Potreba monitorovania softvérových projektov

PETER JANIČKOVIČ

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
janickovic@creeo.net

Abstrakt. Základom pre úspešný projekt je jeho dôkladná príprava ako aj monitorovanie jeho samotného priebehu. Predkladaná esej preto venuje pozornosť monitorovaniu softvérového projektu v jeho rôznych aspektoch. Neštandardné riešenia v návrhu spôsobené zmenenými požiadavkami zadávateľa v pokročilom štádiu projektu, komplikované a nečitateľné prvky v implementácii znižujú kvalitu výsledného produktu, čo následne spôsobuje veľké problémy pri údržbe. Pri precíznom sledovaní projektu, jeho atribútov kvality a správnou a najmä pružnou reakciou na vzniknuté problémy, je možné vyhnúť sa komplikáciám alebo predčasnému a neúspešnému ukončeniu projektu.

Úvod

Základom pre úspešný projekt je jeho dôkladná príprava, ako aj monitorovanie jeho samotného priebehu. Proces tvorby softvérového systému je veľmi podobný procesu stavby budovy. V prvotnej fáze oboch projektov je úlohou pripraviť plán. V oboch typoch projektov je nutné použitie metriky na určenie vývoja projektu. Metrika vytvára akúsi spätnú väzbu pre plán. Na základe výsledkov merania softvéru je možné upraviť plán projektu tak, aby projekt nebol ohrozený z akéhokoľvek hľadiska. Koniec koncov, niektoré zmeny v pláne projektu môžu byť odhalené ešte v začiatočných fázach.

Meranie softvéru je náročné, no je neodmysliteľnou súčasťou v oblasti softvérového inžinierstva. V nasledujúcom texte sa pokúsim opísať základné princípy metriky softvéru a odporučiť niektoré z nich ako vhodné pre malé a stredne veľké projekty. Samotné meranie softvéru však nemá žiadnu výpovednú hodnotu, ak nie je vhodne naplánované.

Manažment v softvérovom inžinierstve, október 2006, s. 1-8.

Ako merať

Je veľmi dôležité osvojiť si potrebu merania softvéru. Používať podporné prostriedky na zjednodušenie tejto činnosti, aby čas strávený meraním bol čo najmenší. Každodenné meranie by sa malo stať návykom, nie bremenom. Je nutné zaznamenať všetky problémy, sledovať ich vývoj v konkrétnych nástrojoch. Podporné prostriedky na to určené identifikujú časti, ktoré obsahujú veľké množstvo problémov.

Podporené prostriedky sledovania projektu umožňujú monitorovať projekt z rôznych uhlov pohľadu. Väčšina z nich podporuje Ganttov diagram, ktorý naznačuje akýsi plán projektu. Definujú sa úlohy v čase a ich závislosť od iných úloh. Každá takto vytvorená úloha je pridelená jednotlivcovi alebo tímu. Práve tu nastáva malý problém. Niektoré podporené prostriedky na základe času vyjadrujú stav konkrétnej úlohy v percentách. Ak systém označí úlohu z polovice za ukončenú, lebo čas vyhradený na túto úlohu je v polovici, evokuje to ilúziu resp. mylnú predstavu. Je totiž veľmi ťažko odhadnuteľné či daná úloha je naozaj v polovici svojho riešenia. [5]

Uvediem príklad: Zadá sa úloha naprogramovať istý algoritmus a vyhradí sa na ňu 5 dní. Po uplynutí 3 dní je programátor úplne bezradný, lebo algoritmus nefunguje. Má ho síce naprogramovaný, systém označuje úlohu ako ukončenú z troch pätín, no manažér netuší, čo sa deje. Najhorší možný koniec tejto situácie je, že programátorovi sa nepodarí opraviť algoritmus ani počas zostávajúceho času. Tento nežiaduci jav riešia podporné prostriedky tak, že umožňujú riešiteľovi tejto úlohy, definovať jeho odhadovaný čas na jej ukončenie. To umožní manažérovi včas reagovať na nesúlad medzi naplánovaným časovým harmonogramom a skutočným priebehom. [1]

Koho merať

Meranie by malo prebiehať na rôznych úrovniach organizácie. Najnižšou úrovňou sú jednotlivci, kde je potrebné merať rozloženie ich úsilia. Ďalej treba porovnať čas, ktorý strávili nad úlohami s časom, ktorý bol predpokladaný na ich vykonanie. Odporúča sa (relatívne k množstvu) zaznamenávať počet nájdených chýb v testovaní kódu, ktorý bol vytvorený jednotlivcami. V neposlednom rade musíme zaznamenávať kvalitu kódovania jednotlivca.

