Monitorovanie a riešenie kríz v softvérovom projekte: analyticky či empiricky?

JAKUB ŠIMKO

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
sharak[zavináč]post[.]sk

Abstrakt. Softvérové projekty vo väčšine prípadov končia neúspechom. Nečitateľnosť prostredia a podmienok v tejto oblasti je vyššia než v ktorejkoľvek inej. Manažment v IT je kritický jednak v čase plánovania, ale aj počas priebehu projektu. Práve rozpoznávanie a riešenie akútnych kríz často zlyháva. Manažér sa pritom môže oprieť o rôzne softvérové nástroje monitorujúce priebeh projektu a uľahčujúce detekciu problémov a rozhodovanie. Môže tak isto vsadiť na empíriu a využiť skúsenosti kolegov v minulosti sa ocitnuvších v podobnej situácii, za použitia scenárov. Sú tieto metódy naozaj nepostačujúce, alebo ich manažéri len nesprávne využívajú? Nezabúdame na psychológiu jednotlivca v kríze? Ako východisko sa javí syntéza rôznych prístupov.

Neúspechy softvérových projektov

Iba jedna šestina softvérových projektov sa končí úspešne, teda sa pri splnenej funkcionalite zmestí do svojho prideleného času a prostriedkov [1]. Čo je však dôvodom takejto nelichotivej miery neúspechu? Vývoj softvéru je pohybom v ťažko predvídateľnom prostredí. Plánovanie projektu samo o sebe nezaručí úspech. Plán je takpovediac „nutnou podmienkou“ úspechu, pretože poskytuje základný kurz a oporu pre vývojárov. Bez efektívneho vedenia však nemá šancu na úspech, pretože ho bude treba zmeniť hneď v prípade prvej situácie s ktorou sa nepočítalo.

Aj pri dobrom pláne je manažment počas behu projektu kritický. Vznik nepredvídanej situácie môže mať fatálne následky a záleží na okamžitých rozhodnutiach. Práve tu sú časté zlyhania manažérov. Vznikajú stresové situácie a je ľudské podliehať panike, ktorá plodí rozhodnutia často ešte zhoršujúce daný stav [3]. Dôležitým faktorom je tiež, ako zavčasu bol problém v rámci projektu rozpoznaný. Včasne rozpoznané problémy mávajú spravidla menší dopad, pokiaľ sú riešené v ešte v zárodku.

Úspešné sa vyrovnanie sa s neočakávaným problémom si teda vyžaduje jeho včasnú identifikáciu a podniknutie správnych krokov na jeho odstránenie. Prvý

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-8.

moment záleží na kvalite monitorovania priebehu projektu, druhý na metodike riešenia kríz softvérovej spoločnosti (ak existuje). Nedostatky v obidvoch fázach majú za následok neúspechy. Súčasné trendy indikujú, že v tejto oblasti sú naozaj rezervy. Je nejaká možnosť posunúť túto problematiku ďalej? Domnievam sa že áno.

Výpočet verzus skúsenosti

Dotknime sa najskôr všeobecnejších konceptov. Konkrétne známeho súboja teórie s praxou. Na úvod najskôr uvediem príklad matematika vyrábajúceho drevený rám na obraz. Jeho skúsenosti s drevom sú mizerné, ale zato pri práci bohato využije svoj matematický aparát, vymyslí si vlastný dizajn, prepočíta každý uhol a vzdialenosť, vytvorí výkres. Nanešťastie však rám skonštruuje za cenu pár zničených nástrojov a za spotreby dvojnásobného množstva materiálu. Na druhej strane tesár vyučený praxou zvládne bez problémov (a z hlavy) desať rôznych rámov, prácu s nástrojmi má v malíčku a drevo reže od oka. Technický výkres však nakresliť nedokáže a nové rámy nevymyslí.

Príklad ilustruje, ako sú praktické zručnosti a generický potenciál teórie vzájomne komplementárne. Kde jedno zlyháva druhé môže excelovať. Platí to v dobre preskúmaných oblastiach a domnievam sa, že tak isto v softvérovom inžinierstve, resp. manažmente softvérových projektov.

Ideálna predstava

Aj pre túto doménu sa snažíme vyvíjať softvérové nástroje čo by prácu robili za nás. Expertné systémy analyzujúce proces vývoja, v ideálnom prípade poskytujúce na výstupe sekvencie krokov čo treba bezprostredne vykonať, berúce do úvahy všetky okolnosti, produkujúce správne výsledky v každom špecifickom projekte. Generické, pre všetko. Manažment vykonávaný ľuďmi by sa pri nich zúžil na plánovanie.

