AKO VYBERAŤ SOFTVÉROVÉ METRIKY, KTORÉ NEKLAMÚ

Čo tak odmerať spokojnosť zákazníka riadkami zdrojového kódu.

Martin Jaborník

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

Abstrakt. *Pri monitorovaní softvérového projektu sa sledujú jeho vybrané metriky. Navzájom sa môžu od seba líšiť úrovňou, uhlom pohľadu na projekt a tiež svojou presnosťou. Úspešnosť projektu, jeho dynamickosť a pružnosť zabezpečuje práve monitorovanie. Preto je dôležité zamyslieť sa nad tým, ktoré metriky vyberať. Kedy nám metriky poskytujú pravdivý obraz o projekte a kedy môžu naopak zavádzať? Esej uvažuje zaužívané spôsoby merania softvérového projektu, ich kladné a záporné stránky. Okrem toho naznačuje nové myšlienky, ako s metrikami pracovať. Každý merací postup vedie k samostatným výsledkom. Môžu tieto výsledky poskytnúť viac ako priamočiaru odpoveď? Medzi zdanlivo nesúvisiacimi metrikami existujú skryté prepojenia. Výsledok jednej metriky sa možno dá vyrátať inou metrikou. Nikde to nie je napísané, ale intuitívny pohľad na problém to potvrdzuje. Esej hľadá nové súvislosti a snaží sa novými pohľadmi na vec posunúť hranice využiteľnosti merania softvérového projektu.*

Kľúčové slová: *meranie, softvérové metriky, softvérový projekt, monitorovanie*

Úvod

V každej vedeckej oblasti vytvára meranie kvantitatívny opis príslušných procesov a vecí, čo umožňuje pochopiť ich správanie a výsledok. Dôsledkom je výber lepšej techniky

a postupov a následné zdokonaľovanie procesov a vecí. Z výskumnej úrovne sa časom nové merania pretavujú do praxe [3]. V prípade informatiky tomu nie je inak.

Pri práci na softvérových projektoch má meranie svoj význam. V prvom rade si treba uvedomiť dva aspekty tohto typu projektu. Na jednej strane je to projekt ako každý iný a preto je potrebné merať netechnické vlastnosti projektu. Jeho predmetom je softvér, ktorý meriame inými metódami – softvérovými. Aby malo meranie zmysel, musí dávať použiteľné výsledky. Výsledky, ktoré dostaneme, záležia najmä od výberu metrík. Dobrý výber je podmienený poznaním súvislostí, dôsledkov, výhod a nedostatkov spôsobov merania.

Riadenie projektu a metriky

Vybrané projektové metriky by mali zodpovedať potrebám zákazníka a zároveň by mali pokrývať interné potreby merania projektu. Aby boli ľahko použiteľné, mali by byť jednoduché, priamočiare a zmysluplné. Majú vytvárať „jazyk“ medzi rôznymi členmi tímu.

V návrhu metrík na konkrétny projekt je potrebné uvažovať ako sú konkrétne metriky prepojené s kľúčovými metrikami biznisu. Zvyčajne sa nedá nájsť metrika, ktorá by napĺňala všetky požiadavky konkrétnej situácie. Klasický prístup, ktorý tímy používajú je pochopenie problému, brainstorming rôznych metrík a na záver výber tých najvhodnejších. Tím potom konzultuje metriky s výkonným manažmentom, aby sa uistil, že vybrané metriky sú v súlade so stratégiou. K finálnemu zoznamu sa spoločne dopracujú napríklad iteratívnym prístupom [5].

Pri prvotnom návrhu metrík projektu je podstatné uvedomiť si, čo je merané. Metriky by mali byť založené na tom, čo potrebujeme merať, aby sme mohli vylepšiť proces. Spôsoby merania by sa nemali prispôsobovať systému. Nezáleží na tom ako budú vybrané metriky spolu zladené, ale na tom ako každá jednotlivá prispeje k získaniu čo najlepších výsledkov. Okrem tohto jednoduchého postupu pri výbere metrík je možné uplatniť aj konkrétne metódy. Jednou z často používaných je Balanced Scorecard (BSC). Príklad Balanced Scorecard je uvedený v tabuľke č. 1.

Tab. 1. Príklad vyváženej tabuľky metrík projektu [5].

