Monitorovanie vývojového procesu softvérového produktu, jeho prínos a analýza možných prístupov

DUŠAN LAMOŠ

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

Abstrakt. Prevažná väčšina súčasného softvéru je výsledkom koordinovanej práce veľkého počtu ľudí, pričom hovoríme o tzv. softvérových projektoch. Nie každý softvérový projekt však musí nutne viesť k vzniku hotového softvérového produktu. V ceste tejto snahe stoja jednak problémy s dodržiavaním stanoveného časového harmonogramu projektu, ako aj problémy spojené s kvalitou kódu a obmedzeným rozpočtom.

Vzhľadom na rozsah súčasných projektov a s tým súvisiace nemalé finančné náklady, je viac než žiadúce zabezpečiť úspešný koniec vývojového procesu. Aby bol daný softvérový projekt úspešne realizovateľný, je nutné sledovať priebežný stav spomínaných faktorov, čiže monitorovať proces vývoja. Na základe zistených skutočností sa následne vykonajú vhodné opatrenia, čím sa zasahuje do procesu plánovania vývoja. Priebežné monitorovanie je nevyhnutné hlavne z dôvodu neustále sa meniacich podmienok procesu vývoja, či už sa jedná o zdroje vo vnútri spoločnosti, alebo vonkajšie podmienky trhu atď.

Spôsob monitorovania závisí od sledovanej veličiny. V praxi hovoríme často o tzv. metrikách (či už kvantitatívnych, alebo kvalitatívnych), pričom od ich vhodnej voľby závisí efektivita a prínos samotného monitorovania. Proces získavania údajov pre jednotlivé metriky je závislý od monitorovanej veličiny. Môže sa jednať napríklad o rôzne dotazníky, ale aj o štatistické výstupy z vývojových prostredí atď. Táto problematika je bližšie analyzovaná v hlavnom texte.

Úvod

Monitorovanie procesu vývoja softvéru, ako už samotný význam pojmu *monitorovanie* naznačuje, predstavuje *priebežné* sledovanie stavu určitých vybraných veličín, či už

Manažment v softvérovom inžinierstve, október 2006, s. 1-7.

kvantitatívneho alebo kvalitatívneho charakteru. Jedná sa o vhodne vybrané veličiny, na základe ktorých je možné odhadovať momentálnu mieru úspešnosti riešenia projektu.

Cieľom každého projektu je vytvorenie produktu, zodpovedajúceho stanoveným požiadavkám, v stanovenom čase a s dodržaním stanoveného rozpočtu (resp. s minimálnym rozpočtom). Práve monitorovanie procesu vývoja nám pomáha tento cieľ dosiahnuť, keďže sme na základe zisteného momentálneho stavu schopný kedykoľvek počas trvania projektu rozhodnúť o jeho ďalšom smerovaní a v prípade zistených nedostatkov vykonať príslušné kompenzačné opatrenia. Táto flexibilita výrazne zvyšuje pravdepodobnosť úspešného ukončenia projektu.

Možné úskalia softvérového projektu

Základnou podmienkou úspechu každého softvérového projektu je jeho kvalitný a efektívny manažment. Či už sa jedná o oblasť plánovania a riadenia projektu, manažment rizík, alebo ľudské zdroje, kľúčom k efektívnemu manažmentu je prístup k aktuálnym a relevantným informáciám, ktoré umožňujú vykonať v daný čas "to správne" rozhodnutie. Podrobnejšie je problematika manažmentu a monitorovania SW projektov rozoberaná v [1]

Veľké množstvo projektov stroskotá (prípadne sú neúmerne prekročené investície, alebo sa nepodarí dodržať stanovený časový rámec) práve preto, že sa nedostatky a chyby plánovania a rozhodovania zistia príliš neskoro, prípadne sa nemusia zistiť vôbec.