Nateraz utópia. Nemáme potrebné výpočtové prostriedky, ani teoretické modely. Jednoduchšie veci však dokážeme. Pomocou softvéru organizujeme projekt, evidujeme jeho pokrok, ohodnocujeme kroky. Ide o množstvá dát a stroj výrazne uľahčuje ich prehľadávanie, dokonca umožňuje vyhľadávať anomálie, ktoré môžu indikovať problémy. Stopercentne sa však o tieto nástroje opierať nemôžeme.

Na druhej strane si môžeme predstaviť ostrieľaného manažéra, ktorému stačí pohľad do dokumentácie, polhodinka rozhovorov s členmi vývojového tímu a sú mu jasné kľúčové nedostatky a časované bomby. Keďže už podobné situácie mnohokrát zažil, nestratí hlavu a chladnokrvne podnikne všetky možné kroky k náprave.

Asi už menšia (ale stále) utópia. Verím, že jednotlivec môže za svoj život nabrať dostatočné skúsenosti pre prácu v nejakom „prehľadnejšom“ odbore. Oblasť softvérového inžinierstva sa však rozvíja príliš prudko, takže manažér v pokročilom veku, hoci na tom so skúsenosťami bude veľmi dobre, nebude tak dynamický v konfrontácii s najmodernejšími technológiami, podobne ako v situáciách v akých sa ešte neocitol. Nedostatkom je tiež veľmi malý počet takýchto ľudí. Ich prítomnosť v projekte je priam luxus.

Pokúsme sa teda nahradiť zlyhávajúce úsilie jedného prístupu tým druhým. Pozrime sa ako by sa táto syntéza mohla uplatniť v oblasti monitorovania a riešenia kríz v priebehu softvérového projektu.

Rozpoznávanie kríz

Pôvodný spôsob

V minulosti, nielen v oblasti softvérového inžinierstva, nebol projekt monitorovaný strojovo nijak alebo len minimálne. Analýzu priebehu vykonávali samotní manažéri nad množstvom správ podriadených, inšpekcií, doterajších výsledkov práce a spotrebovaných zdrojov.

Už na prvý pohľad sú zrejmé nevýhody. Množstvo informácií sa môže aj skúsenému manažérovi vymknúť z rúk a ľahko prehliadne nebezpečenstvo. Nakoniec väčšina dodaných materiálov predsa obsahuje pozitívne, resp. pozitívne ladené správy. Pracovník ako človek má prirodzené sklony prezentovať svoju činnosť skôr v pozitívnom svetle [4]. V takomto balaste je zdĺhavé a ťažké hľadať indikátory chýb.

Nemali by sme však túto metódu principiálne zavrhnúť. Ak by sa naozaj zamestnanec pracujúci odhodlal z akéhokoľvek dôvodu opísať problém o ktorom vie, komunikáciou prostredníctvom voľného textu, prípadne reči oveľa adresnejšie opíše problém a zasadí ho do aj do správneho kontextu. Tým nesporne ušetrí manažérovi prácu na identifikácii tohto problému. Každý manažér by si prial (aspoň) takéhoto zamestnanca. Ako však takých vychovať je skôr otázka pre psychológov.

Podpora softvéru

Je celkom prirodzené, že množstvo práce pri monitorovaní priebehu projektu sa manažéri snažia redukovať softvérovými riešeniami. Ide o komplexné systémy zahŕňajúce aj nástroje na plánovanie, komunikáciu a ďalšie činnosti súvisiace s manažmentom projektu, okrem iného aj monitorovanie priebehu, plnenie úloh a podobne.

Práve v rámci monitorovania s nástupom softvéru odpadá obrovská záťaž mechanickej práce pri „ručnej“ analýze správ. Napriek tomu že mnohokrát stále ostáva tradičný spôsob písania správ zamestnancami, tieto už majú len „poradný“ význam. Primárnym zdrojom informácií o priebehu projektu od vývojárov sa stávajú nimi vypĺňané, štruktúrovanejšie, elektronické formuláre. Tieto informácie sa dajú lepšie strojovo spracovať. Dajú sa napríklad agregovať do správ za dlhšie obdobie. Môžu byť konfrontované s existujúcim plánom a podobne. Záleží už na metódach a metrikách konkrétneho systému.