Finančné	**Zákaznícke**
Úrovne zásob	Spokojnosť zákazníka
Náklady na jednotku	Dodávka na čas
Skryté financie	Kvalita finálneho produktu
Financovanie aktivít	Bezpečnostná komunikácia
Náklady kvôli nízkej kvalite	
Celkové úspory projektu	
Interné biznis procesy	**Vzdelávanie zamestnancov a rast**
Závady, kontrola dát, DPMO	Kvalita vzdelávania
Doba cyklu	Efektívnosť
Dodávaný objem	Celkové vzdelanie
Kvalita dodávateľa	Časový harmonogram verzus aktuálny dátum

Táto metóda zohľadňuje finančné aj nefinančné metriky, metriky využívané v úvodnej časti projektu, ale aj tie, ktoré nadobúdajú dôležitosť vo finálnych fázach [4]. Metriky aplikuje v štyroch perspektívach – finančnej, zákazníckej, internej a personálnej. Pri vytváraní Balanced Scorecard je odporúčané využívať brainstorming. Treba ale pamätať na časť metrík, ktoré sú podstatné pri finalizácii projektu.

O softvérových metrikách

Softvérové metriky ako pojem a ako mierky vlastností softvéru existuje na vedeckej scéne už polovicu storočia. V skutočnosti sa však dá povedať, že len malá časť poznatkov z výskumu metrík softvéru je prenášaná do praxe a uplatňovaná pri realizácii projektov. Vedomosti projektových manažérov často končia pri vymenovaní metrík [3]. Po detailnejšom prieskume zistíme, že v skutočnosti už vieme o metrikách oveľa viac. Niektoré sú spoľahlivé, iné naopak veľmi nepresné, ďalšie existujú len na teoretickej úrovni. Napriek tomu dokážu poskytnúť prínosné výsledky. Encyklopedická a vzdelávacia literatúra aktuálne výsledky v tomto smere len málo zohľadňuje. Stále je predostieraný zoznam metrík, ktoré sa dajú použiť, bez ohľadu na pravdivosť výsledku a uvedenia podmienok, ktoré musia byť pri ich použití dodržané. Ak prejdeme k metrikám, ktoré sú používané, narazíme na ďalšie problémy. Výsledok, ktorý má vysokú mieru nepresnosti, nemusí byť vždy správne využitý. Dôsledkom je chybné rozhodnutie a tým degradácia projektu. Nestačí preto poznať metriky ale tiež vedieť s nimi pracovať.

Kým sa metriky nebudú používať korektne a nesprávne metriky používať bezstarostne, nemôžme hovoriť o softvérovom inžinierstve, ale iba o práci s hrubými odhadmi namiesto presných meraní [2].

Zavádzajúce softvérové metriky

Niektoré softvérové metriky sú využívané často a sú brané ako samozrejmosť pri získavaní štatistík o stave projektu. To však nezaručuje ich spoľahlivosť. Medzi na pohľad jednoznačné a široko využívané metriky patria riadky kódu (LOC metriky), Halsteadove metriky a metriky uvažujúce náklady na chybu [2]. Pri ich nesprávnom pochopení hrozí, že môžu projektu nie pomôcť, ale uškodiť.

LOC metriky uvažujú ako základnú výrobnú jednotku jeden riadok zdrojového kódu. Tu vzniká prvý problém – nepresnosť. Nikde nie je zadefinovaná dĺžka riadku kódu ani jeho obsah. Takúto prekážku vieme odstrániť dodefinovaním. Oveľa väčšie komplikácie nastanú pri uvažovaní rôznych programovacích jazykov. Dnes ich existuje viac ako 400. Čo sa stane ak chcem porovnať dva softvérové produkty s tými istými vlastnosťami ale vytvorené v rôznych jazykoch? Zoberme si jednoduchý príklad. Projekt 1 vytvorí program v jazyku Assembler v rozsahu 15000 riadkov kódu. Napísanie tohto kódu trvá 10 mesiacov. Ďalšie práce ako dokumentácia, analýza trvajú 5 mesiacov. Potom celý projekt potrvá 15 mesiacov a na každý mesiac pripadá priemerne 1000 riadkov kódu. Ak budú celkové náklady na vývoj 3000 € na mesiac, jeden riadok kódu bude stáť 3 €. Teraz uvažujme Projekt 2, ktorý vyvíja ten istý produkt v jazyku Java. Na vytvorenie programu je v tomto prípade potrebných iba 3000 riadkov kódu. Réžia projektu potrvá opäť 5 mesiacov ale čas písania zdrojového kódu sa skráti na 5 mesiacov. Preto na mesiac

