DOSTANEME LEN TO, ČO NAMERIAME

Ak rozoberieme systém na množstvo častí a následne tieto časti znovu poskladáme, nemusíme dostať pôvodný systém.

Vladimír Mamatej

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

Abstrakt. *V procese vývoja a tvorby softvérového produktu je potrebné mať neustály prehľad nad postupom projektu. Sledovaním jeho postupu zabezpečíme viditeľnosť projektu v každej fáze jeho vývoja. Aby sme toho dosiahli, potrebujeme sledovať kritériá, charakterizujúce atribúty softvérového projektu. Tie nám pomáhajú hodnotiť softvérový výrobok a proces, ktorý prebiehal pri jeho tvorbe. Avšak nástroje, ktoré používame pri monitorovaní projektu, nám nemusia vždy zaručiť prísun relevantných informácií o stave produktu. Získané dáta je potrebné pozorne študovať a hlavne si byť vedomý, čo v skutočnosti reprezentujú. Tieto nástroje sú preto len prostriedkom, ktorý nás môže doviesť k vytýčenému cieľu a je len na nás, ako ich využijeme. V tejto eseji vám predvediem, že je potrebné pochybovať o dátach získaných z jednotlivých metrík a tiež je opodstatnené nedôverovať samotným metrikám.*

Kľúčové slová: *počet riadkov textu programu (LOC), počet funkčných bodov (FP), metriky, postup v softvérovom projekte*

Motivácia pre meranie

Je jednoduché povedať, že niekto pracuje, ale môžeme tvrdiť, že robí pokrok, alebo napreduje za svojím cieľom ? Pri jednoduchej práci to vieme povedať, pretože každá ukončená časť práce je pre všetky zúčastnené osoby dobre viditeľná. Máme napríklad v rámci školenia vytvoriť triedu reprezentujúcu maticu a funkcie, ktoré s ňou budú

narábať. Jednotlivé „podúlohy“, ako je vytvorenie krátkych funkcií a dátovú triedu, sa dajú ukončiť veľmi rýchlo. Ale komplexná práca, ako je napríklad vytváranie softvérového produktu v rámci tímového projektu, sa skladá z veľkého množstva „podúloh“. Tie vyžadujú analýzu alebo rozhovory medzi ďalšími členmi tímu a tu nemôžeme dopredu vedieť, či naše úsilie vynakladáme efektívne a smerujeme ku cieľu. V zložitých úlohách alebo pri práci s väčším množstvom ľudí je ťažké určiť, kde sa začína práca a kde postup

Táto skutočnosť vyvoláva množstvo problémov, jedným s nich je aj zloženie tímu. Ako môžeme vedieť, či v našom tíme nie sú „prížívníci“ ? Pokiaľ nevieme rozlišovať medzi prácou a progresom, takýto typ ľudí si nájde v našom tíme vždy miesto.

V priebehu softvérového projektu vzniká množstvo neužitočnej práce. Schopní ľudia môžu začať pracovať na svojich osobných úlohách, veriac, že napredujú v projekte, pričom to v skutočnosti nemusí byť pravda. Ich práca môže byť „nad rámec“, lebo vytvárajú zdrojový kód, ktorý sa nemusí nikdy použiť a tým sa môže projekt dokonca oneskoriť. Nezáleží, do akej miery sa oddáme práci, pokiaľ tá napreduje nesprávnym smerom. Je dôležité usmerňovať ľudské úsilie a to je možné len v dobre riadenom tíme.

Je veľmi dôležité dať vedúcim tímu, alebo pri väčších projektoch projektovému manažérovi, do rúk nástroj, vďaka ktorému by bol schopný rozoznať, či niekto plytvá svojím časom. Vedúci tímu musí definovať ciele celého projektu, ale zároveň usmerňovať snaženie jednotlivcov do zmysluplného celku. Jednoduchým riešením na tieto problémy je začať monitorovať softvérový projekt. Základná myšlienka merania postupu v projekte je, že rozložením projektu na merateľné kritériá získavame nástroj, vďaka ktorému môžeme usmerňovať svoju činnosť. Ale občas to zlyháva: ľudia pracujú kvôli metrikám, nie cieľom.

