DÔLEŽITOSŤ SRDCA A OBALU SOFTVÉROVÉHO PRODUKTU

Nikto nikdy nevie, čo sa skrýva za škrupinou skôr ako ju rozlúskne.

Martin Gregor

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
Greczsky@gmail.com

Abstrakt. *Viete si predstaviť aké je to pri zlyhaní projektu? Najmä pri vývoji softvéru je kľúčovým kritériom aby projekt nezlyhal. O to sa postará manažment monitorovania, ktorý metódami kontroly vývoja a špeciálnymi metrikami dokáže určiť nielen aktuálny stav projektu, ale aj predpovedať progres vývoja a odhadnúť cenu a zdroje potrebné k jeho dokončeniu. Monitorovať sa dá aj interný aj externý stav vývoja. Po internej stránke sa myslí kvalita kódu, ktorá zabezpečí integritu, stabilitu a robustnosť systému. Externý stav sú splnené funkčné a vonkajšie viditeľné vlastnosti produktu. My sa spolu pozrieme na metódy a možnosti monitorovania vnútornej aj vonkajšej stránky vývoja a ich vplyv na výsledný softvérový produkt.*

Kľúčové slová: *monitorovanie internej kvality softvérového projektu, monitorovanie zdrojového kódu, monitorovanie externej kvality softvérového projektu*

Prečo monitorovať vývoj softvéru?

Nejeden manažér si pomyslí: „Prečo by sme mali monitorovať vývoj softvéru? Veď máme skúsených ľudí, ktorí to zvládnu.“ A práve to sú manažéri, pod ktorých vedením projekty väčšinou nedosiahnu konečnú podobu. Chybou je, že väčšina softvérových projektov je monitorovaná iba na začiatku vývoja [5]. A to je etapa projektu, kedy je o projekte najmenej informácií. Manažment monitorovania vývoja softvéru patrí medzi základné postupy počas vývoja softvéru. Sú to určité postupy, ktoré analyzujú súčasný stav projektu vzhľadom na to, čo už je dokončené a čo ešte treba urobiť. Tieto postupy majú za

úlohu hlavne pomáhať pri predchádzaní zlyhania projektu, ale aj pri odhade ceny, trvania a potrebných zdrojov projektu. Kto všetko stratí hodnoty pri zlyhaní projektu, a kto všetko je zodpovedný za chyby softvérového produktu? Na tieto otázky odpovedá samotný manažment monitorovania projektov. Manažment monitorovania softvérových projektov [1] sa môže zamerať na kontrolu internej a externej stránky softvérového projektu. Poďme sa pozrieť aké nástroje, postupy vyžaduje manažment monitorovania internej aj externej kvality vývoja softvérového projektu.

Interné monitorovanie

V [6] tvrdí, že väčšina manažerov hľadí na to, čo už je urobené a nie na to čo ešte treba urobiť. A to môže byť problém lebo budúcnosť projektu nie je v pozornosti tímu. Neskôr môžu nastať napríklad chyby spôsobené neprehľadnosťou kódu a zdržaním vývojara nad refaktorizáciou takého kódu [3]. Preto monitorovanie kvality kódu softvéru, čo je interné monitorovanie softvérového projektu, je veľmi dôležité pri strategických rozhodnutiach manažéra. Vo väčšine prípadov je pre manažment interná stránka projektu neprehľadná a dokonca až neviditeľná. A prečo je monitorovanie internej kvality až tak dôležité? Pri dlhodobo vyvýjaných projektoch je dôležité aby implementovanie nových vlastností softvéru, nových features neprinášalo nové problémy (bugy), ktoré súvisia so zlou internou kvalitou softvéru. Čiže to je napríklad keď má kód slabú robustnosť a integritu. Ďalším problémom je aj už spomínaná prehľadnosť kódu, vyžadujúca refaktorizáciu a tým aj zdržanie vývojara. Nekorektné správanie vývojara môže spôsobiť aj zložitosť algoritmov v kóde a pri implementácií novej vlastnosti sa môže celý projekt správať úplne rozdielne od očakávaní.

Myslím, že toto sú vážne dôvody prečo je pre manažment dôležité monitorovanie internej kvality. Z uvedených príkladov možno usúdiť, že monitorovaním internej kvality manažment získa strategické informácie, podľa ktorých sa rozhodne na akú časť vývoja nasadiť koľko vývojarov.

