Monitorovanie softvérového projektu a vplyv na plánovanie a riadenie

MICHAL LOKŠA

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
loksa00@student.fiit.stuba.sk

Abstrakt. Softvérový projekt bez správneho plánovania a riadenia bude skôr či neskôr odsúdený k neúspechu. Preto je potrebné toto riadenie zabezpečiť a to nie len skúseným manažérom softvérového projektu, ale aj monitorovaním (sledovaním) celého projektu. Toto sledovanie postupu projektu môže manažérov upozorniť na prichádzajúce problémy a umožniť im tak prijať včas potrebné opatrenia. Esej vysvetľuje pojem monitoring softvérového projektu. Zaoberá sa jednotlivými princípmi monitorovania a vysvetľuje ich vplyv na plán a riadenie projektu.

Úvod

Softvérový projekt bez správneho plánovania a riadenia bude skôr či neskôr odsúdený k neúspechu. Preto je potrebné toto riadenie zabezpečiť a to nie len skúseným manažérom softvérového projektu, ale aj monitorovaním celého projektu. Toto sledovanie postupu projektu môže manažérov upozorniť na prichádzajúce problémy a umožniť im tak prijať včas potrebné opatrenia.

Cieľ monitoringu

Sledovanie projektu je kritická aktivita, ktorá zaručuje úspech softvérového projektu. Projektový manažér musí robiť svoje rozhodnutie smerom, ktorý je podložený merateľnými dátami.

Na začiatku projektu sú rozdelené jednotlivé pozície alebo dá sa povedať roly v jednotlivých úlohách tým ľuďom, ktorých má manažér k dispozícii. Počas projektu môže prísť k nejakým zmenám alebo práve monitorovanie projektu odhalí, že na nejakú úlohu treba nasadiť viacerých ľudí alebo naopak, pracuje na nej veľký počet ľudí. Je dosť pravdepodobné, že aj keď bude plán projektu vypracovaný do úplných podrobností a na vysokej úrovni, vzniknú odchýlky od plánu, poprípade bude potrebné

Manažment v softvérovom inžinierstve, október 2006, s. 1-6.

ho v určitom smere prerobiť alebo mierne pozmeniť. Projektový manažér nikdy nemôže dopredu vedieť, čo sa počas projektu zmení a aké možné prípady nastanú.

Proste monitorovanie by malo slúžiť aj na vyťaženie čo najväčšieho výkonu z daného tímu, ktorý pracuje na projekte.

Zaujímavý je názor uvedený v [1], že vo viacerých oblastiach riadime veci lepšie prostredníctvom čísel. Čiže ak projektový manažér má vo svojej pozícii zistiť aktuálny stav projektu, má ako jeden veľmi silný nástroj práve monitoring.

Objekty monitoringu

Ak chceme začať monitorovať softvérový projekt, najdôležitejšou úlohou je správne určiť stav, v ktorom sa projekt aktuálne nachádza. Projektový manažér by sa nemal snažiť zisťovať stav projektu tak, že sa bude pýtať podriadených ako sú na tom so svojou prácou. Ich vyjadrenia o vlastnej činnosti budú veľmi nepresné a zavádzajúce. Projektový manažér musí zistiť reálne čo je na projekte dokončené, podľa jednotlivých naplánovaných úloh a stavu v akom tieto úlohy sú. Následne porovná tieto údaje s tým, čo bolo plánované, a aké boli očakávania stavu projektu podľa plánu. Dá sa povedať, že projektový manažér porovná plán a stav projektu podľa jednotlivých úloh a porovná ich. Informácie o projekte by mal získať na dvoch úrovniach. Na úrovni jednotlivých úloh a následne potom na úrovni celého projektu.

Zistenie aktuálneho stavu projektu je hlavná časť monitorovania projektu. Projektový manažér zistí v akom stave sa projekt nachádza oproti pôvodnému plánu. Stav projektu je ale potrebné sledovať z viacerých pohľadov alebo podľa viacerých kritérií. Ako prvé kritérium by mohol byť práve čas, ktorý bol spotrebovaný od začiatku projektu. Čas je veľmi často dosť veľkým obmedzením pre projekt a tiež pre všetkých čo spolupracujú na projekte. Pri sledovaní projektu je nevyhnutné zistiť ako dlho jednotlivé fázy trvali, a tiež odhadnúť koľko ešte času zostáva do konca projektu. Určite by nebolo dobré, ak by projektový manažér zistil po uplynutí doby určenej pre projekt, že je hotová sotva tretina úloh, ktoré sa mali byť splnené.

