TECHNIKY ODHADOVANIA V PLÁNOVANÍ SOFTVÉROVÝCH PROJEKTOV

Najlepšia technika neexistuje, iba kombinácia všetkých.

Dávid Pszota

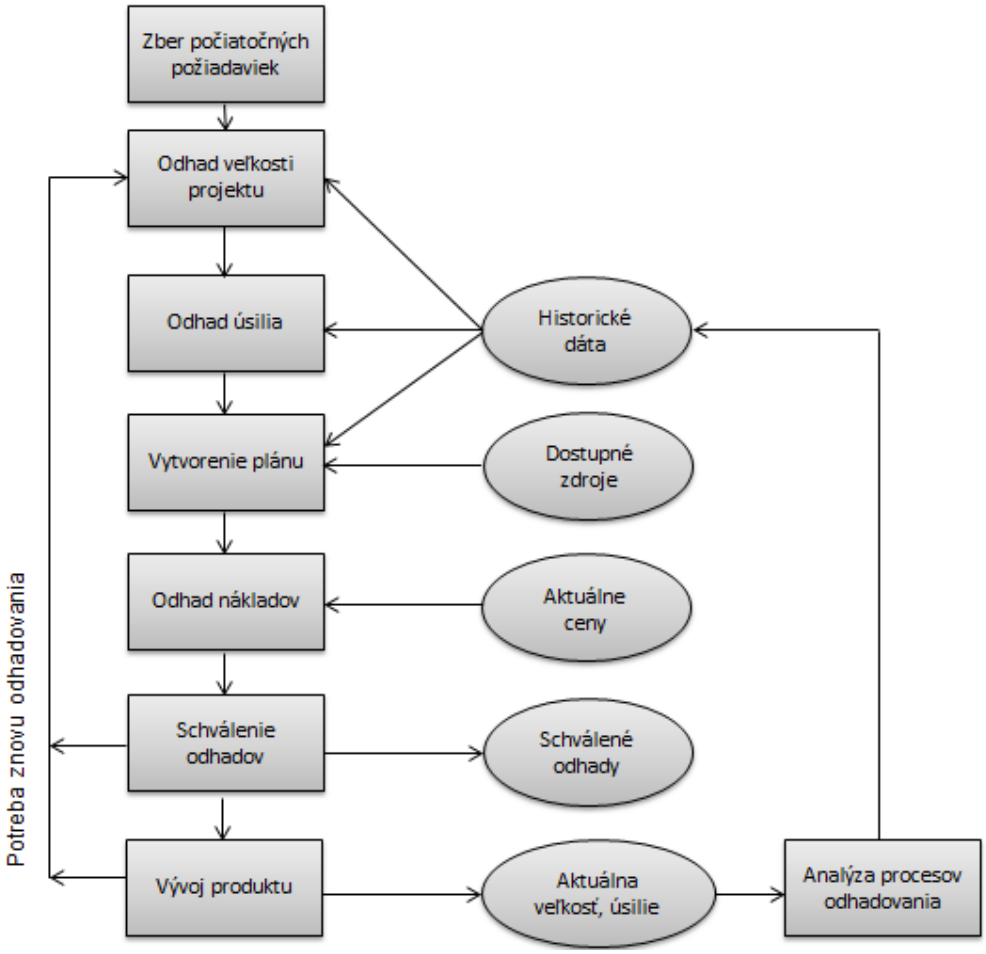
Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
david.pszota[zavináč]gmail[.]com

Abstrakt. *V softvérových projektoch, rovnako ako v projektoch z iných oblasti sa používajú plány. Pre riešenie problémov so správnym odhadnutím dĺžky trvania projektu ako aj pre samotné vytvorenie plánu už existujú známe metodológie. Táto esej sa zameral na najdôležitejší aspekt plánovania softvérových projektov, na odhadovanie. Zaoberá sa s technikami odhadov, identifikuje ich výhody a nedostatky; definuje ich použiteľnosť pre rôzne typy projektov. Pokúša sa pritom nájsť ideálnu kombináciu známych metód na zvýšenie presnosti a efektívnosti odhadu. Pristupuje k problému aj z pohľadu pracovníkov na danom projekte a uvádza potrebu diskusie medzi manažmentom a vývojovým tímom. Zdôrazňuje ďalej nutnosť analýzy historických dát pre využitie v parametrických modeloch a taktiež potrebu využitia súčasných výsledkov z progresu vývoja pre zjemňovanie a revízie plánu projektu.*

Kľúčové slová: *softvérový projekt, plánovanie, odhadovanie, komunikácia, historické dáta*

Procesy plánovania softvérových projektov

Plánovanie softvérové projektu je najdôležitejším aspektom riadenia vývoja projektov. Všeobecný postup vytvorenia plánu softvérového projektu je znázornený na **Obr. 1**. Kde môžeme vidieť, že veľké množstvo potrebnej práce na vytvorenie plánu softvérového plánu zaberá odhadovanie veľkosti projektu, potrebného úsilia a nákladov [1].