Nestrácajme hlavu, že ako to teraz dosiahnuť! Veď sme skúsení manažéri a nejaké monitorovanie nám problém robiť nemôže, keď existuje toľko softvérovo-monitorovacích nástrojov. Ale aj tak na začiatku monitorovania softvérového projektu je potrebné si stanoviť metriky a miery, podľa ktorých budeme softvérový projekt monitorovať.

V nasledujúcej časti si priblížime CQMM, čo znamená *Code Quality Monitoring Method* [1]. Ako už z názvu vyplýva je to metóda, ktorá monitoruje internú kvalitu softvéru, teda kódu. Je založená na metóde vyhodnocovania internej kvality EMISQ (Evaluation Method for Internal Software Quality). CQMM metóda sa sútredí na systematický, cieľovo orientovaný prístup. Predstavme si, že môžme plánovať rôzne opravné akcie ako sú refaktoring alebo prioritizovať opravu chýb. Toto všetko umožňuje CQMM počas životného cyklu vývoja softvéru. Integrácia CQMM do samostatných projektov je už od podstaty odlišná. Je to najmä kvôli rôznorodosti projektov vhľadom na zložité faktory ako je veľkosť kódu, veľkosť a sila tímu, požiadavky na kvalitu, použité technológie a ďaľšie citlivé faktory, v ktorých sa projekty líšia. Preto k ľahšej intergrácii metódy CQMM vznikli požiadavky ako je prispôsobiteľnosť výberu problémov kvality spoločných pre viaceré projekty, odľahčenosť pre správne používanie a rýchle prijatie metódy a zapojiteľnosť. Prezentácia výsledkov musí byť jednotná v celom projekte. Keď si

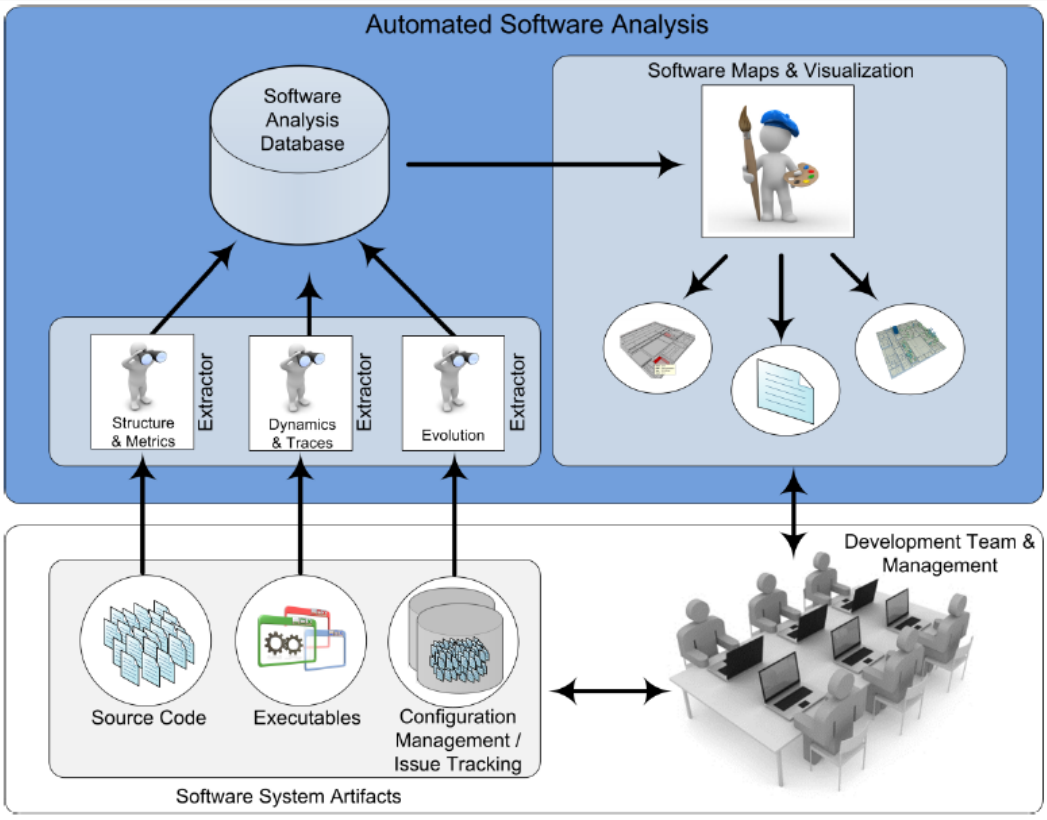
to zhrnieme tak CQMM metóda musí byť jednoduchá, presná v použití naprieč celým projektom a musí ponúkať výber rôznych problémov kvality. Výber rôznych problémov kvality sa získava z dát získaných v minulosti z vyhodnocovacej metódy EMISQ. Tu vidíme potrebu histórie použita metódy pre dosiahutie lepších výsledkov. Zjednodušenie metódy nám popisujú tieto kroky navrhnuté v [1] :