Efektívne merania

Efektívne meranie (metrika), pomocou ktorého chceme sledovať progres, spĺňa nasledujúce charakteristiky [4] :

- Objektivita: meranie by malo byť postavené na kritériách, ktoré môžeme pozorovať a verifikovať. Výsledky objektívnych meraní je náročné zmanipulovať tak, aby sme videli realitu krajšiu ako je v skutočnosti.
- Blízke prítomnosti: meranie by predovšetkým malo odrážať to, čo sa deje teraz, nie to, čo sa stalo pred mesiacom. Radi by sme riadili a usmerňovali prítomnosť, nie minulosť.
- Granularita: viac úrovní dát nám umožňuje izolovať a sledovať konkrétny problém alebo problémovú oblasť, alebo vytvárať komplexné pohľady. Keď sa v priebehu softvérového projektu nestíhajú dodržiavať termíny, je dôležité vedieť, či sa oneskorenie bude týkať celého projektu alebo len konkrétnych činností.
- Predikcia: meranie by malo podporovať odhady do budúcna. Jednoducho povedané, je dobré vedieť, kde bude náš projekt o niekoľko týždňov alebo mesiacov.

Na sledovanie projektu sa najčastejšie používajú nasledujúce metriky [4]:

- Metriky rozsahu projektu: v tomto meraní porovnávame aktuálny postup oproti plánovanému, výsledok sa zaznamenáva ako podiel dokončených zložiek produktu k plánovaným v percentách.
- Metriky dokončených aktivít: v tomto meraní porovnávame aktuálny postup oproti plánovanému, výsledok sa zaznamenáva ako podiel dokončených aktivít k plánovaným v percentách.

Každý z týchto typov metrík ma svoje silné a slabé stránky, tiež záleží od ich implementácie.

Základom každého typu je porovnanie plánovaného progresu s aktuálnym. Jediným spôsobom, ako zistiť, že je náš projekt pozadu alebo popredu oproti plánovanému stavu, je sledovať jeho stav v pomere k očakávaniam, pričom očakávanie je náš plán.

Počet riadkov textu programu

Prvá metrika vyvinutá pre meranie softvérového projektu bola metrika nazvaná „source line of code" skratka SLOC. Táto technika sa zaoberá meraním počtu riadkov zdrojového kódu. Ak za výsledok práce programátora považujeme aj komentáre (hoci to nie je funkčný kód), použijeme variantu „line of code" alebo skrátene LOC.

Prvým problémom je definovanie pojmu *riadok textu programu*. Ak sa spýtate šiestich manažérov, ktorí pracujú v jednej budove, môžete dostať rôzne odpovede, pričom nameraná produktivita práce sa môže v niektorých prípadoch líšiť niekoľkonásobne.

Ďalším problémom môže byť v použití rôznych programovacích jazykov. Je rozdiel, ak programátor napíše 60 riadkov v strojovom jazyku a rovnaké množstvo riadkov v jazyku SQL. V súčasnosti sú tiež dostupné vývojárske prostredia s grafickým prostredím s veľkým počtom tlačidiel, rolovacích ponúk a iných ovládacích prvkov, ktoré umožňujú programátorovi vyvíjať softvér bez konvenčného písania. Komplikácie tiež spôsobuje používanie makroinštrukcií, knižníc tried, ale aj dedičnosti a iné formy znovu–použiteľného kódu.

Je zrejmé, že pri väčšom počte rôznych prístupov k meraniu riadkov kódu je vhodné používať štandard, ktorý by definoval, čo a akým spôsobom sa má počítať. Tu nastáva ďalší problém, pretože neexistuje jeden medzinárodný štandard, ktorý by obsahoval všetky programovacie jazyky. Namiesto toho poznáme niekoľko rôznych prístupov, ako napríklad udáva štandard SPR (Software Productivity Research) [2], ktoré sa však dostávajú do konfliktov. Rôzne spôsoby počítania pre jednu metriku nám dávajú rôzne výsledky.

Teraz sa pozrieme na hlavné výhody a nevýhody merania riadkov kódu, tak ako sa udávajú v literatúre [1] :

- LOC metrika je jednoduchá na výpočet,
- LOC metrika je metóda automatických výpočtov,
- LOC metrika je súčasťou nástrojov na vývoj softvéru.

Slabiny tejto metódy sú:

- LOC metrika môže zahŕňať prázdne riadky a komentáre,

- LOC metrika je mätúca pre projekty, ktoré používajú viac programovacích jazykov,
- Neexistuje konverzia medzi metódou LOC a metódou funkčných bodov.

