NUTNÉ METÓDY MONITOROVANIA A PROBLÉMY S NIMI SPOJENÉ

Dôveruj, ale preveruj! Je dobré veriť svojmu tímu, že projekt dotiahne až do cieľa ale bez poriadnej kontroly sa to podariť nemusí.

Juraj Mäsiar

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

Abstrakt. *Monitorovanie projektu sa ukázalo ako nevyhnutné pri vývoji každého väčšieho softvérového projektu, ktorý plánuje byť úspešne ukončený v dohodnutom termíne. Existuje však množstvo typov softvérových projektov, ktoré sa líšia svojim rozsahom, veľkosťou vývojového tímu, časovou zložitosťou atď. Aby bolo monitorovanie účinné a v konečnom dôsledku splnilo svoj cieľ, je nutné ho prispôsobiť špecifickým vlastnostiam projektu. Uvažovať teda o nejakom najlepšom spôsobe monitorovania aplikovateľnom na rôzne projekty je zbytočné. Čo ak by ale bolo možné definovať základné pravidlá, ktoré by sa dali použiť pri riešení rôznych typov projektov. V eseji sa pokúsim definovať takéto nevyhnutné pravidlá pre úspešné ukončenie softvérového projektu na základe niektorých používaných postupov monitorovania. Taktiež sa budem venovať problémom spojeným s monitorovaním a navrhnem ich možné riešenia.*

Kľúčové slová: *monitorovanie, odhad, sledovanie, progres, postup, problémy, riešenie, projekt, softvér*

Na čo vlastne potrebujeme monitorovanie?

Skúsme sa na začiatok samých seba opýtať otázku „Monitorujem úlohy s ktorými sa každý deň stretávam?“. Po dlhšom zamyslení človek zistí, že aj napriek tomu, že cez deň

nevytvorí ani jeden *Gantt diagram*, takmer všetky svoje činnosti vedome či podvedome monitoruje. Väčšinou stačia jedny rozvarené cestoviny aby si človek uvedomil, že mal priebežne kontrolovať postup varenia. Aj pri učení na skúšku človek priebežne kontroluje, koľko toho musí ešte prebrať. Niektoré bežne vykonávané činnosti za nás monitoruje technika, napríklad rýchlovarná kanvica kontroluje či už voda nevrie.

Podobne to funguje aj pri riešení softvérových projektov. Rozdiel je hlavne vo veľkosti a čase riešenia, kedy človek stráca prehľad o celom projekte a na jeho monitorovanie potrebuje rôzne monitorovacie nástroje a techniky. Tieto nám pomôžu odhaliť časované bomby v projekte, odhaliť odbočenie od cieľa projektu, prípadne aj zastaviť projekt, v ktorom už nemá zmysel pokračovať.

V minulosti bolo monitorovanie často podceňované a riadenie projektu sa spoliehalo len na vytvorený plán. Ten však úspech zaručiť nedokáže, pretože s každou neplánovanou udalosťou je nutné ho meniť. Avšak pravdou ostáva, že monitorovanie je od plánu v určitej miere závislé. Ak si na začiatku nezvolíme konzistentný plán s jednoducho sledovateľnými kontrolnými bodmi, vzniká problém s monitorovaním.

Nutné minimá

Máme projekt a chceme aby bol úspešne ukončený. Na čo všetko treba myslieť pri plánovaní monitorovania? Pokúsim sa navrhnúť nejaké „nutné minimum“, ktoré bude schopné pokryť väčšinu nástrah spojených s vývojom softvéru. Úspešná stratégia by mohla pozostávať z nasledujúcich krokov:

1. Rozhodnúť, ako často bude prebiehať monitorovanie. Ak uvažujeme metódu vývoja SCRUM je potrebné sledovať postup v projekte minimálne po každom šprinte. Zo skúsenosti pri práci na tímovom projekte viem, že sporadická kontrola môže ľahko vyústiť do nedodržania stanovených termínov. Avšak teória, že monitorovanie po každom šprinte bude dostatočné môže v praxi naraziť na zopár prekážok, ktoré budem neskôr popisovať.
2. Ujasniť si, čo má byť cieľom monitorovania. Minimálne treba mať aktuálne informácie o prebiehajúcich úlohách. Na tento účel sa výborne hodí *Gantt diagram*, ktorý prostredníctvom prehľadného diagramu zobrazí priebeh naplánovaných úloh. Aj krátky pohľad naň prezradí, či sú úlohy plnené načas alebo nie. Vďaka tomu vieme odhadnúť aj nasledujúci vývoj vychádzajúc z frázy *„Ak budeme naďalej pracovať takýmto tempom...“*. Vytváranie *Gantt diagramu* nie je náročné a podieľa sa na ňom každý člen tímu, ktorý má pridelenú aspoň jednu úlohu, preto by mal patriť medzi základné monitorovacie techniky v každom projekte.
3. Ak potrebujeme zohľadniť aj vzťahy medzi úlohami, prípadne presnejšie odhadnúť dĺžku trvania projektu, môžeme použiť *Sieťový diagram*. Tento diagram umožňuje zadefinovať vzťahy medzi úlohami, takže môžeme určiť, že niektorá úloha sa môže začať vykonávať až po skončení inej. Z tohto diagramu sa dá získať *kritická cesta*, ktorá nám pomôže určiť dĺžku trvania celého projektu a umožní identifikovať tie úlohy, ktorých oneskorenie priamo predĺži celý vývoj (takým úlohám je nutné venovať zvýšenú pozornosť).

4. Ak však chceme vedieť čo sa ukrýva pod jednotlivými úlohami, musíme siahnuť po ďalších dátach. Ak chceme napríklad sledovať tvorbu dokumentácie, väčšinou nám ako metrika môže stačiť počet strán, prípadne počet riadkov či slov. V prípade sledovania vývoja programu, máme k dispozícii celý rad iných metrík, každá z nich so svojimi výhodami aj nevýhodami. Niekedy totiž napísanie 10-riadkovej funkcie môže byť náročnejšie ako napísanie stoviek riadkov iného kódu (často napríklad pri vytváraní funkcie, ktorú volá viac programových vlákien súčasne). Známym pravidlom je aj to, že pri konštantnej práci na programe nerastie počet riadkov kódu lineárne, ale stále pomalšie [3]. Je to pochopiteľné, nakoľko sa na začiatku programovania vytvára veľké množstvo tried, funkcií a štruktúr. V neskoršej fáze programovania sa často už len pracuje s existujúcimi triedami prípadne sa opravujú chyby.
5. Čo v prípade, že chceme monitorovať aj minuté prostriedky a analyzovať budúce výdaje aby nedošlo k prekročeniu rozpočtu? Ponúka sa medzinárodne známa a používaná [2] metóda *analýzy vytvorenej hodnoty* (anglická skratka EVA – earned value analysis). Ide o komplexnú metódu analýzy vývoja projektu.
6. Vybrať tie dáta, ktoré chceme spracovávať v procese monitorovania. V podstate sa stačí držať vytýčených cieľov monitorovania a tomu prispôsobiť údaje, ktoré budeme analyzovať.

Problémy s monitorovaním, ako ich riešiť?

Proces monitorovania sprevádza množstvo nástrah, ktoré ovplyvňujú jeho výslednú kvalitu. Už niekoľko rokov pozorujem, že s blížiacim sa termínom stúpa výkon práce. Je pravda, že toto nemožno automaticky aplikovať na ľubovoľný vývoj softvéru, je totiž iné keď človek pracuje vo firme kde má každý deň pridelený čas na riešenie úlohy a iné, keď si má človek tých pár hodín nájsť vo svojom voľnom čase. Navyše úvahy typu "Mal by som za týždeň na projekte/úlohe pracovať dokopy 7 hodín, takže ideálne hodinu denne" nefungujú takto pekne aj v praxi. Z vlastnej skúsenosti z tímového projektu viem, že riešenie aj menších problémov často pozostáva z niekoľkých časovo nezanedbateľných činností. Napríklad pridanie novej funkcionality do existujúceho projektu (povedzme že práca na 7 hodín) vyžaduje pri každom začatí riešenia minimálne dva kroky. Prvým je príprava nástrojov, materiálov, podkladov a iných pomôcok nutných pri riešení (niekedy aj 15 minút). Druhým je zorientovanie sa v projekte - fungovanie aj vlastnoručne napísaných zložitejších funkcií môže z hlavy „vyprchať“ behom jedného dňa. Plné vnorenie sa do projektu môže trvať pár minút ale aj pol hodinu. Z uvedeného vyplýva, že časová fragmentácia riešenia problému na malé časti nie je vhodná nakoľko spôsobuje viacnásobné vykonávanie tých istých krokov. Ak chce teda človek čo najefektívnejšie využiť 7 hodín, ktoré má stráviť na projekte, musí si zo svojho voľného času vyčleniť súvislý 7 hodinový úsek. Keď sa toto skombinuje s prirodzenou vlastnosťou ľudí odkladať veci na neskôr, dostaneme spomínaný postreh, že väčšina práce sa vykoná krátko pred termínom. Navyše v snahe stihnúť dokončiť úlohu človek často nehľadí na kvalitu svojej práce. Pri programovaní sa stráca typicky čitateľnosť kódu, bezpečnosť, možnosť budúceho použitia atď. Niekedy sa stane, že človek vôbec nestihne dokončiť to, čo by mal,

