Sledovanie postupu softvérového projektu a manažment

MICHAL JEMALA

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jemala01@student.fiit.stuba.sk

Abstrakt. Mnoho softvérových projektov sa končí odlišne, ako bolo pôvodne naplánované. Bežné je oneskorenie, prekročenie rozpočtu, prípadne nesplnenie požadovanej funkcionality. Sledovanie postupu projektu, môže manažérov upozorniť na prichádzajúce problémy a umožniť im tak prijať včas potrebné opatrenia. Kľúčovým faktorom pri sledovaní postupu projektu je zabezpečiť jeho viditeľnosť a to v ľubovoľnom okamihu riešenia projektu, v najlepšom prípade kontinuálne a automatizovane.

Úvod

Sledovanie postupu softvérové projektu predstavuje integrálnu súčasť procesu referovania o výkone, ktorý predstavuje subproces riadenia softvérového projektu. Pravidelné sledovanie a vyhodnocovanie postupu projektu predstavuje veľmi dôležitý faktor z hľadiska úspešnosti projektu. Niektoré štúdie dokonca uvádzajú, že softvérové firmy v USA premrhali v roku 1995 až 59 mld. dolárov len kvôli predĺženiu projektu a 81 mld. dolárov kvôli úplnému zrušeniu projektu [2]. Je zrejmé, že včasné odhalenie príčin umožní projektovým manažérom prijať potrebné opatrenia, ktoré zmiernia resp. úplne odstránia ich negatívne dôsledky. Nedodržanie plánu nepredstavuje jedinú hrozbu ovplyvňujúcich úspešnosť projektu. Je nutné sledovať aj prekročenie rozpočtu, kvalitu samotného vývojového procesu, efektívnosť použitých metód a prostriedkov a ostatné podporné činnosti projektu.

V tejto rozprave sa budeme venovať sledovaniu softvérových projektov. Predstavíme si zaujímavý model vývoja softvéru, ktorý umožňuje veľmi efektívne sledovať a hlavne prehľadne vizualizovať postup projektu. Ďalej rozoberieme prístup využívajúci pri sledovaní softvérového projektu princípy telemetrie. Na záver si uvedieme príklad sledovania postupu softvérového projektu priamo z akademickej pôdy.

Sledovanie postupu softvérového projektu a manažment, január 2006, s. 55-61.

Princípy sledovania postupu softvérového projektu

Základným princípom pri sledovaní postupu je zabezpečiť viditeľnosť projektu. Teda je nutné čo najpresnejšie určiť aktuálny stav projektu. Stav projektu je možné určovať periodickým zaznamenávaním a následne vhodným vyhodnocovaním špecifických charakteristík projektu. Avšak aké charakteristiky sledovať a ako ich vyhodnocovať, aby sme mohli exaktne určiť stav projektu? Neexistuje jednoznačná odpoveď na túto otázku. Jednotlivé projekty, aj keď sa týkajú rovnakej oblasti (v tomto prípade vývoja softvérových produktov), sú značne rozdielne. Je nutné zohľadňovať aj špecifické aspekty týchto projektov, aby sme mohli zodpovedne vyhodnocovať ich postup. Našťastie existuje viacero overených odporúčaní, ktoré výrazne uľahčia proces sledovania, pre ľubovoľný projekt [1]:

- *Plán projektu.* Je značne komplikované, ak nie nemožné, sledovať postup softvérového projektu, ak nie je stanovený akýsi referenčný model. Tým je v tomto prípade plán projektu, obsahujúci časové i finančné predpoklady. V stanovených bodoch, ktoré definuje samotný plán, je možné vyhodnotiť stav projektu, určiť prípadne odchýlky a prijať potrebné opatrenia.
- *Začiatok a koniec každej činnosti.* Dôkladné definovanie termínov pre jednotlivé činnosti a úlohy, na čo najmenšej úrovni abstrakcie, umožní v pláne identifikovať kritické miesta, ktoré treba obzvlášť dôkladne sledovať (metóda kritickej cesty).
- *Zdroje pre každú činnosť.* Ak sú v pláne dôkladne definované zdroje potrebné pre vykonanie danej činnosti alebo úlohy, a zároveň sa sledujú aj skutočne spotrebované zdroje, je možné stanoviť ukazovatele, umožňujúce nielen presne vyhodnotiť stav projektu, ale aj predpovedať ďalší postup projektu.
- *Výsledky činností.* Azda najťažšou úlohou, je sledovanie výsledkov jednotlivých činností. Výsledkom každej etapy v životnom cykle vývoja softvéru má byť určitý výstup. Ak chceme efektívne sledovať postup, je nutné vyhodnocovať aj tieto výstupy. Nástroje na vyhodnocovanie výstupov jednotlivých etáp v životnom cykle vývoja softvérov poskytuje manažment akosti a meranie v softvérovom projekte.

