Projekt je spustený, a čo teraz

JÁN ŠARMÍR

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
sarmir@sorudan.sk

Abstrakt. Projekt je perfektne naplánovaný, máme najlepších manažérov, ktorí dohliadajú na celý projekt podľa plánu. Vznikajú však problémy mimo plánu a tie prinášajú určitý sklz projektu. V akom sme stave? Ohrozí to dokončenie projektu? Na to, aby sme boli schopní odpovedať, je potrebné definovať postupy, ktorými budeme projekt sledovať. Tí lepší si z nich vedia vybrať vhodné metriky na to, aby vedeli odhadnúť doposiaľ vykonanú prácu, čo ostáva a následne preplánovať a prispôsobiť tomu manažérov pre riadenie ako aj samotný plán. Budeme teda projekt priebežne sledovať alebo mu necháme voľný priebeh?

Úvod

Softvérový projekt sa od ostatných líši len málo. Porovnajme ho so stavbou domu. Architekt na základe predstavy domácich pripraví plán stavby, posunie projektovému inžinierovi, ten vytvorí plán etáp a rozdelenie prác robotníkov. Zo skúseností vie povedať, koľko ktorá vec vyžaduje človeko-hodín a pridá k tomu nejakú rezervu. Projekt sa rozbehne, stavba sa buduje až postaví. Nech vznikne časový sklz kvôli nedostatku materiálu. Už v danom čase vieme povedať, koľko sa bude čakať a prispôsobiť plán.

V čom je teda softvérový projekt iný? Podstatný rozdiel je, že ako architekt nakreslí plán budovy, softvér nevieme takto jednoznačne reprezentovať ako to bude vo výslednej forme a používať pritom viacero rovín (prostriedkov). Každá má svoju rovinu a ich spojením nedosiahneme presný opis a tie skryté veci odhalíme až neskôr. Napríklad by sme postavili poschodie a osadili dvere, ktoré sa ale nedajú otvoriť, lebo je tam schodisko, tak ich musíme otočiť. Ale aj keď ich otočíme, tak pôjdu otvoriť len do polovice. Musíme preto zbúrať stenu a posunúť ju ďalej.

Nebúrajme, uvažujme a plánujme

Keď budeme veľa upravovať a opravovať zanedbané veci, nakoniec nám z kockového domu vznikne pyramída. Investujme teda veľké množstvo času do analýzy, návrhu a

plánovania. Lenže čo nám asi tak zákazník povie, keď mu povieme že než začneme projekt, budeme 3 roky vymýšľať? Nájde si niekoho iného. Preto si zvolíme zlatú strednú cestu, kde budeme tiež búrať, ale menej. Takýto charakter má vývoj softvérového produktu, zachytávajúci celú jeho podstatu aj vzhľadom na kladené nároky.

Čo keby sme tvorili inkrementy a iterovali k požadovanému cieľu? Vieme si problém rozdeliť na menšie časti, určiť ich ciele a celý postup navrhnúť, naplánovať. Ale ako každý vie, tak ako v celku, aj v časti vznikajú problémy, prinášajúce časový sklz. V tomto prípade ale nemôžme povedať presne, za koľko to opravíme alebo doplníme, len odhadnúť na základe podobných požadovaný čas. Takýchto prípadov môže byť veľa a my by sme chceli vedieť, kde sa práve v pláne nachádzame. Ako na to? Musíme mať najskôr nejaký plán. Čím podrobnejší, tým vieme miesto presnejšie určiť. Čo znamená presnejšie? Definujeme väčšiu množinu pohľadov na problém, ktorá bude určitým spôsobom vyjadrovať hodnotu príslušného inkrementu projektu. Počet odrobených hodín, napísaných riadkov, počet súborov, ich veľkosť, komponentov, … Z týchto príkladov vieme už určiť vhodnú mieru vykonanej práce a teda aj zostávajúcu časť.

Ako a čo merať

Aspektov je veľa a nemôžme zaznamenávať všetky, lebo by sme nič iné nerobili, len merali a merali a nakoniec merali meranie. V dokumente [3] je spomenutá veľmi efektívna technika merania a príklady najvýznamnejších metrík. Vychádza z toho, že si zvolíme množinu metrík, vyberieme malú ale vyváženú množinu tak, na čo sa sústreďujeme. Napríklad čo si vyberieme medzi produktivitou a kvalitou? Najlepšie oboje, lenže oni sa vylučujú, lebo nemôžme robiť rýchlo a kvalitne. Tu si preto definujeme interval, kde sme ochotní sa pohybovať.