Ďalším významným objektom na sledovanie sú náklady na projekt a na jednotlivé úlohy. Náklady na softvérový projekt vychádzajú väčšinou z odhadu, ktorý je potrebné vykonať na začiatku projektu. Práve na skúsenostiach a schopnostiach manažéra závisí ako veľmi sa tento odhad priblíži ku konečným skutočným nákladom. Dalo by sa povedať, že čím viac sa náklady na projekt odlišujú od odhadu, tým významnejšie vplývajú na úspešnosť projektu. Teda sledovanie nákladov je veľmi dôležité a projekt, v ktorom sa počas jeho trvania spotrebuje dvojnásobok pôvodných nákladov, bude možno úspešne ukončený načas, ale je otázne či to nebude posledný projekt pre takéhoto projektového manažéra [2].

V softvérových projektoch, či už malých alebo väčších, sa vytvára množstvo zdrojového kódu. Tento kód musí spĺňať určité atribúty. V opačnom prípade hrozí, že sa z neho stane len veľké množstvo neprehľadného a nič nehovoriaceho textu. Medzi najsledovanejšie atribúty patrí čitateľnosť, prehľadnosť, znovupoužiteľnosť, výkonnosť, chybovosť a iné.

Ak teda projektový manažér hľadá odpoveď na otázku: Čo je objektom monitoringu pri sledovaní softvérového projektu? Musí vychádzať zo svojich skúseností a vedomostí a zvoliť si pre neho potrebnú a zaujímavú skupinu parametrov. Tieto zvolené parametre musí sledovať do takej miery, aby mu pomohli získať prehľad o stave a budúcom vývoji softvérového projektu.

Monitorovanie kódu

V malom, alebo strednom projekte je potrebné monitorovať zdroje, plnenie plánov, ale najviac pozornosti si zaslúži vytváraný kód. A to hlavne preto, že na projekte sa nepracuje jeden deň, ale sú to rádovo stovky dní. Tým pádom je možné, že z projektu odstúpia niektorí ľudia a tak isto ich pár môže aj pribudnúť. Je preto dôležité aby noví ľudia nemali problém začať pracovať už na rozrobenom projekte. Hlavné problémy nastávajú pri čítaní existujúceho kódu. Niektorí programátori patria medzi inteligentnejších ale zároveň aj lenivejších. Tým pádom môžu napísať kód, ktorý nie je veľmi čitateľný.

Ako príklad si zoberme jazyk C/C++, ktorý obsahuje množstvo operátorov, ktorých kombináciou môžeme v jednom riadku programu dosiahnuť vysokú funkcionalitu, ale na druhej strane minimálnu čitateľnosť. Potom sa môže stať, že náš programátor napíše výraz, ktorý plní na 100% svoju funkciu, ale je napísaný tak nezrozumiteľne, že sám autor už po pár dňoch azda len tuší, na čo daný výraz slúži, ale nevie ako funguje. Predpokladajme že v okolí tohto výrazu sa hľadá chyba. Čo sa stane ak tento výraz objaví náš nováčik? Zamyslí sa, ale je malá pravdepodobnosť, že výraz pochopí. Preto bude výraz refaktorovať a určite ho prepíše na väčšiu postupnosť krokov. Či ho už bude prepisovať, alebo nie, tak sa v danom momente plytvalo drahocenným časom. Preto treba zabezpečiť, aby programátori tvorili čitateľný kód, ktorý bude menej náchylný na chyby. Na druhej strane sa tým môže v niektorých prípadoch spomaliť výkon programu, preto ak je potrebný rýchly kód, môžu sa použiť aj komplikovanejšie zápisy výrazov, ktorých vykonanie je väčšinou rýchlejšie. V tom prípade sa musia jednotlivé sekcie viac komentovať.