Ďalšou úrovňou meranie je projekt, kde je nutné sledovať veľkosť produktu, rozloženie výkonu a percento úspešných testov. Musíme porovnávať predpokladaný harmonogram s reálnym, ďalej úrovne jednotlivcov (ako sa jednotlivec vyvíja, zlepšuje sa, zhoršuje sa) ako aj počty problémov vyskytujúcich sa v rôznych úrovniach vývoja (programovanie, testovanie a pod).

Najvyššia úroveň meraní sa týka organizácie, kde merania podávajú výpovednú hodnotu o jednotlivých projektoch, tímoch a to v rôznych aspektoch ako sú napríklad náklady, znovu použiteľnosť, časová precíznosť a dĺžka životného cyklu projektu (relatívne v veľkostiam). [5]

Základným problémom týchto meraní je však reakcia ľudí na to, že je nutné merať! Ľudia sa boja, že dáta, obzvlášť tie, ktoré sa týkajú ich ako jednotlivcov, budú použité proti nim. Majú strach z toho, že zbieranie a spracúvanie týchto informácií zaberie neúmerne veľa času. Zabezpečujeme, aby sa zbieranie a vytváranie týchto dát stalo zvykom a takto sa docieli to, aby tím umelo neudržiaval koeficienty merania v normále na úkor kvality softvéru! Aby sme pomohli ľudom preklenúť prvotný strach z týchto činností, musíme im zdôrazniť aké dôležité je meranie, ubezpečiť ich, že výsledky meraní nebudú mať vplyv na ich ohodnotenie. Nikdy to nesmie dospieť do naznačeného stavu. Ak budeme ohodnocovať tím na základe týchto informácií, nepoužívame metriky na zlepšenie kvality, ale na to aby sme šetrili! V neposlednom rade sa nesmie zabudnúť na ochranu týchto údajov na každej úrovni merania. Vytvorenie tzv. kultúry merania a sledovania a odstránenie vzdoru voči týmto praktikám vytvorí vynikajúcu východiskovú pozíciu pre úspešné ukončenie projektu.

Čo merať?

V úvode som naznačil, prečo je dôležité merať softvér. Dôležitou častou celej problematiky je však zvoliť správne aspekty merania, teda zvoliť správne metriky. Exituje mnoho rôznych spôsobov ako merať softvér, no najlepšia cesta je zvoliť si malú a vyváženú sadu metrík, ktorých výpovedná hodnota umožní sledovať približovanie sa k stanoveným cieľom.

Metóda GQM (Goal-Question-Metric teda Cieľ-Otázka-Metrika) je vhodný spôsob pre výber metrík. Túto metódu predstavil Victor Basili (profesor Univerzity v Maryland-e a člen tímu softvérových inžinierov NASA). Metóda definuje model merania na 3 úrovniach [5]

- Koncepčná úroveň (cieľ): Je stanovený cieľ pre objekt, pre rôzne dôvody, s prihliadnutím na modely kvality a z rôznych uhlov pohľadu. Typickými príkladmi sú napr. dokumenty, výrobky, ale aj procesy napr. testovanie, špecifikovanie.
- Operačná úroveň (otázky): Množina otázok popisujúca spôsoby určenie a dosiahnutie cieľa na základe modelu. Otázky sa snažia charakterizovať objekt z koncepčnej úrovne s prihliadnutím na daný atribút kvality zo zvoleného uhla pohľadu.
- Kvantitatívna úroveň (metrika): Množina metrík, založená na modeli, je priradená ku každej otázke, aby na ňu kvantitatívne odpovedala. Jedná sa o atribúty typu: množstvo hodín, počet verzií, veľkosť programu alebo úroveň spokojnosti zákazníka, čitateľnosť kódu.