Obr. 1 Základe procesy odhadovania softvérového projektu.

Odhadovanie je najcitlivejšia časť plánovania, jednoducho si môžem predstaviť situáciu, keď sa na úlohu vyčlení málo času a tým sa vývojový tím alebo programátor dostáva do časového stresu, kedy je náchylnejší na chybovosť. Opačným prípadom je, keď sa čas na úlohu precení a nastane situácia, kedy pracovník skončí danú úlohu výrazne skôr. Toto má dopad na efektivitu daného pracovníka, či celého tímu, pretože napriek tomu, že pracovník vykazuje prácu na úlohe v skutočnosti je nevyužitý a teda sa plytvá zdrojmi určenými na projekt.

Odborná literatúra predstavuje rôzne techniky odhadovania, pritom uvádza aj to, že neexistuje najlepšia alebo najhoršia technika. Pre zvýšenie presnosti odhadov, treba používať viac techník súčasne. Najtriviálnejší spôsob by bolo vykonanie čím viac metód a následne vypočítať priemer, ale v konečnom dôsledku to by bolo cenovo a časovo veľmi náročné.

Ako rozbiť projekt na rozumné kúsky

Dekompozícia projektu sa robí ako prvý krok vytvorenia plánu, ktorá je aj podmienkou pre úspešné vytváranie celého plánu vývoja.

S *analogickou metódou*, kde vychádzame zo znalostí už dokončených projektov, manažér plánovania jednoduchšie môže určiť vhodnú dekompozíciu projektu. V prípade, keď manažér má vhodné schopnosti a potrebné skúsenosti, môže úspešne posúdiť jednotlivé časti projektu, ale môže nastať situácia, keď jeho odhad bude veľmi subjektívny.

Dokonalejšie metódy, ako *technika zhora nadol*, vyžaduje určenie funkčných a nefunkčných požiadaviek. Podľa môjho názoru tento prístup je úspešnejší vtedy, keď projekt je už v začatom štádiu, lebo tento odhad je pomerne abstraktný na začiatku a presnosť sa zvyšuje počas celého procesu. Možné riziko je aj v podceňovaní nákladov na riešenie ťažkých, nízko-úrovňových technických komponentov. Opačný prístup je *zdola nahor*, ktorá kombinuje všetky jednotlivé prvky a komponenty tak, aby výsledkom bol kompletný odhad projektu. Takýto prístup môže byť presnou metódou, ak systém je navrhnutý detailne.

Podľa mojich predstáv najlepšia technika je kombinácia obidvoch metód tak, aby sa na začiatku vytvorenia plánu projektu vyhotovila predbežná verzia rozbitia projektu na menšie úlohy. Chcel by som zdôrazniť, že veľmi dôležité je, aby manažér plánovania mal skúsenosti v oblasti softvérovej architektúry, v ktorom bude daný produkt vytvorený.

Odhad rozsahu softvérového systému

Pokiaľ je už definovaná štruktúra projektu, môže manažér plánovania odhadnúť celkovú veľkosť projektu. S pomocou parametrických a modelových prístupov je možné definovať veľkosť projektu, napr. počet riadkov kódu (ang. *Lines of code,* ďalej *LOC*) čo je vlastne dĺžka textu programu. Nevýhodou je obťažnosť pri sledovaní rôznych programovacích jazykov a môže závisieť aj od štýlu zápisu.

Technika funkčných bodov (angl. *function points*) vychádza zo špecifikácie požiadaviek. Rozšírenie techniky funkčných bodov je technika bodov prípadov použitia (angl. *use case points*), obidve techniky sú nezávislé od programovacieho jazyka. Nedostatok týchto metodík je v tom, že potrebujeme mať dokonalú analýzu a návrh produktu, čo môže zabrať veľa úsilia a v skorých fázach projektu takéto špecifikácie sú menej dostupné. Každopádne treba zvoliť techniku tak, aby bola v súlade s typom navrhnutého systému a programovacím jazykom.