pretože nepočíta s nečakanými situáciami. Tie môžu vyžadovať viac času alebo pomoc inej osoby, ktorá nemusí byť v danej chvíli zastihnuteľná. Pri riešení softvérového projektu sa všetko to čo sa nestihlo automaticky prenesie do ďalšieho týždňa spolu s novými úlohami (ak sa postupuje podľa plánu), no môže byť toho až príliš a situácia sa opakuje aj ďalšie týždne. Riešenie je nenechávať si veci na poslednú chvíľu. Tak jednoduché ako zložité býva jeho aplikovanie v reálnom živote.

Ďalším faktorom prispievajúcim k vzniku problémov je prirodzená vlastnosť človeka podávať informácie o svojej práci v pozitívnom duchu [1]. Typickým príkladom môže byť prehlásenie takmer hotovej úlohy za hotovú. Pritom sa nejedná o zavádzanie alebo klamanie, človek môže mať pocit, že úloha je takmer hotová a vyžaduje si už len málo práce. V skutočnosti sa môže uplatniť *Paretov princíp* a dokončenie posledných 20% percent zaberie až 80% celkového času. Takéto úniky informácií sa len veľmi ťažko hľadajú a preto je lepšia prevencia, ktorá sa dá zabezpečiť iba správnou výchovou tímu. Navrhoval by som systém odmien za pravdivé (aj keď často nelichotivé) správy a systém trestov za prikrášlené správy. Je to práve chýbajúca motivácia čo ľuďom chýba.

Druhým (možno ešte väčším) problémom môže byť prirodzená vlastnosť človeka vyjadrovať svoje názory v každej oblasti vrátane tých, v ktorých nemá prehľad. Výsledkom potom môžu byť nepresné odhady a zlé rozhodnutia. Pritom riešením je jednoduché priznanie svojej nevedomosti (niečo nevedieť nie je hanbou!).

Záver

V tejto eseji som sa snažil obohatiť známe fakty o monitorovaní o vlastné myšlienky a názory. Spomenul som niekoľko možných prístupov použiteľných v rôznych typoch projektov. Monitorovanie je náročný proces zväčša založený na subjektívnom hodnotení. Jeho kvalita vo veľkej miere závisí od jednotlivých členov tímu čo prináša množstvo problémov spojených zo získavaním skutočných údajov.

Použitá literatúra

1. Fenton, N., et al. Making resource decisions for software projects. In *Proceedings of the 26th International Conference on Software Engineering ICSE'04, 2004.*
2. Jinhua L., Zhibing M., Huanzhen D.: *Monitoring software projects with earned value analysis and use case point*. Seventh IEEE/ACIS International Conference on Computer and Information Science, 2008, 475-480.
3. Král, J. a Demner, J.: *Softwarové inženýrství*. Academia, Praha, 1991.

Annotation

Necessary methods of software monitoring and problems comings with them

Project monitoring should be important and integral part of each project development. Any project development cannot end successfully without it. But on the other hand there are many problems that have influence on monitoring. If we solve those problems, we can make monitoring much

better. This paper discusses different monitoring methods, features and problems. Into this work I have included my suggestions of solving some monitoring problems.