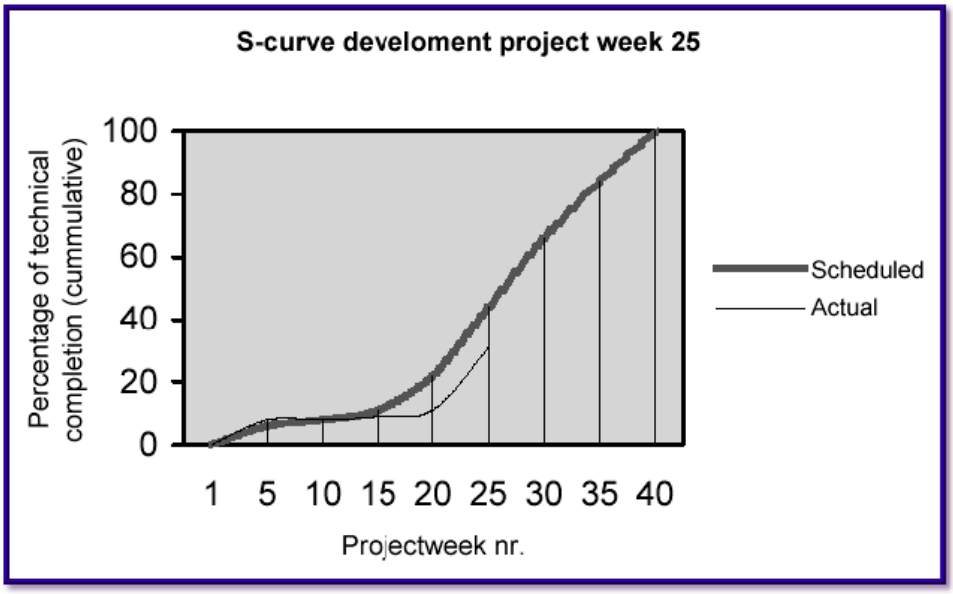
Na objasnenie uvediem jednoduchý príklad: Ako cieľ si stanovím zníženie ročných nákladov o 50% (Relatívny zmysel nášho cieľa nie je náhodný, je oveľa vhodnejšie použiť takúto formuláciu cieľa. Neodporúča sa napr. stanoviť cieľ formulovaný absolútne, teda „zníženie nákladov na 10%". Takýto cieľ totiž nehovorí o aktuálnom stave, čo keď ste už prekročili hranicu 40% nákladov???). Následne si položíme otázky, ktorých odpoveď nám povie, ako plníme svoj cieľ. Teda vhodné

otázky sú nasledovné: Koľko minieme za jeden mesiac?; Akú časť nákladov míňame na jednotlivé projekty ?; Akú časť nákladov míňame priebežne na vyžiadané zmeny, opravy a pod? Nakoniec identifikujeme metriky, ktoré nám zodpovedajú jednotlivé otázky.

Ďalšou možnosťou ako si zvoliť metriky je použiť doporučenú sadu metrík. Nasledujúce podkapitoly stručne popisujú výber najdôležitejších a najpoužívanejších metrík zhodné s odporúčaniami CMM (Capability Maturity Model) [6]

Postup (Progress)

Poskytuje informáciu o tom, ako postupuje projekt v súlade so svojím časovým harmonogramom. V tomto prípade porovnávame počty plánovaných a reálne vykonaných úloh a dĺžky trvania jednotlivých úloh v porovnaní s ich očakávanou časovou náročnosťou. Je však nutné, aby každá úloha mala jasne definované vstupy a výstupy, aby bolo možné jednoznačne určiť či bola, alebo nebola splnená. V rámci projektu sa stanovuje istá hranica pre výsledky týchto meraní, tieto hranice sú potom vstupmi pre manažment rizík. Nasledujúci obrázok, obrázok č. 1 naznačuje túto metriku, ako percentuálny počet splnených úloh. Jasne je vidno, že projekt stagnoval asi dva týždne.[2]



Obr. č. 1 Náhľad metriky postupu [2]

Snaha (Effort)

Indikátory snahy umožňujú softvérovým koordinátorom sledovať ľudské zdroje. Poskytuje náhľad prispievania jednotlivcov do vynaložených finančných nákladov, udržiavania časového plánu a kvality produktu. Túto metriku je možné použiť na každej úrovni merania. Každá úroveň si tvorí svoju vlastnú „históriu snahy“. Je nutné merať tento aspekt on samotnej inicializácie projektu až po jeho ukončenie a vytvárať správy aspoň raz do mesiaca.