1. Analýza vývojového prostredia, kde najlepšie zjednodušenie je v použití jednotného vývojového prostredia v celej spoločnosti.
2. Definovanie cieľov kvality.
3. Definovanie modelu kvality, čo je definovanie zoznamu problémov kvality na základe historických dát z EMISQ.
4. Definovanie monitorovacieho a takzvaného krajčírskeho prístupu, ktorý zabezpečí naplánovanie a opravu problémov kvality.
5. Krajčírsky model kvality a náhľad na model. Vyhádzajú z definovaných cieľov kvality, zo zoznamu problémov kvality.
6. Integrácia do vývojového procesu

Hlavným cieľom metódy neustáleho monitorovania, a pre nás manažérov podstatným, je minimalizovanie problémov kvality kódu pomocou skorej detekcie. Podľa môjho názoru tento cieľ nie je ľahko implementovateľný. V tomto prípade je správne si určiť limity, kedy je kód kvalitný a kedy nie. Vyriešiť sa to dá určením množstva porušení kvality na verziu produktu. Pri určení množstva vychádzali z veľkosti kódu a množstva implementačných problémov na verziu. CQMM považovaný za všeobecný proces monitorovania a riadenia kvality kódu určite dokáže byť silným nástrojom v správných rukách.

Ako ďalší prístup monitorovania internej kvality uvediem monitorovanie pomocou softvérových máp alebo artefaktov [3]. Artefakty alebo mapy vznikajú rozdelením kódu na funkcionálne časti, ktoré zvykneme nazývať aj moduly. Podľa môjho názoru už samotné rozdelenie do modulov zvyšuje udržovateľnosť a prehľadnosť softvéru. Je to vďaka dokonale prehľadnej prezentácii softvéru. Medzi hlavné výhody monitorovania pomocou softvérových máp, ktoré uprednostňujem aj ja patrí, že nevyžaduje ďalšie nové metriky na monitorovanie kvality pomocou softvérových máp lebo samotná prezentácia poskytuje užitočné metriky. Pomocou softvérových máp dokážeme monitorovať aj internú aj externú kvalitu kódu a monitorovanie pomocou softvérových máp môžme využiť aj ako systém na zaznamenávanie chýb pri používaní správnych kľúčových slov v správach z monitorovania. Pre nás manažérov má monitorovanie pomocou softvérových máp vizuálnu 2,5 dimenzionálnu reprezentáciu, kde môžme vidieť kde a ktorý programátor koľko času strávil alebo aj ako je daný artefakt výpočtovo zložitý. Monitorovanie pomocou softvérových máp sa využíva v automatizovaných analýzach softvéru ako je znázornené na obrázku Obr. 1. Údaje o softvéri sú zbierané v analytickom databázovom serveri, z ktorých analytik vytvorí softvérové mapy z určených metrík, ktoré sa následne prezentujú nám a nášmu vývojovému tímu. Metriky sú stanovené ako dimenzie v dátovom sklade údajov o softvéri. Preto nás nemusí trápiť identifikácia ďalších nových metrík. Na neštastie sú tu problémy s určovaním hodnôt stanovených metrík, ktoré ale softvérové mapy s ľahskosťou riešia. Určenie či je súbor so zdrojovým kódom v dobrej alebo zlej kvalite sa ťažko určuje lebo je to relatívna hodnota vzhľadom na ostatné súbory.