pripadne 300 riadkov kódu. Pri nákladoch 4000 € na mesiac stojí jeden riadok kódu 13,3 €. Z príkladu vyplýva, že LOC metriky penalizujú vyššie programovacie jazyky. I keď nás vyjde jeden riadok kódu viac, pri zohľadnení celkových nákladov na projekt zistíme, že sa ho oplatí uprednostniť. Vždy správne výsledky by sme dostali, ak by sme prevádzali každý zdrojový kód z rôznych programovacích jazykov na jednotný tvar.

Halsteadove metriky pracujú s kódom komplikovanejším spôsobom, ale výsledkom sú tie isté anomálie ako pri LOC metrikách [2]. Preto sú nespoľahlivé.

Metriky využívajúce náklady na chybu ako ukazovateľ, sú ďalším kandidátom zavádzajúcich metrík. Tu je znevýhodňovaná kvalita. Výsledky týchto metrík sa zhoršujú s lepšou kvalitou kódu [1]. Ak to premietneme opäť do príkladu, v Programe 1 napísanom v jazyku Assembler nájdeme 150 chýb. Napísanie testovacích prípadov bude trvať týždeň, ich otestovanie ďalší týždeň a oprava ďalší mesiac. Pri mesačných nákladoch 3000 € je cena celého procesu odstránenia chýb 4500 € a teda cena jednej chyby 30 €. Teraz si zoberme Program 2 s tými istými vlastnosťami ako Program 1, ale napísaný v jazyku Java. Tento bude obsahovať iba 30 chýb. Na vytvorenie testovacích prípadov a na otestovanie postačí jeden týždeň. Na opravu chýb spotrebujeme dva týždne. Ak sú mesačné náklady rovnaké ako v prvom prípade, čiže 3000 €, celková cena opravy chýb bude 2250 €, čo je menej ako pri Programe 1. Preto by sme logicky volili Program 2 – vyšší programovací jazyk. Čo nám ale hovorí meranie nákladov na chybu? Pri hore uvedených celkových nákladoch na opravu Programu 2 sa nám cena jednej chyby vyšplhá na 75 € za jednu chybu. To je viac ako dvojnásobok ceny chyby v Programe 1. Podľa metrík nákladov na chybu sa javí Program 2 ako menej výhodný.

Z uvedeného príkladu môžeme konštatovať, že výrazné zníženie celkových nákladov je pri nepremyslenom meraní na báze ceny chýb neviditeľné. Inak povedané, dobrovoľne uprednostníme kvalitatívne horšiu a finančne menej výhodnú možnosť.

Aj softvér sa dá merať spoľahlivo

Na druhej strane máme aj metriky, ktoré poskytujú spoľahlivé informácie. Medzi ne zaradzujeme metriky zložitosti a metriky funkčných bodov. Obidva prístupy sú známe už od druhej polovice 20. storočia.

Pravdepodobnosť chybovosti rastie úmerne so zložitosťou softvéru. Zvýšená chybovosť vedie k zníženiu produktivity pri vývoji rovnako ako pri obsluhe. V praxi sa potvrdzuje, že prehnaná zložitosť je okrem výnimočných prípadov spojená s priemernou, až podpriemernou kvalitou a nadpriemernými výdavkami na prevádzku. To je dôvod, prečo je potrebné tieto metriky používať. Zložitostnou metrikou môžeme kvantifikovať každý všeobecný výrok. Vo výskume bolo zavedených viac spôsobov, ako sledovať zložitosť. Jednou z nich je podmienená zložitosť [2]. Je to miera vetvenia kontrolného toku programu. V ideálnom prípade sa program nevetví. Vtedy je hodnota zložitosti 1. Pri grafickom zobrazení tejto metriky sa dá vyčítať, kde sa program neopodstatnene vetví a teda kde je možné zjednodušiť ho.