Práve voľba metriky je najčastejšou otázkou pri vyhodnocovaní automaticky zbieraných informácií v systéme kontroly softvérového projektu. Medzi jednoduchšie a v mnohom intuitívne patrí počet riadkov vytvoreného zdrojového kódu či počet odpracovaných osobohodín. Tieto sa dobre merajú. Dajú sa konfrontovať s naplánovanými úlohami a k nim priradenými odhadovanými hodnotami týchto

metrík. Na druhej strane môže byť ich použitie krátkozraké, nemusia a často ani neodrážajú skutočné úsilie vynaložené na projekt. Ich kladné výsledky nemusia byť v korelácii s úspešnými testami modulov alebo s pozitívnou reakciou budúceho používateľa (tie však nie sú neustále k dispozícii a aktuálne) [4].

Aby mohol podobný systém pracovať, vyžaduje sa definícia cieľov, resp. úloh, ktorých plnenie sa dá hodnotiť. Tomu predchádza podrobné plánovanie. Úlohy by mali byť rozdelené až na úroveň jednotlivcov, najmä pre jednoznačné rozdelenie zodpovednosti.

Nemusí a ani nemôže ísť o kompletný podrobný plán celého projektu už od začiatku. V prípade dnes často používanej iteratívno-inkrementálnej paradigmy musí byť naplánovaná a rozpracovaná aspoň bezprostredne nasledujúca iterácia. Detaily už závisia na zvolenom WBS (Work Breakdown Structure) [5].

Čo je výsledkom monitorovania? Sumárne správy a v nich indície o nesprávne fungujúcich zložkách, „podozrenia“. K identifikácií problému ešte kus cesty. Tú už musí prekonať človek. Široký záber systému – pokrytie celého projektu mu dáva dobrý základ. Aj neskúsenému manažérovi poskytne východiskový bod, kde začať hľadať. Potom môže nasledovať analýza doplňujúcich materiálov, ako spomínané zamestnanecké správy. Už bude mať na stole len tie relevantné. S využitím priamej komunikácie s podriadenými potom môže dospieť k identifikácii problému oveľa rýchlejšie. Bohužiaľ ani dodané podklady často krát nepomôžu, rozhodujúce sú skúsenosti.

Príklad: Analýza dosiahnutej hodnoty

Jednou z metód kontroly priebehu projektu je aj Analýza dosiahnutej hodnoty („Earned Value Analysis“ - EVA). Ide o prístup s využitím metriky spotrebovaných prostriedkov na vykonanie jednotlivých úloh. Každý segment v projekte (úloha, softvérový modul) je ohodnotený v jednotkách ceny (či už ide o cenu vynaloženého času alebo materiálu). Prvou hodnotou je plánovaná hodnota (PV – planned value), teda aká je očakávaná cena vývoja daného segmentu. Ďalej sa určuje dosiahnutá hodnota (EV – earned value), ktorá je vyčíslením nakoľko si dokončenú prácu na segmente ceníme. Poslednou je aktuálna hodnota (AC – actual cost), vyjadrujúca cenu, ktorú sme v skutočnosti za segment zaplatili. Ukazovatele sa prirodzene dajú sčítať v rámci skupín segmentov tvoriacich väčšie úlohy alebo moduly [2][5].

Vzájomné vzťahy medzi hodnotami napovedajú o efektivite využívania zdrojov v projekte. Takýto prístup má výhodu jasnejšej súvislosti s dvoma z troch známych indikátorov kvality projektu: cenou, vynaloženým časom [5]. Spomedzi možných metrík (a s nimi súvisiacich metodológií vyhodnocovania) sa mi táto javí ako najschodnejší kompromis jednoduchosti ohodnocovania a reálnou výpovednou hodnotou.

Vývoj EVA

Podobne ako samotné paradigmy procesu vývoja softvéru (od vodopádu až po agilné metódy) aj EVA podstúpila paralelne s nimi vývoj.

Prvá forma využívala WBS orientovanú na komponenty systému. Jednalo sa o strom subsystémov a modulov dostatočne rozvitý na malé komponenty. Na tie a spätne na celý strom sa potom aplikovalo vyhodnocovanie. Výhodou bolo, že ak bol WBS dostatočne podrobný, dalo sa pomerne presne určiť, ktoré súčasti zlyhávajú, resp. kde prekračuje rozpočet a podobne. Podstatná nevýhoda bola nutnosť mať celý strom pripravený pred spustením projektu. Vec možná jedine pri vodopádovom modeli.