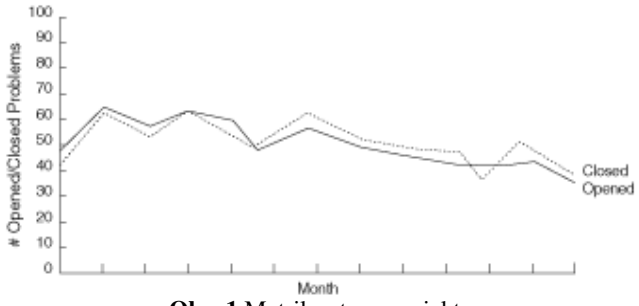
Softvérový projekt môžeme rozdeliť na nasledujúce 3 oblasti :

- *produkt* charakterizujúci mohutnosť, komplexnosť, dizajn, produkcia, požadovaná kvalita. Jeho charakter sa prejavuje najmä na konci životného cyklu, kde jednotlivé body tvoria jeho kvalitatívne ohodnotenie (stredná doba k vzniku chyby, hustota defektov, problémy zákazníka , uspokojenie zákazníka)

- *proces* na zlepšenie efektívnosti vývoja a údržby, odstraňovania chýb (hustota defektov pri testovaní, odstraňovanie defektov vo fázach)

- *projekt*, jeho implementácia, počet špecialistov k oblasti, náklady a plánovanie, prechod medzi stavmi častí (viď obr. 1)

Nesmieme zabudnúť, že ohodnotenie každej metriky by sme radi použili na upravenie vývoja projektu. Nebudeme teda používať také formy, ktoré nám v tomto nepomôžu, ako napríklad merať čas oddychu zamestnanca nám nič nepovie, lebo vhodnejšie je zisťovať odpracovaný čas a vykonané množstvo, čo nám povie o efektívnosti. Prečo by nemohol mať viac voľna aj je potom výkonnejší?



Obr. 1 Metrika stavu projektu

Výpovedná hodnota tejto metriky spočíva v určení v akej fáze sa nachádzame. Keď prevažuje počet uzatváraných problémov, systém sa stáva stabilnejší a je menšie percento otvorenia nových problémov.

V dokumentoch [2] a [4] autori opisujú metriky uvažujúce mieru výdavkov od výnosu pri znovupoužívaní existujúcich systémov, čo sa v praxi často vyskytuje :

- výdavky = 1 / produktivita

  výdavky = (relatívne výdavky na každý nový riadok) * (pomer nových riadkov) + (relatívne výdavky na použitý softvér) * (pomer použitia softvéru)

- kvalita investícií = (výnos znovupoužitia) / (investície znovupoužitia)

  ak je menšia ako 1, projekt je v strate

- relácia znovupoužitia na kvalitu a produktivitu

Tieto metriky ťažko aplikovať na projekty malého charakteru. Kým by sme odhadovali relevantné ohodnotenia, projekt by sme ukončili a rýchlejšie spočítali príjem a výdaj. Má význam ich ale uvažovať na veľkých množstvách, kde sa chyba odhadu minimalizuje a dosiahne sa presnejší výsledok.

Techniky merania

Určitú množinu metrík používa niektorá technika. Ide o definovaný postup, ktorý v špecifických situáciách zbiera údaje a vyhodnocuje ich. Je ich viacero, každá má svoje výhody a nevýhody.

Empiricky

Túto techniku by som považoval za najbližšiu Vychádza z teórie vodopádového vývoja softvéru. Podobne ako je to pri menších projektoch (pokus-omyl). Tak ako sa vraciam k jednotlivým etapám, tak mám zozbierané informácie o aktuálnom stave a keď prejdem späť a pokračujem vývojom, viem porovnať predchádzajúce ohodnotenie.