Vychádzajme z toho, že aj keď náš projekt nebude tvorený zložitými konštrukciami, bude musieť byť komentovaný. Nemyslí sa tým len komentovanie hlavičiek funkcií. Napr. v novobudovanom európskom navigačnom projekte Galileo musia programátori dodržiavať viac štandardov ako je bežné pri iných projektoch. Dôvod je ten, že na tomto projekte pracuje niekoľko sto ľudí a musia sa minimalizovať všetky druhy chýb. Je vylúčené, aby v projekte s takým veľkým rozpočtom boli prehliadnuté niektoré chyby, ktoré by v neskoršom spúšťaní mohli spôsobiť pád systému a to už po stránke softvérovej ako i hardvérovej. Preto programátori v tomto projekte napíšu v priemere až 4 riadky komentáru na 1 riadok kódu.

Náš projekt (malý alebo stredný) nie je takto zameraný, preto si vystačíme s menším počtom komentárov. V praxi sú to iba percentá z celkového počtu riadkov kódu. Metrika ktorá sa zaoberá komentármi sa volá Percentuálny podiel komentárov (COM, z angl. Comment percentage). Hodnota tejto metriky sa vypočíta ako percentuálny podiel počtu riadkov obsahujúcich komentár k celkovému počtu riadkov

kódu (okrem prázdnych riadkov). Výsledok odráža mieru pochopiteľnosti, udržateľnosti a znovupoužiteľnosti skúmaného programu.

Dobrú čitateľnosť a prehľadnosť kódu predurčuje aj používanie tzv. štábnej kultúry, ktorá definuje jednotné pravidlá pre úpravu zdrojového kódu. Jedná sa tu hlavne o štruktúrované odsadzovanie riadkov pomocou tabulátorov alebo medzier. Taktiež o výstižné pomenovanie premenných, metód a aj tried. Niektoré editory zdrojových kódov sami automaticky odsadzujú zdrojový kód. Tým odbremeňujú programátora od rutiny a zvyšujú čitateľnosť kódu. Tak isto existujú programy, ktoré vedia nanovo naformátovať zle formátovaný kód.

Keďže na projekte pracujú ľudia, je jasné, že v tíme nenájdeme dvoch identických ľudí. Vývojárov môžeme zadeliť do viacerých skupín. Pre projekt je zaujímavé delenie do skupín podľa výkonnosti. Pretože ľudia môžu na projekte stráviť viac, či menej času, vyprodukovať viac, či menej riadkov kódu a tým pádom viac, či menej chýb. Ideálne by bolo, ak by programátor venoval projektu 100 % času stráveného v práci, vyprodukoval čo najviac riadkov kódu a vytvoril čo najmenej chýb. Kombináciou týchto atribútov by sa dali zoradiť členovia tímu podľa výkonnosti. Monitoring výkonnosti napomôže projektovému manažérovi v rozhodovaní a upravovaní plánu, pretože vie čo môže od svojich ľudí očakávať. Tak isto sa dá pomocou týchto alebo podobných podrobnejších metrík zistiť, či niektorý člen tímu ovláda potrebné technológie, resp. či sa im nesnaží vyhýbať. Vtedy po vzájomnej dohode je najvhodnejšie zaškoliť člena tímu.

Mnoho vývojových prostredí už dnes podporuje meranie zdrojového kódu. Používateľ si zvolí pre neho zaujímavé metriky, nastaví časový interval ako často sa má meranie spúšťať. Výsledky meraní môžu byť ľubovoľne spracované a zobrazované (napr. mail, web). K podstatným metrikám, ktoré majú veľký význam patrí meranie duplicity kódu. Ak je časť kódu veľakrát opakovaná, tak je kód náchylnejší na chyby. Dôvod je ten, že ak sa v jednom bloku našla chyba a táto chyba by mala byť odstránená aj v duplicitných kódoch, tak je neefektívne prehľadávať celý zdrojový kód a vyhľadávať dané duplicity. Skôr či neskôr sa určite na niečo zabudne a bude sa hľadať predchádzajúca chyba odznova. Pri odstránení duplicity takéto šírenie chýb nehrozí. Navyše sa zlepší čitateľnosť programu, keďže duplicitný kód sa nahradí volaním funkcie s výstižným názvom.