Celková zložitosť je odvodená od podmienenej. Jej koncept je podobný, ale je obohatená o techniky na odstránenie redundancie. Problém rozdielnych programovacích jazykov, ktorý sa vyskytol v predchádzajúcich metrikách tu nehrozí. Zložitosť je možné

určiť manuálne, ale v súčasnosti existujú automatizované riešenia. Tie sú prispôsobené jednotlivým programovacím jazykom a ako vstup používajú priamo zdrojový kód.

Metrika funkčných bodov je ďalšou zo skupiny „pravdovravných". Funkčné body sú vážené sumy piatich externých atribútov – vstupov, výstupov, dopytov, logických súborov a rozhraní – ktoré ovplyvňujú zložitosť [2]. Pravidlá pre výpočet funkčných bodov sú pomerne komplexné, ale to je dnes opäť úlohou automatizovaných podporných prostriedkov.

Prečo rieši meranie funkčných bodov problém rôznych programovacích jazykov? Napríklad dva programy, prvý v jazyku Assembler, druhý v jazyku Java, budú obsahovať rovnaké funkčné body, pokiaľ budú oba vykonávať rovnaké funkcie. Ak zoberieme do úvahy fakt z predchádzajúcich príkladov, že vo vyššom programovacom jazyku trvalo vytvorenie programu kratšie dostaneme nasledujúci výsledok. Na jednu časovú jednotku pripadne viac funkčných bodov pre vyšší programovací jazyk ako pre nižší. Po prerátaní na finančné výdavky sa ukážu funkčné body tvorené nižším programovacím jazykom ako drahšie. Metrika funkčných bodov určí za výhodnejší vyšší programovací jazyk, čo je očakávaný správny výsledok.

Novším riešením, ktoré odstraňuje nedostatky LOC metrík, ale zároveň ich používa je technika nazývaná „backfiring". V princípe ide o kombináciu LOC a funkčných bodov. Metóda je využívaná hlavne v automatizovaných výpočtoch [2].

Ako ešte merať softvér?

Dôležitou súčasťou vývoja softvéru je práca s informáciami a údajmi. V tejto oblasti nie je využívanie metrík natoľko rozvinuté. Pri každom softvérovom projekte sú založené úložiská a databázy. V nich sa zbierajú, vytvárajú, premiestňujú, vymazávajú údaje. Databázoví experti odhadujú, že cena vytvárania a vyžívania údajov je zhruba porovnateľná s cenou vytvárania softvéru [2]. Preto má zmysel uvažovať aj nové, zatiaľ zriedkavo používané metriky.

Ako prvá ma napadá kvalita údajov. Dobrá štruktúrovanosť údajov v databáze je často riešená otázka. Mojim zámerom je zamyslieť sa nad dôsledkami, ktoré z tejto vlastnosti vyplývajú. Je isté, že nízka kvalita zvyšuje nároky na výkon. Projekt postihnutý týmto nedostatkom je kandidátom na časté problémy, zbytočnú zložitosť, časté požiadavky na zmenu. Poznáme pravidlá správneho návrhu organizácie údajov, ale hotovú štruktúru nevieme odmerať.

Ďalšou možnosťou, ktorá sa doposiaľ neuplatnila pri meraní vlastností softvéru je pohľad na systém ako celok [2]. Kompatibilita hardvéru a softvéru a ich vzájomná spolupráca ovplyvňuje kvalitu vytvoreného celku. Hardvér má vlastný mikrokód, softvér má svoj zdrojový kód. Je pomerne jednoduché predstaviť si meranie obidvoch súčastí samostatne. Ako ale odmerať kvalitu výstupu celku?

Metriky môžu navzájom komunikovať

V predchádzajúcich odsekoch boli opísané softvérové metriky a metriky projektu samostatne. Nebola uvažovaná možnosť prepojenia metrík. Otázka, ktorú kladiem, znie: „Existuje súvislosť medzi metrikami, ktoré merajú na úplne odlišnej báze?" Môj názor je

kladný. Dospel som k nemu hľadaním konkrétnych na pohľad nesúvisiacich kombinácií projektových a softvérových metrík. Túto cestu vidím ako šancu na dosiahnutie nových lepších výsledkov. Vo zvyšku kapitoly ponúkam dva zo svojich nápadov.