Ale koľko je tých ostatných súborov? Softvérové mapy pracujú so zdrojovými súbormi rozdelenými na artefakty, čiže určovanie relatívnej hodnoty sa vykonáva vzhľadom na ostatné zdrojové súbory v artefakte. Určenie či je hodnota metriky príliš vysoká závisí na umiestnení kódu v architektúre systému. Artefakty v softvérových mapách sú rozdelené na základe architektúry softvérového systému. A je po probléme. Výhodou pre manažérov je, že dokážu správne odhadnúť cenu vývoja softvéru, správne naplánovať zdroje a čas na určitú časť kódu a tiež drill-down technikou vyhľadať konkrétny problém. Uveďme si príklad keď vývojari chcú pracovať na kóde s nižšou kvalitou tak im to asi bude trvať dlhšie ako práca na kóde s vyššou kvalitou. Tu si už vie každý manažér rozhodnúť koľko a akých zdrojov použije. Podľa môjho názoru sú softvérové mapy ťažko integrovateľné, ale na druhej strane sa určite oplatia pri monitorovaní dlhodobo vyvýjaného softvéru keďže manažérom ponúkajú automaticky generovaný, faktový a aktuálny prístup k reprezentácii kvality implementácie softvérového systému.



Obr. 1. Použitie softvérových máp v kroku vizualizácie v procese automatickej analýzy softvéru. Obrázok bol prevzatý z [3].

Externé monitorovanie

Investícia do externého monitorovania vývoja softvéru je oveľa rýchlejšie návratná ako z interného monitorovania [3]. Prečo je tomu tak? Z môjho pohľadu to vychádza hlavne

z primárneho zámeru manažmentu a to sú funkcionálne vlastnosti systému. Tieto vlastnosti sa dajú dosiahnúť aj menej kvalitnou, ale rýchlejšou cestou. Z toho môže vyplývať aktuálna spokojnosť manažmentu a zákazníkov. Samozrejme ale pri pohľade do budúcnosti softvérového produktu je otázna jeho udržovateľnosť a stabilita.

Potreba externého monitorovania vývoja softvérového produktu vznikla hlavne kvôli povrchnosti manažérov, ktorých zaujíma čo najrýchlejšie dokončenie projektu a tým vyšší zisk. Pri týchto myšlienkach vychádzam z vlastných skúseností z firmy, v ktorej momentálne pracujem. Technického manažéra nezaujímalo vôbec aké technológie použijem a akým štýlom ich implementujem. Išlo mu len o rýchle odovzdanie projektu. A ja ako vývojár som použil najrýchlejšie možné riešenie aby všetci boli spokojní. Ale nebolo tomu vždy tak. Vtedy bolo vidieť nedostatok v už spomínanom internom monitorovaní.

Monitorovanie externej kvality softvérového produktu sa hlavne konzultuje s vlastníkami softvérového produktu. Na začiatku monitorovania vývoja softvérového produktu je dôležité mať vytvorený model monitorovania so správnymi metrikami [4]. Môj osobný názor je redukovať množstvo metrík, tak ako to je opísané v [2]. Nielen kvôli vyššej prehľadnosti monitorovania, ktorú tam ja vidím, ale ja kvôli zvýšeniu presnosti odhadu ceny vývoja softvérového produktu. Dôležité je si atribúty zoradiť podľa použitia a vhodnosti, určiť im cenu. Cena sa dá určiť napríklad použitím COCOMO (Constructive Cost Model) modelu. Potom už následuje jednoduchý krok redukcie najlacnejších atribútov, ale postupne až kým posledných 5 odstránení neprinieslo zlepšenie monitorovania.

Metriky ale môžme vyberať na základe PLAN-DO-CHECK-ACT cyklu monitorovania v časti PLAN [4]. V ostatných častiach metriky meriame, analyzujeme a vyhodnocujeme. Pri výbere metrík zohľadňujeme ciele vlastníka softvérového produktu. To či je cieľ vlastníka splnený je často nejasná záležitosť. Túto nejasnosť by mal pomôcť ozrejmiť KGI, čo je kľúčový indikátor cieľa. Samozrejme každý jeden cieľ vlastníka softvérového produktu má iba jeden KGI. KGI sú buď určené metriky s požadovanými hodnotami alebo podrobnejšie špecifikované ciele vlastníka. Ďalším využiteľným indikátorom pri externom monitorovaní softvérového produktu s vlastníkmi je kľúčový indikátor procesu, ktorý vyjadruje postup k naplneniu cieľa. Ak by proces zlyhával musí mať definované opravné akcie vzhľadom na úroveň pokroku procesu. Namerané hodnoty indikátorov sa dajú vizualizovať do rôznych grafov, ktoré sú prínosom pre manažérov napríklad na zistenie úrovne pokroku k dosiahnutiu cieľov vlastníka.