Na zdrojový kód sa môžeme pozrieť aj z pohľadu štýlu programovania [3]. Samotný štýl sa nedá merať priamo, ale dá sa vypočítať ako súčet vážených charakteristík programu. Každá charakteristika má pridelené určité bodové rozpätie, pričom čím viac bodov, tým je lepší výsledok merania. Ako príklad si zoberme jazyk C, kde sa sledujú tieto charakteristiky: *Priemerná dĺžka modulu*, *priemerná dĺžka identifikátora*, *percento komentovaných riadkov*, *odsadenie textu*, *prázdne riadky*, *definície konštánt* a iné. Tieto charakteristiky sú rôzne váhované. Zaujímavosťou je, že charakteristika *Počet GoTo príkazov* má záporné hodnotenia. Čiže výrazne zhoršuje celkové hodnotenie programovacieho štýlu.

Je už na manažérovi, ako nastaví parametre sledovaných metrík. V podstate má viac možností. Buď budú metriky nastavené prísnejšie, t.j. budú v prípade, že projekt sa uberá zlým smerom varovať skôr a tým pádom nútiť vývojárov priebežne

vylepšovať zdrojový kód. Naopak, ak budú metriky nastavené benevolentne, budú nútení vývojári menej často vylepšovať zdrojový kód. Tým pádom to bude trvať dlhšie, zastaví sa celkový postup na pár dní a zmení sa aj plán projektu. Táto možnosť má viac záporov ako kladov. Pretože hromadenie „chýb“ meranými metrikami nenarastá lineárne s časom, ale exponenciálne. V tom prípade je nutné po dlhšom čase urobiť v zdrojových kódoch rozsiahlejšie úpravy. Hlavné nebezpečenstvo tkvie v tom, že ak napr. dva tímy robia nezávislú prácu a v určitom čase by sa mali ich produkty spojiť, tak ak jeden tím prestane stíhať plán, tak ovplyvňuje chod druhého tímu.

Keď cítime, že sa už do softvérového projektu nedá pridať ďalšia funkcionalita, tak je potrebné začať s refaktoringom kódu. Začiatkom refaktoringu je zavedenie automatických testov. V zásade sa refaktoruje po malých krokoch, pretože pri nesprávnom zásahu je jednoduché krok napraviť. Podľa Kenta Becka sa tu uplatňuje princíp dvoch klobúkov. V jednom pridávame funkcionalitu a nemeníme štruktúru, v druhom meníme štruktúru a nepridávame žiadnu funkcionalitu. Môžeme ich meniť aj behom piatich minút pri novej funkcii, ale vždy musíme vedieť, čo robíme (aký klobúk máme na hlave) a nemiešať pridávanie funkcionality s refaktoringom. [2]

Záver

Ak projektový manažér monitoruje softvérový projekt, musí zistiť stav v ktorom sa projekt aktuálne nachádza. Sledovať úroveň plnenia jednotlivých úloh, sledovať mieru nákladov a tieto údaje porovnávať s plánovanými hodnotami. Na základe týchto porovnaní musí projektový manažér upravovať stratégiu riadenia softvérového projektu.

V monitoringu zdrojových kódov musí projektový manažér nastaviť parametre sledovaných metrík tak, aby k úprave zdrojových kódov nedochádzalo príliš často (nízke kritické hodnoty sledovaných metrík) a naopak, aby sa úpravy nevykonávali až vtedy keď bude neskoro (vysoké kritické hodnoty sledovaných metrík).

Dôležité je však pripomenúť, že projekt by sa v žiadnom prípade nemal zmeniť na preteky v sledovaní. Aj tu platí, že nakoniec všetkého veľa škodí.

Použitá literatúra

1. Michael C. Mah, Lawrence H. Putnam: *Software by the numbers*. Dostupné na http://www.qsm.com/aerialview.html (Október 2006)
2. Software Project Tracking and Oversight process. Dostupné na http://sepo.spawar.navy.mil/SPTO.doc (Október 2006)
3. Bieliková, M.: *Softvérové inžinierstvo*. Princípy a manažment. Vydavateľstvo STU, Bratislava, 2000
4. Polášek, I.: *Refaktoring* (poznámky k prednáške), Dostupné na http://www.gratex.com/Download/OOANS04Refact.pdf (Október 2006)

Annotation

Monitoring of the software project and influence on planning and management

Software project without proper planning and management will be sooner or later doomed to failure. And that is why this management is necessary to assign and not only by experienced manager of software project, but also by monitoring of the whole project. This monitoring of the project's procedure can draw manager's attention to coming problems and hereby enable them to take a proper measures soon enough. This essay explains the concept of monitoring of the software project. It deals with individual monitoring principles and explain their influance on planning and management of the project.