Sledovanie postupu a modely životného cyklu vývoja softvéru

V [3] sa uvádza, že súčasné modely životného cyklu vývoja softvéru podporujú sledovanie postupu projektu len málo alebo vôbec nie. Zdôrazňujú, že tento proces manažmentu projektu, je podstatný a je vhodné ho zakomponovať priamo do modelu životného cyklu vývoja softvéru. V článku sa demonštruje Rekurzívny multivláknový model (RMT).

RMT je založený na dekompozícií vyvíjaného softvérového systému do tzv. vlákien. Vlákno predstavuje činnosť – implementáciu softvérového komponentu. Každé vlákno pozostáva z niekoľkých fáz (analýza požiadaviek, plánovanie, analýza,

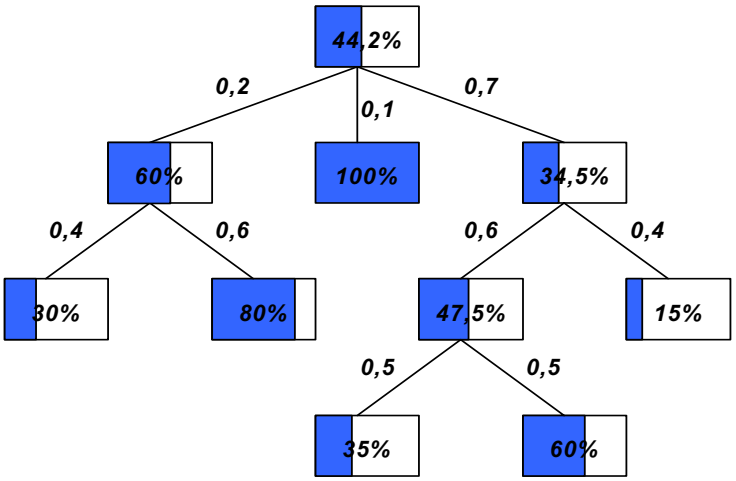
návrh, implementácia, testovanie). Jednotlivé vlákna sa priradia manažérom a tímom vývojárov, ktorí sú zodpovedný za ich realizáciu. Vlákna je možné jednak dekomponovať a vytvárať tak hierarchiu (rekurzia vlákien), a jednak plánovať ich verzie (iterácia vlákien). Pričom každé vlákno, ktoré je potomkom, má priradenú váhu zodpovedajúcu percentuálnemu podielu úsilia potrebného na vykonanie jeho rodiča (suma váh všetkých potomkov daného rodiča je 100%). Taktiež každá iterácia je ohodnotená percentom z celkového úsilia potrebného na realizáciu daného vlákna. Takto štruktúrovaný projekt je možné následne veľmi efektívne sledovať. Vyhodnotenie stavu projektu má potom 2 fázy:

1. *vyhodnotenie listov* – $Effort^{leaf} = \sum_{j=1}^{k-1} e_j + p \cdot e_k$, pričom $Effort^{leaf}$ – percento vyjadrujúce vykonané úsilie v danom liste, e_j – percento prislúchajúce *j*-tej iterácií, e_k – percento prislúchajúce aktuálne rozpracovanej iterácií, p – percento vyjadrujúce stav aktuálne rozpracovanej iterácie.

2. *vyhodnotenie uzlov* – $Effort^i = \sum_{j=1}^{J} w_j \cdot Effort_j^{i-1}$, pričom $Effort^i$ – predstavuje úsilie vykonané na *i*-tej úrovni, $Effort_j^{i-1}$ – predstavuje úsilie vykonané na (*i*-1)-vej úrovni pre *j*-teho potomka, w_j – predstavuje podiel úsilia potomka potrebného na ukončenie rodiča.