Problémom väčšiny projektov je, že prejdu monitorovaním len na začiatku vývoja projektu. A vtedy samozrejme o projekte najmenej vieme [5]. Tak ako pri každej metóde monitorovania softvéru musíme aj pri metóde PPMC (Parametric Project Monitoring and Control) najskôr určiť metriky, pomocou ktorých budeme monitorovať stav vývoja. Z externého hľadiska sú to napríklad dokončené a nedokončené vlastnosti jednotlivých modulov, stav testovania nad daným modulom alebo počet vývojarov pracujúcich na danom module. Dôležitými následujúcimi krokmi sú určite priebežné monitorovania pri stanovených mílnikoch. Pri monitorovaní je tiež dôležité využívať históriu monitorovania resp. minulé prehliadky a ich výsledky. Zložitejšie časti projektu si môžme rozložiť na menšie časti a tie monitorovať. Výstupom PPMC sú rôzne užitočné grafy, v ktorých sú

zobrazené metriky meraní. Tento prístup je asi najpriamočiarejší a najednoduchší. Metóda PPMC je určite vhodná aj na menšie projekty.

Súvislosti medzi interným a externým monitorovaním

Pomer interného a externého monitorovania kvality softvérového produktu zavisí najmä na cene, ktorú sú vlastníci ochotní zaplatiť. Monitorovanie externej kvality softvérového produktu je jednoduchšie lebo nás nezaujímajú detaily vývoja, ale len výsledky a ciele vlastníkov. Ale úlohou manažmentu by malo byť prekonzultovanie dôležitosti interného monitorovania. Ako argumenty určite môžu použiť výsledky rôznych analýz ale aj skúsenosti z praxe a zdravý rozum, ktorý vlastníkom povie aká by mohla byť cena implementácií nových vlastností softvérového produktu v nekvalitnom kóde. Taktiež ako aj presnosť odhadu ceny pri nesprávnom monitorovaní môže byť +/- 600% je dosť silný argument [2].

Záver

Monitorovanie softvérového produktu za každých okolností odhalí pravé srdce softvérového produktu. Ako manažéri monitorovania dbajme na udržiavanie sŕdc projektov čo načistejších, veď svet nám ponúka toľko veľa možností ako to dosiahnuť.

Použitá literatúra

1. Chaitanya Kothapalli, S. G. Ganesh, Himanshu K. Singh, D. V. Radhika, T. Rajaram, K. Ravikanth, Shrinath Gupta, and Kiron Rao. 2011. Continual monitoring of code quality. In *Proceedings of the 4th India Software Engineering Conference* (ISEC '11). ACM, New York, NY, USA, 175-184.
2. Chen, Z.; Menzies, T.; Port, D.; Boehm, D.; , "Finding the right data for software cost modeling," Software, IEEE , vol.22, no.6, pp. 38- 46, Nov.-Dec. 2005
3. Johannes Bohnet and Jürgen Döllner. 2011. Monitoring code quality and development activity by software maps. In *Proceedings of the 2nd Workshop on Managing Technical Debt* (MTD '11). ACM, New York, NY, USA, 9-16.
4. Masateru Tsunoda, Tomoko Matsumura, and Ken-ichi Matsumoto. 2010. Modeling Software Project Monitoring with Stakeholders. In *Proceedings of the 2010 IEEE/ACIS 9th International Conference on Computer and Information Science* (ICIS '10). IEEE Computer Society, Washington, DC, USA, 723-728.
5. Mike Ross. 2005. Parametric Project Monitoring and Control: Performance-Based Progress Assessment and Prediction. Galorath Incorporated. El Segundo.
6. Otterholt, Barry. Why Projects Are Late. http://stoufferco.blogspot.sk/2008/09/why-projects-are-late_24.html

Annotation

The Importance of a Heart and a Cover of a Software Product

Can you imagine the situation when a project fails? Especially in software development is a key criterion don't let the project fails. Management of monitoring takes care of that and can determine not only the current status of the project with methods of a development control and special metrics, but also predict the progress of the development and cost of the development and resources required for its completion. You can monitor the status of internal and external development. An internal monitoring is meant quality of code that ensures the integrity, stability and robustness of the system. External state are completed external visible and functional properties of the product. We will look at methods and monitoring capabilities for both internal and external aspects of the development and its impact on the resulting software product.