Myslím si, že na odhad rozsahu celkového projektu sa dajú najvhodnejšie aplikovať spomínané parametrické techniky, hlavne kvôli možnosti použitia historických údajov. Mnoho výskumných projektov [3] sa už zaoberali s analýzou výstupov monitorovania projektov. Organizácie používajú rozvrhy pracovných činností (angl. *timesheets*) na sledovanie stráveného času pracovníkov, členov vývojového tímu na jednotlivých úlohách. Analýza historických dát z rozvrhov môže byť užitočná, ak zhŕňame aj iné zdroje, ako dáta z logov repozitára zdrojového kódu. Napríklad v situácií, keď pracovník mal úlohu splniť v danom rozsahu, ale so svojou prácou skončil skôr do rozvrhu bude doplniť celý čas, bez ohľadu na to, koľko reálne strávil na dokončením úlohy. Preto spojenie LOC (alebo príslušná metóda na meranie množstva vytvoreného kódu) s analýzou rozvrhov

bude poskytovať oveľa presnejšie historické údaje, ktoré sa potom budú môcť využiť ako parametre v ďalších odhadoch.

Ako nezaťažiť pracovníkov?

Určenie potrebného úsilia je nutné, aby pracovníci nemali pridelený viac úloh, ako reálne zvládnu. Vhodným odhadom úsilia je možne tento problém obísť, ale všeobecne je dokázané, že nie je to také jednoduché.

Z vlastných skúseností viem, že manažéri v niektorých menších firmách nepoužívajú žiadny nástroj na odhad, len určia úlohu, riešiteľa (zodpovedného pracovníka) a dátum dokončenia. Takýto prístup je veľmi kritický a zvyčajne spôsobí sklz v pláne projektu. Netvrdím to, že heuristické metódy nie sú dokonalé, ale podľa môjho názoru odhad expertov musí zahŕňať aj produktivitu daného pracovníka, či reálne dokáže úlohu splniť v stanovenom termíne. Joel Spolsky taktiež spomenul na svojom blogu [4], že *„Len programátor, ktorý pracuje na úlohe môže spraviť odhad. Hociktorý systém, kde manažment píše plán a dodáva to programátorovi je odsúdený na neúspech.“* Zdôvodnil to s viacročnými skúsenosťami v manažmente a vývoja softvérových produktov.

V agilnom vývoji, ako je SCRUM, odhad scenárov (angl. *user stories*) plánovania manažér neurčí subjektívne, ale kolektívnym spôsobom. Myslím si, že každý manažér by mal odhad úsilia a potrebný čas na dokončenie úlohy konzultovať minimálne s pracovníkom, ktorý bude danú úlohu riešiť. Najlepšia je otvorená diskusia, kde sa môžu pripojiť aj ďalší vývojári, ktorí sa v danej oblasti majú skúsenosti. Podobne ako aj metóda DELPHI, ktorá používa odhad viacerých expertov.

Definovanie úsilia (*Effort*) je kritickým krokom, hoci základné modeli odhadovania úsilia sú založené na Putnamovej softvérovej rovnici (1), ktorá nie je najflexibilnejším spôsobom.

$$S = E \times (Effort)^{1/3} t_d^{4/3} \tag{1}$$

Viackrát sa dokázalo aj to, že Putnam SLIM (Software Lifecycle Management) je vhodný len na využitie pri odhadovaní menších projektov [1]. Taktiež faktor prostredia (E) nie je definovaný individuálne, je určený pre celý projekt.

Ďalším použitým modelom je COCOMO, ktorý je založený na bodoch prípadov použitia. Dokonalejšia je jeho druhá verzia, ktorá zahŕňa viac parametrov, ako životný cyklus projektu a prístupy k vývoju softvéru. Pri tejto metóde vidím problém v tom, že pre neskúseného manažéra je príliš zložitou metódou, ktorá zaberie aj veľa úsilia.

Kolobeh plánovanie, vykonanie a revízie

Zostaviť dokonalý plán pre menšie projekty je jednoduché, keď sme si určili všetky metriky správne. Projekty väčšieho rozsahu sú oveľa zložitejšie, ale vytvorenie plánu s použitím dobrej dekompozície je možné prácu manažéra plánovania uľahčiť. Cieľom celého vývojové tímu je splyniť projekt a odovzdať produkt na čas.

Súhlasím s M. Nasirom, ktorý vo svojej práci [1] uviedol, že časté revízie spôsobia urýchlenie práce, a zároveň aj skrátia celkový čas potrebný na odovzdanie produktu.

Podľa môjho názoru najlepší spôsob je taký, že sa revízie plánu robia po každej dokončenej úlohe. Hlavne treba použiť výstup monitoringu, ktorý zmení historické dáta; je lepšie použiť vedomosti získané z práve aktuálneho projektu, ako z niečoho staršieho, ktorý mal aj iné podmienky vývoja, ako napr. zloženie tímu.