Ako reakcia na tento problém sa neskôr začala používať EVA orientovaná na proces vývoja. Prvkami WBS stromu sú v tomto prípade úlohy namiesto modulov. Takýto strom môže podliehať zmenám aj v priebehu projektu. Zavádzajú sa tu dva druhy vrcholov: rozpracovaná a plánovaná úloha. Podrobne musia byť popísané len rozpracované úlohy, tie ktoré práve prebiehajú alebo už prebehli. Plánované úlohy sú naopak známe len rámcovo, konkrétnejšiu podobu získajú až v budúcich iteráciách. V strome WBS tvoria plánované úlohy často ešte nerozvité vetvy. Aj plánovaným úlohám sa však dajú priradiť aspoň odhadované hodnoty v rámci EVA metrík.

S nástupom riadenia softvérového procesu prípadmi použitia sa tieto dostali aj do EVA. Vrcholmi stromu WBS sa stali prípady použitia. Keďže prípady použitia vychádzajú predovšetkým z funkcionálnych požiadaviek, zväčšila sa týmto výpovedná hodnota o situácii v projekte aj v poslednom z troch smerov (okrem ceny a času), o kvalite produktu samotnej [5].

Stačí to?

Je kombinácia človeka a stroja v dnešnej podobe dostatočná na rozpoznávanie problémov v procese tvorby softvéru? Trendy úspešnosti projektov naznačujú, že nie, aj v prípade že by sme prijali tézu, že zlyhania spôsobuje skôr nesprávny postup riešenia zistených problémov. Softvér na podporu riadenia projektu sa postupne vyvíja a zlepšuje, v dohľadnej dobe však z tohto smeru zázrak čakať nemôžeme. Zlepšenie ľudského faktoru má zdanlivo jednoduchý recept: naučiť manažérov lepším praktikám. Jednou z možností výučby sa zaoberá aj druhá časť práce.

Riešenie kríz

Kríza, resp. problematická situácia v projekte je vždy jedinečná, to vyplýva už z jedinečnosti projektu ako takého. Iste, môžeme pristúpiť k zovšeobecneniu prípadov a vytvoreniu kategórií a typov typických kríz. Rovnako môžeme problematický stav projektu definovať implicitne. Mohli by sme vytvoriť zložitý stavový model projektu v ktorom by manažérske rozhodnutia tvorili prechodové hrany. Za pomoci takéhoto modelu by sme strojovo interpretovali všetky možné varianty vývoja situácie, našli by sme tú najlepšiu možnú a tú použili.

Takýto model však pre potreby manažmentu softvérového projektu ešte dlho mať nebudeme. Príliš zložité, nerealizovateľné. Napriek tomu existujú niektoré modely použiteľné už dnes, ktoré pri procese rozhodovania v čase krízy aspoň asistujú.

Bayesova sieť

Jedným z modelov na podporu rozhodovania je Bayesova sieť (BS), koncept známy už z osemdesiatych rokov, avšak výpočtovo uskutočniteľný až v dnešnej dobe. BS je orientovaný graf spolu s priradenou množinou tabuliek pravdepodobností. Vrcholmi sú premenné (známe aj neznáme) a hranami vzťahy medzi nimi. Pomocou BS možno vytvárať predikcie dôsledkov určitých krokov. Pri riešení problému to má význam pri rozhodovaní sa medzi viacerými možnosťami ďalšieho postupu. BS dobre zvládajú prepočet medzi cenou, časom a kvalitou produktu. Vyčíslia napríklad o koľko ochudobníme zvyšné dve pridaním do tretej [4].

Nevýhodou Bayesových sietí je, že sú vytvárané ľuďmi a na základe skúseností. Nečrtá sa teda sľubnejšia perspektíva ich využitia v zložitejšom modelovaní a podpore rozhodovania. Na druhej strane, je to aspoň niečo. Riešenie kríz je inak drvivou väčšinou na pleciach manažérov.

Bludný les

Manažér s krátkou praxou je po odhalení problému v projekte, ktorý vedie, v nezávideniahodnej situácii. Jasne si uvedomuje dopad následkov, oveľa nejasnejšiu predstavu má o vhodnom riešení. Chce sa oprieť o svoje teoretické znalosti, ale v strese toho nie je schopný. Začne prvým chybným krokom [1][3].