Miera ceny a vynaloženého úsilia sú závislé. Podľa zvyku býva miera úsilia nekumulujúca sa časť vydajov na ľudských zdroje a miera ceny je kumulatívna časť úsilia, daná získanou hodnotou. Preto miera ceny je kumulatívnym zobrazením úsilia.

Výdavky (Cost)

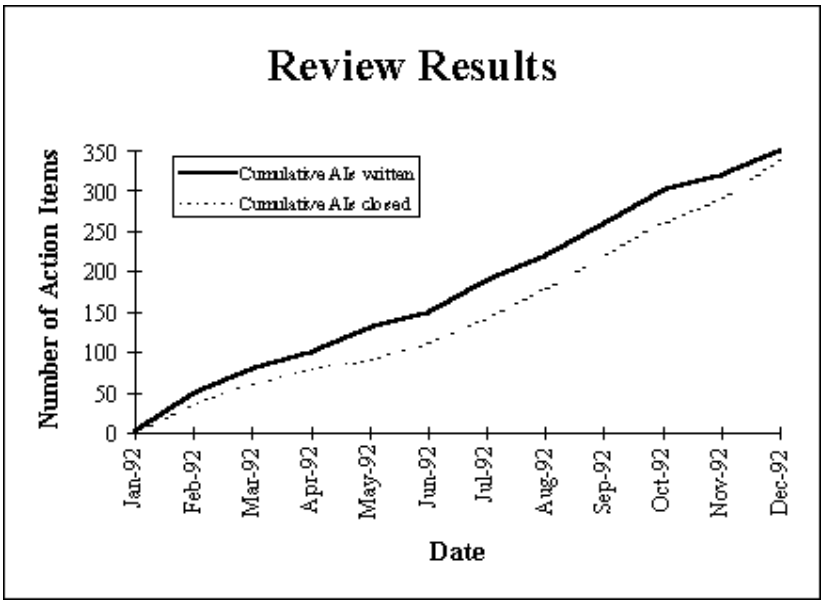
Táto metrika je jedna z najdôležitejších. Zanedbanie jej monitorovania by mohlo mať za následok nutnosť predčasne ukončiť projekt z dôvodu vyčerpania rozpočtu. Musíme si teda stanoviť aj plán čerpania rozpočtu a v každom okamihu mať možnosť porovnať si plán s realitou. Používajú sa človeko-hodiny na vyjadrenie „výdavkov“ nie priamy finančný obnos. Tu je dôležité aby sa plán veľmi nelíšil od reality. Má to klamlivý efekt, manažéri si myslia že sme ušetrili, no opak je pravdou. S najväčšou pravdepodobnosťou sa zanedbalo niečo, čo minie oveľa viac na konci projektu ako teraz. [4]

Je vhodné prerozdeliť rozpočet nie na základe časového plánu, ale na základe úloh. Na najvyššej úrovni sa najväčšia časť rozpočtu rozdelí na analýzu, návrh, implementáciu, testovanie, integráciu a pod. Potom sa delí napr. v implementačnej časti na jednotlivé moduly a pod. Vedúci tímu na základe podporných prostriedkov vidí počet človeko-hodín strávených nad istou úlohou a vie porovnať naplánované náklady so skutočnými.

Výsledky testovaní (Review results)

Výsledky tejto metriky vychádzajú z formálnej kontroly dokumentov, programu a požadovaných cieľov zákazníka. Je nutné kontrolovať súlad návrhu s implementáciou. Osobitne je nutné zdokumentovať a zahrnúť do metriky počty chýb (malých aj veľkých), výskyt takýchto chýb v rámci dokumentu, kódu, ale aj periodicitu výskytu takýchto chýb v časovom aspekte. V neposlednom rade musíme monitorovať stav jednotlivých problémov ako je počet vyriešených, otvorených problémov, priemerný čas potrebný na ich odstránenie. Dôležitou časťou takejto metriky sú aj údaje, ktoré hovoria o tom ako daný problém vplýval na ostatné metriky napr. výdavky, časový harmonogram a pod. [4]

Je nutné sa vyvarovať tomu, aby výsledky týchto meraní boli použité na ohodnocovanie jednotlivcov. Vypracovať správu, ktorá hovorí o týchto výsledkoch je odporúčané robiť aspoň raz za mesiac a vždy po ukončení nejakej etapy projektu.