Ďalšie nevýhody metódy LOC

Je dôležité uvedomiť si, že nie vždy môžeme predchádzať problémom použitím štandardov, nakoľko sa vyskytujú situácie, ku ktorým musíme pristupovať individuálne a subjektívne. Uvažujme nasledujúci jednoduchý príklad. Máme dvoch programátorov, pričom produktivitu práce meriame prostredníctvom metódy LOC. Jeden programátor za deň vyprodukuje 5 riadkov a druhý 200. Záver je podľa použitej techniky zrejmý. Ale čo ak je prvý programátor zručnejší a jednoducho zdrojový kód optimalizoval, pričom veľký počet riadkov zmazal. S použitím metódy LOC sme dosiahli práve opačné vyhodnotenie produktivity a to len preto, lebo táto uvedená metóda neráta s možnými okolnosťami. Tie môžu byť:

- Rôzne požiadavky na projekt: niektoré tímy sa môžu sústrediť na optimalizáciu bezpečnosti systému, čo môže priniesť nižší počet napísaných riadkov kódu oproti ostatným. Určite však nemôžeme hovoriť o zníženej produktivite práce,
- Neskúsený tím: Nový a neskúsený tím môže zo začiatku vykazovať takú istú produktivitu práce ako zabehnutý tím, avšak funkčnosť kódu je nižšia a počet chýb väčší,
- Rôzne skúsenosti členov tímu: Ak zadáme tú istú úlohu študentovi bakalárskeho stupňa vysokej školy, doktorandovi a programátorovi s 10 rokmi praxe, dostaneme rôzne výsledky, ktorých funkčnosť nemôžeme merať podľa dĺžky kódu.

Uvedené okolnosti skresľujú výsledky metódy LOC. Neexistuje spôsob, ako by sme ich zohľadnili pri automatických meraní.

Metóda funkčných bodov

Meranie funkčných bodov je objektívna technika monitorovania softvéru určená na meranie rozsahu softvéru, rozdelením funkcionality produktu na základe požiadaviek a vnútornej logiky. Technika rozkladá systém na menšie komponenty, ktoré vieme lepšie analyzovať a pochopiť.

Technika funkčných bodov vznikla vo vývojárskom tíme spoločnosti IBM, pod vedením Alana Albrechta. Prvotnou myšlienkou bolo vytvoriť metriku, ktorá nahradí LOC. Meranie funkčných bodov je preto nezávislé od použitého programovacieho jazyka alebo kombinácií jazykov, ktoré sa použili pre vývoj softvéru a tiež vstupné dáta poznáme už počas vývoja softvéru.

FP používa pri odhade veľkosti systému takzvané externé aspekty softvéru. Tie sa určujú na základe týchto piatich parametrov [1]:

- Počet logických vstupov,
- Počet výstupov,

- Počet dopytov,
- Počet logických dátových súborov,
- Počet rozhraní.

Prvé tri sa nazývajú transakčné funkčné typy a posledné dve dátové funkčne typy. Každému sa pridelí váha. Tieto váhy reprezentujú náročnosť implementácie príslušných bodov.

Technika funkčných bodov má niekoľko nevýhod. Sú to:

- Zložitosť výpočtu: výpočet tejto metriky prebieha manuálne, pretože údaje o váhach vstupujúce do výpočtov sú založené na subjektívnom úsudku. Tu je veľmi dôležité pracovať so správnou granularitou, najviac by sme mali použiť 5 až 7 úrovní váh; v opačnom prípade môžeme skĺznuť do „triafania sa" správnej hodnoty.
- Vyžaduje veľkú úroveň detailov: ohľadom funkčných bodov je nevyhnutné detailne sa venovať odhadu veľkosti softvéru.
- Vyžaduje skúsenosti: Aby sme túto metódu precízne zvládli, potrebujeme skúsenosti s projektovaním. Pravidlá pre výpočty a nastavenie funkčných bodov sú pomerne zložité. Vyžadujú tréning a je žiaduce splniť certifikačné skúšky skôr, ako použijeme túto metódu.

Nedôverujte meraniam

Meranie rozsahu softvéru prináša dôležité informácie pre náš projekt. Získané údaje sú podstatné nielen pre odhady produktivity práce, ale aj pre odhady ceny nákladov, zdrojov projektu a prípadne riadenie zmien.

Uviedli sme si dve základné metódy pre tento druh merania. Prvá metóda LOC má väčší počet nedostatkov ako pozitívnych vlastností. Táto skutočnosť však neprekáža, pokiaľ si ich uvedomujeme a sme s nimi oboznámení. Metódu LOC používame ako automatickú metódu merania. To ale neznamená, že nebudeme získané dáta analyzovať.

Ak rozoberieme systém na množstvo častí a následne tieto časti znovu poskladáme, nemusíme dostať pôvodný systém. Pri zbieraní štatistických údajov sa strácajú súvislosti. Ukázali sme si to v časti „nevýhody metódy LOC". Strata relácií medzi jednotlivými údajmi môže vyvrcholiť niektorými nepríjemnými situáciami v tímovom projekte. Jedna z nich je syndróm 90%. Tento syndróm hovorí o tom, že na posledných 10% projektu musíme vynaložiť 90% celkového úsilia. Tomuto známemu paradoxu nevieme predchádzať metódou LOC a to z toho dôvodu, že táto metrika sa sústreďuje na meranie prítomnosti. Meriame aktuálne vynaložené úsilie, na základe ktorého vieme určiť produktivitu a odhadnúť, koľko úsilia ešte potrebujeme vynaložiť (v obmedzenej miere). To však neidentifikuje paradox 90% , dokonca údaje získané z tejto metódy nás môžu utvrdzovať v opaku.