Ako naberá skúsenosti, príde konečne prvý úspešný projekt. Neskôr už je zlyhanie skôr výnimkou. Skúsenosti získané ohňom. Za cenu neprimeraných strát na neúspešných projektoch. Dá sa tento drahý spôsob nejako obísť?

Využitie scenárov – cesta z bludného lesa

Kevin C. Desouza predostrel vo svojom článku „Scenario Management: From Reactivity to Proactivity“ myšlienku využitia scenárov ako prostriedku na prenos „know-how“ skúsených manažérov služobne mladším kolegom. Inšpirovaný využívaním v iných oblastiach, navrhuje autor ich aplikáciu aj v manažmente softvérových projektov.

Scenár vystupuje ako modelová situácia s otvoreným koncom. Opisuje jej typický, často sa opakujúci priebeh, riešený už mnohokrát. Scenár najskôr opisuje východiskový stav, napríklad projektu. Následne poskytuje sekvenciu krokov – odporúčaný postup pre riešenie podobnej situácie. Scenár je písaný pre ľudí, jeho obsahom je zvyčajne štruktúrovaný text. Môže sa v ňom vyskytnúť aj polemika, prípadne alternatívy k hlavnému postupu.

Desouza navrhuje tvoriť scenáre za účasti skúsených manažérov softvérových projektov. Vytvoril by sa ich dostatočný počet tak, aby dostatočne pokryli špecifické situácie. V krízovej situácii potom poskytnú manažérovi oporný bod a čo je hlavné, upokoja ho. Stresovú situáciu zvládne už preto, že bude mať pocit že ju môže zvládnuť. Scenáre sa dajú navyše využiť aj pri výučbe a tréningu manažérskych zručností [3].

Osobne ma zaujala diametrálna odlišnosť scenárov (ktoré, ak to preženieme, stačí mať niekde na papieri) a analytických pokusov o predikcie a podporu rozhodovania. Scenáre sa v tomto smere ukazujú stále ako silnejší nástroj. Na druhej strane nezaškodí pokiaľ budú rozhodnutia vytvorené na ich základe dodatočne konfrontované s výsledkami predikcií. Nezhody budú využiteľné pri korekciách použitých metód, súhlas potvrdí správnosť oboch (pokiaľ samozrejme úspešne dopadne aj projekt).

Syntéza

Ukazuje sa, že snaha presunúť všetko úsilie v manažmente projektov na plecia softvérových nástrojov je predčasná. Na druhej strane zostávať v časoch, keď softvérová podpora neexistovala je priam hlúpe. Syntéza toho najlepšieho čo nám súčasný potenciál ľudskej mysle a výpočtových technológií dáva je jasným cieľom. Sila ľudského mozgu v softvérovom inžinierstve stále rozhodujúca, potrebnými skúsenosťami ale nedisponuje každý. Sústreďme sa preto najmä na vyvinutie rýchlejších a efektívnejších spôsobov ako sami seba učiť. Tvorivý vek človeka je nanešťastie zapratávaný činnosťou vedúcou k osvojovaniu si už existujúceho. A keď už existujúce ovládame, je už zvyčajne neskoro, lebo tvorivosť sa vytratila.

Použitá literatúra

1. ADDISON, Tom, VALLABH, Seema. Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers.. In *Proceedings of SAICSIT 2002*. [s.l.] : [s.n.], 2002. s. 128-140.
2. BOEHM, Barry, HUANG, LiGuo. Value-Based Software Engineering: Reinventing "Earned Value" Monitoring and Control. In *Software Engineering Notes vol 28 no*. [s.l.] : [s.n.], 2003. s. 1-7.
3. DESOUZA, Kevin C. Scenario Management: From Reactivity to Proactivity. *IT Pro* [online]. 2005 [cit. 2008-10-09], s. 42-48.
4. FENTON, Norman, et al. Making Resource Decisions for Software Projects. In *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*. [s.l.] : [s.n.], 2004.
5. LI, Jinhua, MA, Zhibing, DONG, Huanzhen. Monitoring software projects with earned value analysis and use case point. In *Seventh IEEE/ACIS International Conference on Computer and Information Science*. [s.l.] : [s.n.], c2008. s. 475-480.

Annotation

Monitoring and crisis-solving in software project: analytical or empirical?

Paper brings a look on the field of detection and solving the problems that appear during software projects. Several examples are discussed. Paper focuses on scope of confrontation of human approach with automated analytical methods for project support.