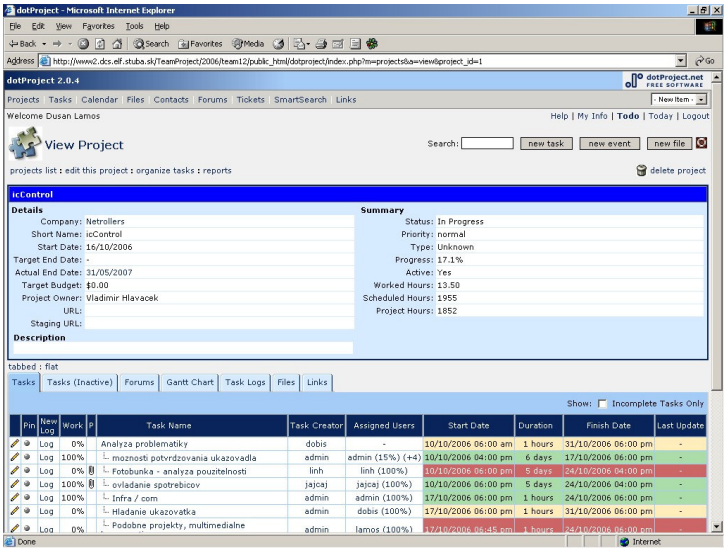
Už pred začatím práce na projekte musí zodpovedný manažér správne odhadnúť veľkosť a zložitosť budúceho softvérového systému, aby mohol správne stanoviť množstvo potrebných pracovných síl a stanoviť predpokladaný rozpočet [1]. Následne sa stanoví plán projektu a delegujú sa úlohy. V našom konkrétnom prípade (5-6 členný tým na predmete Tímový projekt) sa úlohy delegujú na základe analýzy osobitých schopností jednotlivých členov tímu, v oblasti veľkých tímov sú úlohy členov zväčša už pevne stanovené. Po vykonaní všetkých spomínaných úkonov je možné začať prácu na samotnom projekte.

Počas procesu tvorby SW systému sa môžu objaviť rôzne viac či menej predvídateľné problémy. Môžu to byť napríklad náhle zistené sklzy v plnení projektového plánu, nemenej dôležité sú aj zmeny vnútorných, alebo vonkajších podmienok. Spomeňme napríklad podmienky trhu, alebo náhle zmeny požiadaviek zo strany zákazníka. Rovnako je nutné reagovať aj na zmeny na personálnej úrovni. Priebežné monitorovanie je nutné aj z dôvodu, že vývoj softvéru prebieha v určitom vývojovom cykle s niekoľkými diskrétnymi štádiami, pričom v každom z týchto štádií je nutné sústrediť pozornosť na iné ťažiskové body.

Monitorovanie SW projektov

Počas práce na projekte je potom nutné priebežne monitorovať jej postup, tj. koľko práce už bolo vykonanej a koľko ešte len treba vykonať, pričom skutočný stav neustále porovnávame s želaným stavom, uvedeným v pláne projektu. Rovnako, ako dodržiavanie časového harmonogramu, monitorujeme priebežne aj kvalitu vykonanej práce.

Postup pri monitorovaní projektu závisí do veľkej miery aj od jeho rozsahu. Pri projekte malého rozsahu s niekoľko málo členmi v tíme je monitorovanie neporovnateľne jednoduchšie, ako v prípade rozsiahlych projektov. V malom tíme je komunikácia medzi jeho členmi pomerne jednoduchá, postačujú preto občasné (v ideálnom prípade pravidelné) stretnutia členov tímu, počas ktorých sa rozoberá aktuálny stav projektu a stav riešenia úloh, delegovaných jednotlivým členom, následne sa rozhodne o ďalšom postupe. Okrem osobných stretnutí a rôznych projektových denníkov je vhodné používať taktiež dodatočné podporné prostriedky, napríklad vhodný elektronický systém pre správu projektu. Pri malých projektoch nie je takýto systém bezpodmienečne nutný, zefektívňuje však koordináciu členov tímu a umožňuje im pohodlný a rýchly prístup k informáciám (napr. cez webové rozhranie). K tomuto rozhodnutiu sme dospeli aj v rámci predmetu Tímový projekt, kde náš tím používa systém dotProject (pozri Obr. 1), ktorý je voľne dostupný v rámci projektu Open Source.



Obr. 1. Príklad obrazovky systému dotProject

Odlišný prístup je nutný v prípade rozsiahlych projektov, kde je potrebné koordinovať prácu veľkého množstva ľudí. Aj v takomto prípade sa využívajú podporné prostriedky [1], bývajú však spravidla komplexnejšie a štandardizované (napríklad produkty firmy Rational Software, Starbase, alebo Telelogic [4]). Pri rozsiahlych projektoch je nutné vo väčšej miere dbať na formálnu stránku vývojového procesu, ako je tvorba rôznych modelov systému (napr. v praxi často používané modely znázornené UML diagramami), na ktorých sme schopný odhaliť prípadné chyby a nedostatky v návrhu ešte v počiatočnej fáze projektu, nie až vo fáze implementácie, kde by bolo odstránenie tej istej chyby vyžadovalo neporovnateľne vyššie náklady.