Goal-Question-Metric

Najjednoduchšou technikou je Goal-Question-Metric (GQM) [5]. Je to pomerne všeobecná technika, ktorá nedefinuje presne časovanie. Najskôr sa určí menší počet všeobecných cieľov a neskôr sa podľa potreby dopĺňajú. V procese je možné použiť ľubovoľnú metriku a určiť jej periódu kontroly, tá sa definuje rozdielom aktuálneho stavu (definuje správa) od plánovaného.

Autor definuje nasledujúce jemu relevantné metriky :

jedinec

- pracovný výkon
- odhadovaný vs. aktuálny čas potrebný na splnenie úlohy
- rozsah pokrytia unit testami
- komplexnosť zdrojových kódov a dizajnu

tím

- mohutnosť
- požiadavky na dosiahnutý stav (schválené, implementované, overené)
- podiel vykonaných testov
- rozdiel vynaloženého času odhadovaného a aktuálneho
- ohodnotenie stability a jej kritických sekcií
- plnenie naplánovaných úloh

organizácia

- pomer množstva známych a opravených chýb
- tvorba znovupoužiteľných častí
- naplánované a skutočné náklady na vývoj

Ako manažér mám vybrané metriky vhodné pre moju situáciu. Aby nevznikala frustrácia u členov v tíme alebo organizácii, je potrebné nielen vykonávať monitorovanie, ale aj vysvetliť prečo to robím, čo tým cielim a ako to môžem ja alebo niekto použiť v prospech jednotlivca alebo skupiny. Pri tomto postupe je vhodné, aby sa napríklad zaznamenávanie počtu odpracovaných hodín stali samozrejmosťou, realizované nejakým prostriedkom. Pre metriky vyžadujúce zber údajov sa musí definovať osoba, najlepšie samotný manažér, ktorý sa bližšie pozná s jednotlivými členmi skupiny a vie využiť získané výsledky napríklad na dosiahnutie lepšej výkonnosti.

Z môjho pohľadu metrika pracovného výkonu nevystihuje dostatočne prácu. Je rozdiel o akú fázu vývoja ide. Tiež je napríklad rozdiel, koľko času venujem

implementácii komponentu na začiatku a koľko na konci a výsledne aká je efektívnosť. Na začiatku je potrebné naštudovať danú problematiku a tam je minimálna produktivita. Preto by bolo vhodné toto ohodnotenie rozdeliť na viacero skupín. Efektivitu vieme určiť od posunu prác v projekte a keď zvážime v akej fáze pracujeme, dostaneme také ohodnotenie. Efektívnosť na začiatku je nízka, pri písaní jadra vzrastie, lebo presne viem čo a ako mám robiť. V závere zasa klesne, lebo funkcionalita je síce pripravená, ale je potrebné ju otestovať. Už aj z tohto dôvodu pokladám nástroje, ktoré ohodnocujú fázy percentuálnym vyjadrením mieri splnenia problému za klamlivý.

Z uvedených metrík skupina používajúca sa na ohodnotenie jedinca sa nedajú vyhodnotiť tabuľkovo, ale subjektívne. Ak by sme uvažovali nasledujúcu techniku RTM a metriky premietli ako vlákna, ktoré sa inkrementovaním došpecifikuje, mohli by sme tento výsledok použiť pri vykresľovaní osobnosti a upravenie tak jeho analýzu.

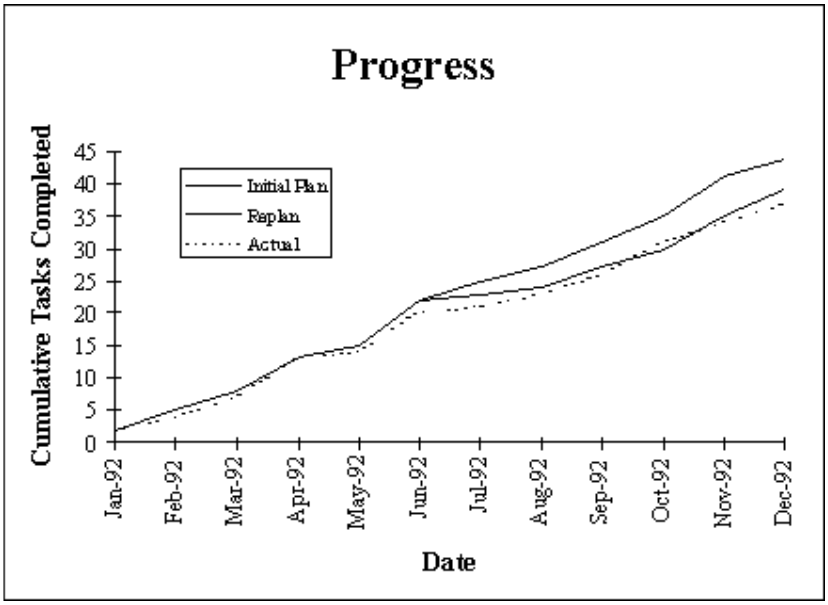
Capability Maturity Model (CMM)