Čo tak odmerať spokojnosť zákazníka riadkami zdrojového kódu? Zákazník, často krát bez odborných znalostí, ani poriadne nechápe čo je zdrojový kód. Prečo by teda mal tento kód ovplyvňovať jeho spokojnosť? Na vyriešenie problému využime niekoľko všeobecne známych pravdivých závislostí. Čím je zdrojový kód dlhší, tým viac chýb sa v ňom nachádza. Čím viac je v kóde chýb, tým dlhšie trvá ich oprava. To predlžuje vývoj softvéru a oddiali sa jeho dokončenie. S nedodržanými termínmi klesá spokojnosť zákazníka. Výsledkom implikácií je, že čím dlhší je zdrojový kód tým menej je zákazník spokojný. Podobne by sa dalo uvažovať v smere skracovania kódu.

Medzi projektové metriky patrí aj spokojnosť členov tímu. Zaradzuje sa zvyčajne medzi interné metriky spoločnosti. Vyjadruje napríklad zaťaženie zamestnancov, pracovné podmienky na projekte. Spokojnosť môže ovplyvniť naozaj všeličo. Podľa môjho názoru napríklad aj softvérová metrika zložitosti. Čím je projekt zložitejší, kvôli chybám v analýze alebo vo vývoji, tým viac problémov počas celej doby trvania projektu nastane. Veľa nevynútených problémov znamená prácu navyše. Zväčšený objem práce sa prenáša na zamestnanca. To spôsobuje prácu pod zvýšeným tlakom. Stres samozrejme prispieva k zníženiu spokojnosti zamestnanca. Preto pri zistení vysokej nevynútenej zložitosti projektu sa dá predpovedať vysoké riziko nespokojnosti pracovníkov v tíme.

Podobných kombinácií by sa našlo určite oveľa viac. Mojim zámerom však bolo ukázať myšlienkové smerovanie, ktoré podporím jasnými príkladmi. Uvedené dvojice naznačujú, že dôkladným hľadaním je možné pospájať skupinu softvérových a projektových metrík do grafu. Potom by sme dokázali zistiť nové informácie, napríklad ktorá metrika je kľúčovejšia a na ktoré sa treba zameriavať viac.

Záver

Na konci eseje si uvedomujem, že problematika metrík softvérového projektu nie je priamočiara. Od svojho počiatku už prešla malým vývojom, ale stále má veľa pred sebou. Mnohé metódy používané dodnes v praxi boli identifikované krátko po samotnom vzniku celej problematiky. Preto je ich použitie v súčasných projektoch často riskantné a plné nástrah. To mi úplne nejde dokopy s tendenciou exponenciálneho rozvoja informačných technológií. Obrazne povedané: „Nemeriame čas ešte stále presýpacími hodinami?“ Ako bolo v úvode povedané, softvérové metriky nezaostávajú v pravom slova zmysle. Mnohé však zostávajú vo výskume. Tam asi svoj účel nenaplnia.

Použitá literatúra

1. Fenton, N.E., Neil, M.: Software metrics: successes, failures and new directions. *The Journal of Systems and Software* 47, Centre for Software Reliability, City University, London (1999), 149-157.
2. Jones, C.: Software metrics: Good, bad and missing. *Computer*, Vol. 27, No. 9 (1994).

3. Pfleeger, Lawrence, Jeffery, Ross, Wales, South, Curtis, Bill, Metrics, Teraquest, Kitchenham, Barbara: *Status report on software measurement*. IEEE Software, 1997.
4. Phadnis, S.: Selection of Project Metrics. Dostupné na internete: http://www.isixsigma.com/library/content/c011008a.asp, [cit: 2009-Október].
5. Willard, B.K.: Project Success: Looking Beyond Traditional Project Metrics. Dostupné na internete: http://www.maxwideman.com/guests/metrics/abstract.htm, [cit.: 2009-Október].

Annotation

How to choose not lying software metrics

Monitoring of software project includes using chosen metrics. Each other may vary from level, view or accuracy. Success of every project, its dynamic and flexibility is influenced by monitoring. It is important to choose metrics carefully. Sometime they are correct, another time they show wrong way. The essay considers traditional measuring methods. It tries to explain which metrics are reliable. Further it is finding new connections between completely different methods. Finally essay suggests several new possibilities of measurement in software project.