čo môže byť použité proti nám. Dôležité je preto presvedčiť ľudí, že metriky nie sú namierené proti nim, ale že metriky sú tu preto aby sledovali procesy a produkty. Ďalej ich treba presvedčiť transparentnosti meraní, teda že metriky vyjadrujú skutočnosť, teda zabrániť v demotivácii z presvedčenia o manipulácii meraní manažérmi aby vyzerali dané čísla lepšie ako v skutočnosti sú.

Odstrániť predsudky z hodnotenia podľa čísiel nebude podľa mňa nikdy možné. Hlavným dôvodom je preto hlavne fakt že každý človek je iný, a hodnotenie strojom, teda pomocou nejakých výstupných čísel, nebude pokrývať všetky aspekty, ktorými sa konkrétny vývojár zaoberal. Práve preto by som radil manažérom, respektíve ľudom zodpovedným za sledovanie metrík, aby k výsledkom metrík napríklad k rôznym odhaleným chybám v programe pristupovali citlivo a zohľadňovali aj ďalšie problémy ktoré môže nastať zlým prístupom ako demotivácia, podsúvanie klamlivých vstupných dát pre metriky, nedostatočné zapájanie sa do procesu zberu dát pre metriky, zatajovanie odhalených problémov atď. Stačil by pravidelný osobný rozhovor medzi manažérmi a vývojármi. Určite by som neradil aby sa preberala výkonnosť konkrétneho človeka, pred jeho spolupracovníkmi. Vhodné by bolo aj hodnotenie problémov alebo obtiažnosti odvedenej práce vývojármi, podľa čoho by sa dalo sledovať ako konkrétny človek znáša prácu ktorá mu bola pridelená. Takáto úzka spolupráca medzi manažérmi a vývojármi je veľmi dôležitá, ale aj časovo náročná, hlavne pre manažérov, ktorý by museli riešiť všetky tieto problémy osobne.

Ďalším odporúčaním, ktoré by som navrhol je obsadzovať konkrétne pozície ľuďmi, ktorý kedysi vykonávali pozície, ktoré momentálne monitorujú. Tým by sa dosiahlo práve tá lepšia spolupráca, keďže takýto človek by si vedel vžiť do problémov ktoré zažívajú

jeho kolegovia. Skúsenosť na danej pozícii ale stále nezaručuje že takáto komunikácia bude fungovať, treba hlavne vyberať ľudí empatických, komunikatívnych a schopných pracovať a motivovať ľudí. Práve preto si myslím že dosadenie správnych ľudí na pozície zodpovedné za monitorovanie a vyhodnocovanie metrík je jedným z kľúčových faktorov kvalitného monitoringu.

Záver

Monitorovanie softvérového projektu nemožno chápať len ako sledovanie vybraných metrík a konanie na základe ich výsledkov aplikovaním najčastejšie sa vyskytujúcich problémov, vyplývajúcich z týchto nameraných hodnôt. Túto problematiku treba chápať aj v širších súvislostiach, treba si byť vedomí nedostatkov týchto metód, príčin týchto nedostatkov ako aj spôsobov ich nápravy. Ďalším faktorom, ktorý treba brať do úvahy je že ľudia sú spokojný len do tej miery, pokým sú ich hodnotenia pozitívne. Správnym prístupom k vyhodnocovaniu týchto metrík je možné dosiahnuť ušetrenie nemalých nákladov, ako aj dodržanie naplánovaného časového harmonogramu.

Použitá literatúra

1. Pandey, R.K.: Relativity in software engineering measurements. *ACM SIGSOFT Software Engineering Notes*, vol. 34, No. 2, (2009), Dostupné na internete: <http://portal.acm.org/citation.cfm?doid=1507195.1507211>

2. Cote, V.: Software metrics: An overview of recent results. *Journal of Systems and Software*. (1988), 121-131.

3. Garcia, C.A.L, Hirata, C.M.: Integrating functional metrics, COCOMO II and earned value analysis for software projects using PMBoK. *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*. (2008), Dostupné na internete : <http://portal.acm.org/citation.cfm?doid=1363686.1363873>

4. Hall, T., Fenton, N.: Implementing effective software metrics programs. *IEEE Softw.*, vol. 14, (1997).

5. Umarji, M., Seaman, C.: Why do programmers avoid metrics? *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '08*. (2008), Dostupné na internete : <http://portal.acm.org/citation.cfm?doid=1414004.1414027>

6. Umarji, M., Seaman, C.: Gauging acceptance of software metrics : Comparing perspectives of managers and developers. *Information Systems*. (2009), 236-247.

Anotation

Influence of Measurment on software project

Monitoring of software project is highly important due to control of observing plan. There are metrics which serve us to monitor the state of the project , with different measure of complexity, relaibility so as evaluation value. The choosing of the correct metric and sensible approach of their evaulation might have crucial influence on time keeping of planned harmonogram. This essay deals with the disputableness of measure in which we could rely upon metrics as well as concluding results from them and simultaneously deals with the probability of revealing the correct reason with assistance of metrics. The essay makes effort to point the importance of human approch in choosing proper metrics and correct evaluation of their results by analyzing and comparing of different methods of software project monitoring.