Definuje množinu metrík, ktoré sa sústredenie na zachytenie množstva vykonanej práce. Jeho použité metriky sú nasledovné :

- vývoj – miera potrebnej a vykonanej práce na projekte

  Názorná ukážka vplyvu monitorovania a jeho zaradenie do zmien v plánovaní a riadení je znázornené na obr. 2

- vynaložené úsilie – práce na dosiahnutie naplánovaného stavu pri zachovaní ostatných činiteľov ako kvalita
- náklady
- posúdenie priebehu – ciele, stav kódu, dokumentácie
- problémy – chyby, defekty, požiadavky
- požiadavky
- komplexnosť (mohutnosť)
- zaškoľovanie

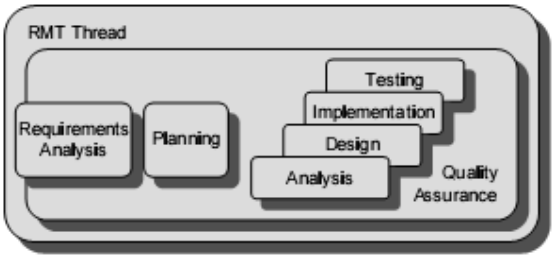


Obr. 2 Metrika určovania plnenia plánu

RMT (Recursive Multi-Threaded)

Autori v dokumente [1] opisujú túto techniku ako súčasť životného cyklu, ktorý má charakter iteračného vývoja založeného na veľkých častiach inkrementov. Projekt sa rozdelí na časti, tie na menšie, tak ako hlboko ide analýza a návrh kde sme schopný definovať jednotlivé kroky a pre ne určiť ciele, míľniky. Vzniká tak robustný strom, ktorého vetvy tvoria vlákna. Tento spôsob organizovania a monitorovania projektu je vhodný pre väčšie projekty, kde sa vyžaduje kontrola veľkého počtu ľudí a je rozumné použiť tvorbu paralelných procesov. Veľké uplatnenie pri objektovo orientovaných projektoch, kde problém dekomponujeme na veľa malých častí.

Každé také vlákno pozostáva z akcií, definovaných cieľov jednotlivých fáz a všetkých vstupov a výstupov. Niektoré časti sa môžu prekrývať s inými vláknami. Nasledujúci obr. 3 opisuje takéto vlákno :



Obr. 3 Vlákno v RMT technike

Každé vlákno musí mať prideleného manažéra, ktorý definuje jednotlivé body vlákna. Proces inkrementov je obsiahnutý v upravovaní jednotlivých komponentov, ktoré reprezentujú vlákna. Ich inkrementovaním dosahujeme vyhodnocovanie aktuálneho stavu a je možné určiť percentuálny odhad kompletnosti vlákna. A čo nám hovorí rekurzia? Tá práve vyjadruje iteračnú časť vývoja, kde každé vlákno (komponent) môže prechádzať procesom úprav (nedodefinované, alebo pozmenené). V prípade zmien môže dôjsť aj k potrebám prispôsobiť väzby medzi vyššie definovanými vláknami v úrovni stromu.