Vytvorenie podobného modelu (modelov) systému umožňuje úplne samostatný prístup k monitorovaniu [3]. V tomto prípade je použitý spomínaný jazyk UML, od ktorého sa odvíjajú monitorované veličiny (v oblasti softvérového inžinierstva sa tiež často používa výraz *metrika*). Sledujeme napríklad mieru chybovosti diagramov prípadov použitia, alebo ich úplnosť. V oboch prípadoch ide o tzv. *kvalitatívne metriky*, čiže údaje, ktoré vypovedajú o kvalite vykonanej práce, nie však o aktuálnom stave dodržiavania stanoveného plánu projektu (na tento účel sa využívajú tzv. *kvantitatívne metriky*). Niektoré kvantitatívne metriky z oblasti UML schém sú uvedené v tab. 1, prevzatej z [3]. Softvérovým metrikám sa budem podrobnejšie venovať neskôr.

Artefakt	**Projekt 1**	**Projekt 2**	**Projekt 3**
Modelovací nástroj	Rose™		
Databáza požiadaviek	RequisitePro™	DOORS™	Caliber™
Prípady použitia	179	804	1121
Poč. diagramov prípadov použitia	40	224	1013
Hráči	20	160	96
Iné typy diagramov	26	138	74
Priemerný počet prípadov použitia na diagram	5.5	5.0	2.4

Tab. 1. Informácie o projektoch

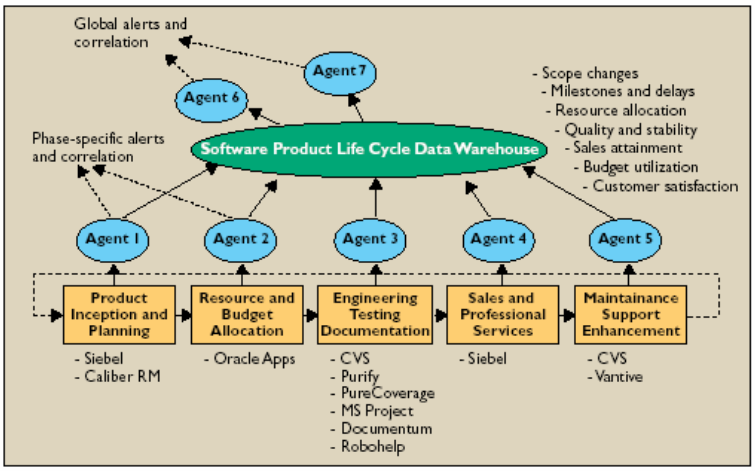
Softvérové metriky a spôsoby získavania príslušných údajov

Pri monitorovaní SW projektov používame, ako už bolo v texte spomenuté, tzv. softvérové metriky. Ako príklad uvádzam niektoré metriky zo životného cyklu softvérového produktu, uvedené v [4]. Uvedené metriky majú všeobecný charakter.

- Zmeny oblasti zamerania projektu
- Časové odstupy medzi tzv. míľnikmi projektu
- Rýchlosť zmeny funkčnej špecifikácie
- Percento celkovej časti systému, ktoré už prešlo testovaním
- Čas potrebný pre odstránenie závad
- Čas venovaný stretnutiam tímu
- Čas odozvy pri riešení problémov zákazníka
- Dostupnosť kritických systémov

V ideálnom prípade sú dáta zhromaždené a uložené v "skladisku dát" (Ang. *data warehouse*) životného cyklu produktu, kde sú dostupné pre generovanie pohľadov, automatizovaných upozornení a pravidiel a korelačných pravidiel z každodenných úkonov, ďalej zhrnutí stavu produktu a agregovaných dát na najvyššej úrovni a pre oblasť riadenia. Tieto pohľady, upozornenia a pravidlá slúžia rôznym skupinám užívateľov, od manažéra sortimentu produktov až po procesného analytika a od technika zodpovedného za testovanie až po hlavného technického vedúceho [4].

Informácie pre hore uvedené metriky je možné spravidla získavať rôznymi spôsobmi. Môžeme napríklad vytvoriť dotazníky, ktoré vyplnia jednotlivý členovia tímu, resp. pri projektoch väčšieho rozsahu, vedúci tímov a iný vedúci pracovníci. Výhodnejšie je však prenesenie úlohy získavania údajov na autonómne softvérové agenty. Možný prístup, využívaný v systéme nazvanom MAUI (pozri [4]) využíva monitorovanie pomocou agentov znázornené na Obr.2.



Obr. 2. Riadený životný cyklus softvérového produktu

Jednotlivé softvérové agenty samočinne dolujú dáta z danej problémovej oblasti. Tieto dáta sú ukladané do spoločnej databázy (*Software product life cycle data warehouse*), odkiaľ sú potom prístupné pre ďalšie spracovanie.