Autori tohto modelu implementovali uvedené koncepty do jednoduchého CASE nástroja s názvom RMT Tool. RMT Tool umožňuje veľmi efektívne a prehľadne zobrazovať postup v softvérovom projekte. Príklad vizualizácie vytvorenej počas riešenia projektu je uvedený na **Obr. 1**.

Každý uzol umožňuje zobraziť doplňujúce informácie (stav prislúchajúcich fáz a zoznam riešiteľov). Informácie, poskytované pomocou nástroja RMT Tool, sú prístupné jednak pre vývojárov, pre projektových manažérov, ale najmä pre tzv. manažérov postupu (angl. progress managers). Manažéri postupu majú na starosti analyzovanie a vyhodnocovanie existujúcich dát (generovanie rôznych reportov) a doplňanie nových dát. Takéto sledovanie a zaznamenávanie postupu projektu umožňuje predpovedať ďalší vývoj, odhadovať ukončenie jednotlivých vlákien a aj celého projektu.



Obr. 1 Vizualizácia stavu softvérového projektu vytvorená pomocou RMT Tool

Uvedený prístup predstavuje zaujímavý spôsob sledovania postupu v softvérom projekte, avšak má podľa môjho názoru niekoľko nedostatkov. Prvým nedostatkom je, že vôbec neuvažuje fakt, že ukončenie všetkých potomkov nemusí znamenať zákonite aj ukončenie prislúchajúceho rodiča. Keď si uvedomíme, že vlákno predstavuje implementáciu softvérového komponentu, potom je jasné, že úspešným implementovaním všetkých komponentov nedostávame hneď aj fungujúci modul alebo podsystém. Analogicky implementovaním všetkých modulov alebo podsystémov nedostaneme hneď fungujúci systém. V prípade softvérových projektov, v tejto situácií, určite neplatí jednoduchá rovnosť 1+1=2. Riešením by mohlo byť naplánovanie a vytvorenie ďalšieho vlákna, ktoré by predstavovalo činnosť spojenú s integráciou prislúchajúcich komponentov, modulov a podsystémov do väčšieho celku. Ďalším nedostatkom je fakt, že výhody takéhoto modelu sledovania postupu softvérového projektu sa strácajú v tom prípade, ak sa aplikuje na rozsiahly projekt. Takýto projekt môže mať tisícky vlákien a uvedený graf postupu by bol enormne rozsiahly. Sami autori priznávajú, že tento model je vhodný pre malé a stredne veľké projekty. Teda mohol by byť využitý napríklad aj pri riešení tímového projektu. Posledný nedostatok vidím v tom, že tento prístup nedáva návod ako sledovať jednotlivé činnosti, aké charakteristiky sledovať a ako ich vyhodnotiť tak, aby sme mohli doplniť do grafu údaj, že toto vlákno je ukončené napríklad na 33%. Pri výpočte vynaloženého úsilia pre listové uzly sa sumujú jednotlivé iterácie, podľa môjho názoru nie je možné dostatočne exaktne dopredu určiť počet iterácií a ešte aj prislúchajúce percento z celkového úsilia potrebného na ukončenie daného vlákna. Okrem toho vo vzorci vystupuje ešte parameter *p* vyjadrujúci postup aktuálne rozpracovanej iterácie. Ako teda určiť skutočne objektívnu hodnotu tohto parametra?

Telemetria v softvérovom projekte

V predchádzajúcej časti sme prestavili model vývoja softvéru založenom na precíznej dekompozícií vytváraného systému a podrobnom pláne, na základe ktorého sa sleduje postup projektu. Pričom uvedený plán sa vytvára väčšinou na základe skúseností z predchádzajúcich projektov a s využitím nejakého zaužívaného modelu. V [4] sa diskutuje prístup so zaujímavým názvom Telemetria v softvérovom inžinierstve. Pod pojmom telemetria sa chápe výrazne automatizovaný proces vzdialeného merania a zhromažďovania dát. Pričom dôraz sa kladie práve na to, aby sa dáta zbierali vzdialene a automatizovane, t.j. aby boli vývojári a vedúci pracovníci odbremenení od periodického ručného zhromažďovania dát. Ďalším kľúčovým faktorom tohto prístupu je okamžitý prístup k dátam vyplývajúci z toho, že dáta sa zbierajú kontinuálne a nie v diskrétnych časových okamihoch. Aké dáta sa vlastne sledujú? Autori uvádzajú nasledovnú kategorizáciu:

1. *telemetria pri vývoji* – sleduje sa počet editovaných súborov, čas strávený v danom vývojovom prostredí, zmeny vykonané v jednotlivých častiach systému (artefakty), počet riadkov kódu a poradie vykonávaných príkazov.
2. *telemetria pri zostavovaní* – sledujú sa výsledky z kompilácie, linkovania a unit testov.
3. *telemetria pri vykonávaní* – sleduje sa správanie systému počas behu (angl. runtime) a výsledky rôznych záťažových testov (angl. load and stress tests).
4. *telemetria pri používaní* – sleduje sa správanie používateľa a jeho interakcia so systémom.

Ako súvisí takáto telemetria so sledovaním postupu v projekte? Telemetria poskytuje, v kombinácií s klasickými metódami, veľmi efektívny nástroj ako včas určiť anomálie a rôzne odchýlky v projekte a podporuje okamžité rozhodovanie o nasledujúcom postupe. Analógiou môžu byť preteky formuly 1. Ak by sa technici a stratégovia stajne dozvedeli požadované dáta o stave monopostu iba počas naplánovanej zastávky v boxoch, mohli by zasiahnuť do monopostu a reagovať tak na špecifické situácie až počas ďalšej zastávky. A vtedy už môže byť neskoro. Avšak vzhľadom na to, že monitorujú monopost kontinuálne, môžu okamžite odhaliť prichádzajúce problémy, obmienať plán alebo predpovedať ďalší priebeh pretekov. Takýto prístup umožňuje veľmi flexibilný spôsob riadenia projektu.

Samotné zbieranie dát sa vykonáva pomocou tzv. senzorov monitorujúcich príslušný program. V aktuálnej verzii systému HackyStat[2], ktorý implementuje tento prístup, existujú senzory pre bežne používané vývojové prostredia ako: Eclipse, Emacs, JBuider, Vim, Visual Studio, pre kancelárske balíky Word, Excel, pre zostavovacie nástroje ako sú Ant, Make, pre monitorovanie práce s CVS a unixovským príkazovým riadkom, a nakoniec aj pre automatizované testovanie pomocou JUnit a meranie pomocou nástroja JMeter. Architektúra systému umožňuje jednoducho

[2] www.hackystat.org

pridávať ďalšie senzory pre špecifické nástroje a tiež definovať aké dáta sa majú sledovať.

Z uvedeného je zrejmé, že takýto prístup vedie k enormnému množstvu nazbieraných dát. Autori systému HackyStat prišli na tento problém počas využívania tohto systému priamo pri jeho vývoji. Riešením bolo vybudovanie kontrolného centra, ktoré zahŕňalo 9 monitorov kontinuálne zobrazujúcich rôzne sledované charakteristiky. Skúsenosť s takýmto využitím telemetrie bola priaznivá, samotní vývojári sa - pri prechádzaní okolo - zastavovali v kontrolnom centre. Mohli sledovať ako napreduje ich projekt a taktiež získavali potrebný nadhľad nad celým projektom. Zaujímavé by bolo sledovať, či by sa takýto sústavný monitoring ujal v externom prostredí, či by pracovníci pracovali vo firme, kde by boli sledovaní pri každej činnosti. Napriek týmto otvoreným otázkam si myslím, že sa jedná o veľmi zaujímavý a perspektívny prístup, ktorý má potenciál ujať sa napríklad v rámci agilných prístupov k vývoju softvérových systémov.