Výhody tejto techniky sú v tvorbe abstrakcie procesu vývoja umožňujúci kontrolu nad komplexnými projektmi. Kroky monitorovania sú dosť jemné a dávajú podrobný prehľad od stave. Jednotlivé vlákna sú percentuálne ohodnotené a je možné tak sledovať postup pre jednotlivé vlákna, skupiny vlákien, ako aj celého projektu. Tu si ale myslím, že čísla nehovoria o všetkom, pretože odhadovať percentuálne množstvo vykonanej a zostávajúcej práce je nerelevantné, najmä keď uvažujeme syndróm 90%. Ako chceme odhaliť výskyt tohto syndrómu a ešte ho aj dokonca vyhodnotiť? Vhodnejšie by bolo reprezentovať diskrétne z nejakej množiny. Tým ale by sme museli definovať takú množinu pre celý projekt, pretože výhodou tejto techniky má byť tvorba abstraktných úrovní. Keby sme ale nemali jednotnú množinu, ťažko by sa spájali rôzne. Uvažujúc veľký projekt, na ktorom pracuje veľká zabehnutá spoločnosť s už vyvinutými subkomponentami, poňatie percentuálneho ohodnotenia môžeme zobrať ako relevantný, keďže už neuvažujeme spomenutý syndróm.

Čo so získanými údajmi

Popísali sme si, aké možné techniky sa dajú nasadiť na monitorovanie stavu projektu. Tiež aj aké metriky pri tom použiť. Pretože každý produkt je jedinečný, aj ohodnocovanie metrík by sa malo vykonať s príslušným zameraním. Tabuľkovo definované metriky to majú v tomto smere ľahšie, ale nemusia dobre ohodnocovať situáciu. Máme zozbierané údaje a chceme minimalizovať vzniknutú odchýlku. Nie je také jednoduché, ako pri výpočtových algoritmoch, pretože tu zohráva rolu aj previazanosť problému s inými časťami. Záleží na manažérovi a jeho zručnosti ako posun posúdi. Navyše uvažujúc väzby medzi časťami, na vyššej úrovni vieme

korigovať pracovné sily medzi mini.

Majme situáciu, že pri tvorbe dizajnu zaznamenávame veľký časový sklz. Metrika pre vývoj nás nepustí a čo teraz? Ako to mám ako manažér zlepšiť, keď tak zle spravená implementácia komponentov pre dizajn, že ich použitie tak dlho trvá? Nepomôže ani prerozdelenie úloh, ani pridanie nových pracovných síl, lebo v potom klesneme v nákladoch. Musí zapôsobiť ľudský faktor na to, aby preniesol do inej sféry problém z dizajnu a tak sa upravil plán na úpravu požadovaného rozhrania.

Záver

Dozvedeli sme sa, prečo je dobré sledovať priebeh vývoja softvérového projektu. Nejde o jednoduchú vec a nie je ani ľahké ju zvládnuť. Vyžaduje si to pár dobre, zle ukončených projektov. Aký je dobrý manažér vidíme na tom, ako zvládne preplánovať projekt na základe vzniknutých výchyliek. Okrem smerovania, riadenia projektu od stavu k stavu cieľovému, vieme odhadnúť čas, náklady potrebné na ukončenie projektu. Tento odhad je však len čisto teoretický, lebo by to nebol softvérový projekt, ktorý nevybočí od svojho plánu.

Použitá literatúra

1. Arturo I Concepcion, Sunny Lin, Scott J. Simon: *Managing the Software Development by Using the Recursive Multi-Threaded (RMT) Tool* (2000)
2. Willian Frakes, Carol Terry: *Software Reuse : Merics and Models* (Jun 1996)
3. Stephen H. Kan: *Software Quality Metrics*. Dostupné na (December 2002) http://www.awprofessional.com/articles/article.asp?p=30306&seqNum=1
4. Everald E. Mills : *Software Metrics*, Seattle University (1988)
5. Karl E. Wiegers: *A Software Metrics Primer*. Dostupné na

http://www.processimpact.com/articles/metrics_primer.htm (December 2005)

Annotation

Project is running, and what about now

The project has been scheduled in perfect fashion. We hold the best managers who supervise and monitor the project with respect to the given schedule. However, the problems and issues not covered in the schedule may arise and impose undesirable delay. What's our status? Is the planned completion of the project going to be put in jeopardy? In order to be able to answer preceding questions, the procedure for monitoring our project must be defined. One can extract suitable metrics to be capable to estimate the work that has been completed so far and to estimate what is still needed to be done. Then, the managers should be adapted for supervision and for the schedule as well. The question is: Are we going to monitor the project on-the-fly or are we going to let it flow by itself?