Druhú metódu, ktorú sme si načrtli, je metóda funkčných bodov. Táto metóda je podstatne komplexnejšia a odstraňuje niektoré nevýhody metódy LOC. Bola navrhnutá tak, aby vstupné dáta pre ňu boli známe už na začiatku projektu. To nám umožňuje zvýšiť presnosť odhadov, v niektorých prípadov až na úroveň plus mínus 10% [1]. Meranie ale môže byť časovo náročné a drahé a je závislé od našich skúseností.

Jednoduchá pomoc

Jedným zo spôsobov, ako zvýšiť výpovednú hodnotu používaných metód, je komunikácia v tíme. Tejto úlohy by sa mal zhostiť manažér, ktorý je ochotný navštevovať členov tímu a klásť nasledovné otázky [1]:

- Ku ktorému cieľu smeruje táto činnosť ?
- Ako nám to pomôže dostať sa bližšie k ukončeniu činnosti ?
- Akou veľkou mierou nám to pomôže (5% alebo 50%)?
- Bude kvalita dostatočne vysoká ?
- Potrebujeme dodatočnú podpornú činnosť ?

Správne položené otázky môžu vyjasniť niektoré okolnosti týkajúce sa výkyvov produktivity zistených pomocou automatických metód. Môžu nám pomôcť pri odhalení skrytých problémov opísaných v stati „ďalšie problémy metódy LOC“. Aj takouto jednoduchou cestou sa dá zlepšiť presnosť meraní.

Ďalšia možnosť spočíva v kombinácií väčšieho počtu metrík. Tiež existujú metriky, ktoré sa snažia kombinovať výhody viacerých meraní. Takouto metódou je napríklad Backfiring [1].

Záver

Predstavil som dve najpoužívanejšie metódy na meranie rozsahu softvérového projektu, pričom dôraz som kládol najmä na ich nevýhody a nedostatky. Je dôležité uvedomiť si, že použitím hocijakej metódy na meranie dostávame do rúk len nástroj. Zameraním na ich chyby a poukázaním na situácie, v ktorých zlyhávajú, som chcel vniesť neistotu do ich používania. Získané dáta je potrebné vždy analyzovať a pochybovať o nich.

Sedemnásobný víťaz Tour de France Lance Armstrong nazval svoju autobiografiu: „Nie je to o bicykli: môj návrat do života“ (*It's Not About the Bike: My Journey Back to Life*). Hoci strávil nespočetné množstvo hodín na bicykli, bol to pre neho len nástroj pre boj s rakovinou. Meranie, podobne, nie je koncom, je len prostriedkom na zdôraznenie aktivít a produktov, ktoré my a náš projektový tím môžeme dosiahnuť. Ale je to iba nástroj. Aby sme sa niekam dostali, musíme správnym smerom navigovať na ceste – musíme sa rozhodovať a konať [3].

Použitá literatúra

1. Berkun, S.: *Work vs. Progress*. Dostupné na internete: http://www.scottberkun.com/essays/45-work-vs-progress, [cit: 2009-Október]
2. Capes, J.: *Strengths and weaknesses of software metrics*. Dostupné na internete: http://www.compaid.com/caiinternet/ezine/capers-StrngWk.pdf, [cit: 2009-Október]
3. Clark, B.: *Eigth Secrets of Software Measurement*. Dostupné na internete: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1032844, [cit: 2009-Október]

4. International Function Point Users Group: *How to Effectively Track Software Progress*. Dostupné na internete: http://www.informit.com/articles/article.aspx?p=27356, [cit: 2009-Október]

Annotation

We only can get what we measure

In the process of developing and creating a software product, it is necessary to have the overall survey over the progress of the project. Watching the progress, we can reach the visibility of the project in every phase of its development. In order to arrive at this, we have to follow the characteristic attributes of the software project. They help us evaluate the software product and the process that was a part of the project. But the tools we use for monitoring do not always and necessarily bring the information needed concerning the state of the project. It is necessary to study carefully the data acquired and bear in mind what they actually stand for. These tools are hence only a means that may lead us to our goal. It is upon us how we will use these tools. In this essay, I will prove that it is necessary to doubt the data acquired from separate measurements and that it is essential not to trust measurements themselves.