Sledovanie postupu a tímový projekt

Ako zakomponovať uvedené prístupy do riešenia tímového projektu? Prvý prístup, si vyžaduje vytvoriť precízny plán. Po vytvorení podrobného plánu, ktorý zahŕňa dostatočnú úroveň detailov pre jednotlivé činnosti, je možné aplikovať princíp modelu RMT. Využitie telemetrie je menej vhodné. Tímový projekt sa nerieši dostatočne dlhé obdobie a neprebieha tak intenzívny vývoj, aby bolo možné počas jedného semestra využiť jej výhody. Oba prístupy však majú spoločné to, že sa snažia automatizovať sledovanie postupu. Podobný spôsob sa aplikuje aj na Arizona State University v USA. Kde študenti počas bakalárskeho štúdia riešia veľmi podobný projekt ako študenti FIIT v rámci tímového projektu. Na tejto univerzite je vybudovaný čiastočne automatizovaný spôsob vyhodnocovania postupu v projekte, ktorý slúži inštruktorom jednotlivých tímov pri ich kontrole, ale hlavne študentom samotným. Pomáha im sledovať aktuálny stav projektu, na základe ktorého môžu modifikovať ďalší plán. Keďže je možné prehliadať aj progres ostatných tímov, zvyšuje sa zároveň úroveň súťaživosti, čo motivuje študentov k lepším výsledkom.

Nástroj predstavuje webovú aplikáciu, pričom podstatou sledovania postupu je zadávanie týždenných reportov. Tieto sa v systéme spracovávajú a následne sa generujú rôzne štatistiky a prehľady. Napríklad čas strávený pri riešení projektu celkovo, ale aj v jeho jednotlivých fázach alebo čas strávený na tímových stretnutiach. Pričom sa dá vyhodnocovať tím ako celok, ale aj jednotliví členovia tímu.

Uvedený prístup má avšak jednu slabinu. Ako zaručiť, že študent vloží do systému ozaj pravdivé dáta? V tomto mieste sa nám kruh uzatvára, dostávame sa opäť na začiatok. Teda aké objektívne merateľné charakteristiky sledovať a ako ich vyhodnocovať, aby sme mohli exaktne určiť stav projektu? Mohlo by sa zdať, že uvedené prístupy nám neprinesú žiadaný recept. Ale nie je to tak. Môžu výrazne prispieť k zlepšeniu celkového procesu, ale nikdy nemôžu nahradiť ľudský faktor. Dobrý projektový manažér, ktorý participoval pri riešení desiatok projektov, má „zabudovaný“ vlastný senzor, ktorý nadobudol skúsenosťami a určite aj tým, že sa

poučil na vlastných chybách. Neznamená to avšak, že treba zanevrieť na uvedené princípy a postupy, treba ich aplikovať, skúšať a modifikovať, ale s rozvahou.

Záver

Ak sa chceme ocitnúť v pozícií manažéra úspešne zakončeného softvérového projektu, je nevyhnutné kontinuálne sledovať jeho postup, reagovať na rôzne odchýlky a anomálie, modifikovať plán, opäť sledovať, reagovať a modifikovať... Pri tomto procese nám môžu veľmi pomôcť viaceré metódy, odporúčania a nástroje, ktoré automatizujú proces sledovania. Ich výsledky treba vyhodnocovať opatrne a hlavne zo zreteľom na ľudí, ktorých sa týkajú.

Použitá literatúra

1. Bieliková, M.: *Softvérové inžinierstvo*. Princípy a manažment. Vydataveľstvo STU, Bratislava, 2000.
2. Collofello, J. S., Hart, M.: Monitoring Team Progress in a Software Engineering Project Class. In: *29th ASEE/IEEE Frontiers in Education Conference*. November 10 - 13, 1999 San Juan, Puerto Rico.
3. Conception, A. I., Lin, S., Simon, S. J.: The RMT Tool: A Computer Aided Software Engineering Tool for Monitoring and Predicting Software Development Progress. In: *Proceedings of the 21st International Conference on Software Engineering* (ICSE'99).
4. Johnson, P. M., Kou, H., Paulding, M., Zhang, Q., Kagawa, A., Yamashita, T.: Improving Software Development Management through Software Project Telemetry. *IEEE Software*, Vol. 22, No. 4 (2005), pp. 76-85.

Annotation

Monitoring progress of software project and management

Many software projects face difficulties because their monitoring relies too strongly on subjective information from the team members. Deviations from planning are often recognized too late. Automated project monitoring can alert managers on coming problems and allow them to take essential precautions. Therefore the key point during monitoring progress is visualizing real project state automatic and anytime.