Zjemňovanie plánu je možné dosiahnuť s revíziou dekompozície, ako som na to poukazoval aj v kapitole *Ako rozbiť projekt na rozumné kúsky*. Každopádne je potrebné zjemňovanie dekompozície počas práce na projekte, treba zabezpečiť také podmienky, aby zmeny v dekompozícií mali najmenší vplyv na celkový plán.

Nakoniec je vhodný spôsob vytvoriť plán tak, aby pred každým míľnikom ostal rezervovaný čas. Problém tohto prístupu je v tom, že nie pri každej iterácii projektu sa využíva rezervovaný čas efektívne. V situácií, keď vývojár skončí svoju prácu v stanovenom termíne, ostane čas, kedy musí čakať na dokončenie ostatných úloh, ktoré práve meškajú. Preto plán musí byť zostavený tak, aby úlohy, ktoré nie sú priamo závisle od dokončenia predchádzajúcich úloh, boli naplánované tak, aby sa dali posunúť do prázdneho intervalu rezervy. Dobrým „zabitím" času môže byť aj opravenie menej dôležitých chýb (angl. *low priority bugs*) alebo zdokonalenie vytvorených prototypov, aby sa dali opätovne použiť v budúcich projektov.

Ale potom čo povedať zákazníkovi, kedy bude produkt hotový?

Je to aj téma manažmentu komunikácie, ako sa dostať ku kompromisu s objednávateľom. Manažér plánovania nesmie určiť dokončenie projektu sám, vyžaduje sa aktívna komunikácia celého manažmentu ako aj s vývojovým tímom.

Nesmie sa zabudnúť na problém „*90% hotový*", čo zmenená, že v stave, keď z špecifikácie požiadaviek produktu je splnený 90%, potrebné úsilie k ukončeniu celkové projektu je ešte oveľa väčšie, ako 10%. Preto treba uviesť zákazníkovi taký termín dodania produktu, aby mal vývojový tím vhodnú rezervu pre dokončenie posledných etáp vývoja.

Záver

Určenie potrebného času a úsilia, teda plánovania pri vykonaní daného softvérového projektu bez samotného odhadovania je vlastne hazard, totiž bez odhadovania ide iba o typovanie potrebného času a úsilia a ako vieme, čas sú peniaze, ktoré asi nechceme míňať hlúpo. Samotný proces plánovania potrebuje významné množstvo úsilia v začiatočných etapách projektu, kedy sa vytvára plán projektu. Preto treba dobre využiť vyhradené zdroje od začiatku a počas vykonania projektu sa nesmie zabudnúť na potrebné zjemnenie plánu.

Podľa môjho názoru metódy spomenuté v obdobnej literatúre, sú podrobne opísane a majú identifikované aj svoje využitie, pre ktoré typy projektov sú vhodné. V mojej eseji som sa pokúsil poukázať na to, že s pomocou kombinácie známych techník odhadovania sa dá vykonať dokonalý odhad veľkosti projektu a potrebného úsilia. Pritom som sa snažil vybrať metódy tak, aby celkový proces bol efektívny, bez uprednostňovania rýchlosti na úkor kvality.

Snažil som sa upozorniť na to, že pri odhadovaní veľkosti projektu a potrebného úsilia manažér plánovania by mal konzultovať svoje výpočty aj s pracovníkmi, vývojármi,

ktorí budú na danej úlohe pracovať. Z praxi taktiež viem, že nedostatok komunikácie medzi manažmentom a vývojovým tímom môže viesť k sklzu plánu a dodania produktu.

Ďalej som poukázal na to, že akú dôležitú rolu hrajú v procese plánovania analýza historických dát a aktuálne reporty z progresu vývoja.

Použitá literatúra

1. H. Leung, Z. Fan. Software Cost estimation. Hong Kong Polytechnic University. 2002. [Cit. 2011-11-23]. [online ftp://cs.pitt.edu/chang/handbook/42b.pdf]
2. M. Nasir. 2006. A Survey of Software Estimation Techniques and Project Planning Practices. In Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD-SAWN '06). IEEE Computer Society, Washington, DC, USA, 305-310.
3. R. Sindhgatta, NC. Narendra, B. Sengupta: Timesheet assistant: mining and reporting developer effort Proceedings of the IEEE/ACM international conference on Automated software engineering - ASE '10 [1-4503-0116-9] Sindhgatta
4. J. Spolsky. 2007. Evidence Based Scheduling. [Cit. 2011-11-23]. [online http://www.joelonsoftware.com/items/2007/10/26.html]

Annotation

Estimation techniques of software projects planning

This paper describes some estimation techniques, which are used in software project planning and scheduling. It also provides comparison and information about advantages and lacks of the described techniques. Here are given some recommendations for better estimation, including the historical data analysis. There is defined the need of discussion between planning management and the members of the developer team.