Systém MAUI okrem samotného monitorovania softvérových metrík generuje na požiadanie prehľady aktuálneho stavu práce na projekte, ďalej umožňuje napríklad automatické adresné rozosielanie poplašných správ pracovníkom, napríklad v prípade hroziaceho problému. S týmto je spojená jedna pozoruhodná vlastnosť systému, a to jeho schopnosť na základe zistených kvalitatívnych a kvantitatívnych údajov o priebehu projektu predpovedať hroziace problémy, ako napríklad oneskorené dosiahnutie nasledujúcich "míľnikov" vo vývoji produktu. Týmto odoberá manažérovi značnú časť práce a umožňuje mu udržať si prehľad aj v rozsiahlom projekte, kde by bolo za normálnych okolností nutné delegovať časť kompetencií manažéra na iné osoby. Dopad na riadenie projektu je zrejmý, keďže sme včas informovaní o hroziacom probléme, máme dostatok času vykonať kompenzačné opatrenia a zamedziť vzniku daného problému, alebo aspoň zmierniť jeho možný rozsah a následky.

Metriky, ktoré boli v texte uvedené ako príklad, sú značne všeobecného charakteru. Pre ilustráciu uvediem metriky, konkrétne využívané spomínaným systémom MAUI (Tab. 2), prebraté z [4].

Metric	Definition	
Stability index	Last week percent change – 70%	percent of the code that has changed in a week
	Last week active users – 10	number of developers that made changes in a week
	Avg. LOC per active user – 10	average number of LOC per developer
	Avg. active user life span – 10	average time spent by a developer on the project
Quality index	Avg. defect line changes – 40	average number of LOC needed to fix a defect
	Documentation level – 30	number of LOC per every line of comment
	Active defects – 20	number of defects
	Avg. routine length – 10	average length of a routine
Documentation level	Number of LOC per every line of comment	
Weekly turmoil	Percent or the source code that changes in a week	
Defect complexity	Average LOC needed to fix a defect	
Average time spent by developers on project	Average length of time a developer has been working on the project or component.	
Average lines of code per developer	How many LOC each developer on the team is responsible for (total LOC/team size)	
Team size	Number of developers active in the last six months	

Tab. 2. Definícia metrík systému MAUI; LOC = *lines of code*

Systém MAUI má množstvo ďalších užitočných funkcií, ktorými sa v tomto texte nebudem dopodrobna zaoberať, v prípade záujmu zo strany čitateľa si dovoľujem odkázať na [4]. Aj keď by sa pri čítaní textu mohlo zdať, že sa príliš zaoberá jedným konkrétnym systémom (MAUI), princípy a vlastnosti, ktorými sa tento systém vyznačuje, majú všeobecnejšiu platnosť. Preto bol tento systém vybraný ako ilustračný príklad, rovnako mohol byť použitý ktorýkoľvek iný ekvivalentný systém.

Záver

Kvalitný a efektívny manažment je kľúčovým faktorom pre úspech každého softvérového projektu. Podmienkou kvalitného manažmentu je včasný prístup k relevantným informáciám o projekte, pričom práve takéto informácie nám poskytuje monitorovanie SW projektu.

Pri SW projektoch s veľkým rozsahom je nutné používať technické podporné prostriedky, riešiace jednak komunikačné problémy medzi jednotlivými účastníkmi projektu, ako aj problémy spojené so zberom informácií pre monitorovanie. Takéto systémy môžu takisto poskytovať prognózy pre včasné predpovedanie možných problémov a umožniť tak vykonať vhodné opatrenia ešte pred nastatím kritickej situácie.

Pri projektoch malého rozsahu, tvorených zväčša v niekoľko málo členných tímoch, je situácia o poznanie jednoduchšia. Napriek tomu je výhodné používať podporné prostriedky, zamedzí sa tak prípadným komunikačným problémom a možným nedorozumeniam a zároveň sa jednotlivým členom tímu uľahčí prístup k informáciám.

Použitá literatúra

1. Tsoi, H.L.: A framework for management software project development. Proceedings of the 1999 ACM symposium on Applied computing (1999) 593 - 597.
2. Nienaber, R., Cloete, E.: A software agent framework for the support of software project management. ACM International Conference Proceeding Series; Vol. 47 (2003) 16 - 23.
3. Berenbach, B., Borotto, G.: Metrics for model driven requirements development. Proceeding of the 28th international conference on Software engineering (2006) 445 - 451.
4. Suardi, L.: How to manage your software product life cycle with MAUI. Communications of the ACM, Vol. 47, No. 5 (2004) 89 - 94.

Annotation

Monitoring of the software production process, its contributions and an analysis of possible approaches

The importance and benefits of monitoring of software projects are discussed. Approaches for small and large scale projects are analysed and briefly compared, examples of existing systems are presented and conclusions are drawn.