Obr. č. 2 Náhľad metriky postupu [2]

Obrázok č. 2 naznačuje graf vývoja úloh, problémov alebo chýb (issues, action items) v rámci projektu. Hrubá linka naznačuje počet všetkých položiek a čiarkovaná linka naznačuje počet uzatvorených napr. úspešne vyriešených problémov. Je jasné, že tieto dve linky sa musia, resp. mali by sa na konci projektu spojiť.

Dôležité metriky softvéru

Softvér je možné merať už v štádiu návrhu. Ak sa jedná o objektovo orientovaný návrh, softvérové inžinierstvo definuje dve vlastnosti dobrého návrhu: *nízka zviazanosť* a *vysoká súdržnosť*.

Pojem zviazanosť znamená mieru previazanosti súčiastok alebo inak povedané stupeň závislosti modulov. Vysoká závislosť komponentov je nežiaduca, pretože sa odráža v prílišnej komunikácii medzi komponentmi. Aby bola dosiahnutá nízka zviazanosť, je potrebné, aby moduly medzi sebou komunikovali len pomocou parametrov alebo správami. Nízka zviazanosť znižuje výkon aplikácie. Prínos pre vývoj softvéru je však väčší ako prínos pre výkon programu s vyššou zviazanosťou.

Súdržnosť je stupeň interakcie v rámci komponenty. Iná definícia hovorí, že súdržnosť znamená mieru ako spolu súvisia schopnosti danej triedy. Nízka súdržnosť je typická pre slabý návrh komponentu. Ideálne súčiastka implementuje jedinú logickú entitu alebo funkciu.

Nedodržanie týchto metrík vedie k náročnému udržiavaniu softvéru, softvér je komplikovaný na pochopenie pre nových programátorov a kód je slabo znovu použiteľný.

Záver

„Nemôžete kontrolovať to, čo nedokážete merať“, povedal pán Tom DeMarco. A čo nemôžeme kontrolovať, nemusíme ani robiť. Ak sa rozhodneme pre nejaký projekt, je potrebné aby sme ho správne navrhli, našli pre neho zákazníkov a aby sme disponovali potrebnými prostriedkami, či už finančnými alebo technologickými, na jeho realizáciu. Toto všetko je však iba malá časť úspechu. Najdôležitejšie je vedieť správne a efektívne rozvrhnúť naše zdroje a dispozície a dohliadnuť nad ich presným a včasným použitím. Inak to nie je ani v prípade softvéru. Nekontrolovaný vývoj softvéru je dopredu odsúdený na neúspech a zdroje doňho vložené môžeme považovať za stratené.

Použitá literatúra

1. Nick Jenkins: *A Project Management Primer or "a guide on how to make projects work"*, 2005.
   http://www.nickjenkins.net/prose/projectPrimer.pdf
   aktuálne k 22.10.2006
2. Joop van der Linden: *Monitoring Progress in Software Development*, júl 2003.
   http://www.stsc.hill.af.mil/crosstalk/2003/07/vanderLinden.html
   aktuálne k 23.10.2006
3. Scott Berkun: *Work vs. Progress*, august 2005.
   http://www.scottberkun.com/essays/essay45.htm
   aktuálne k 22.10.2006
4. University of southern California, Software Engineering II, 2001
   http://sunset.usc.edu/classes/cs577b_2001/metricsguide/metrics.html#p33
   aktuálne k 22.10.2006
5. Karl E. Wieger, *Software Development*, July 1999
   http://www.processimpact.com/articles/metrics_primer.html
   aktuálne k 23.10.2006
6. Capability Maturity Model, Wikipedia
   http://en.wikipedia.org/wiki/Capability_Maturity_Model
   aktuálne k 22.10.2006

Annotation

Having your project watched over carefully is the very essence of having it done successfully. Therefore this essay covers the issue of monitoring a software project in various aspects. Non-conventional design solutions caused by changed customer requests in advanced stage of the project, complicated and unclear implementation elements can decrease the final project quality. That eventually makes maintenance very difficult. By having the project watched

closely, checking its quality attributes and by reacting to arising problems in a flexible fashion you can avoid complications and premature project cancellation.