ODHADY V SOFTVÉROVÝCH PROJEKTOCH V MALÝCH FIRMÁCH

Každý kto má za sebou viac softvérových projektov, by mal na základe týchto skúseností vedieť spraviť odhad na iný softvérovy projekt.

Róbert Šopinec

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

Abstrakt. *Presadiť sa na rozrastajúcom trhu so softvérom je pre malú novovzniknutú spoločnosť náročná úloha. Kľúčovým momentom sa pre ňu stáva správny odhad softvérového projektu, z ktorého následne vypracuje pre klienta zaujímavú ponuku. Preto je dôležité, aby každý úspešný projektový manažér vedel vytvoriť čo najreálnejší odhad softvérového projektu. V tejto práci sa zaoberám odhadmi softvérových projektov, ich klasifikáciou a zároveň porovnaním, či už teoretickým alebo odskúšaným na reálnych projektoch. V eseji sa venujem odhadom veľkosti projektu a úsilia potrebného na dokončenie tohto projektu. V eseji sa zaoberám metódou počtu riadkov kódu, funkčných bodov, modelom COCOMO a expertnými odhadmi (kontrolný zoznam, DELPHI metóda) pre softvérové projekty. Ďalej sa venujem problémom a rozdielom v softvérových projektoch malých firiem (malé softvérové projekty, malý tím ľudí) a väčších softvérových projektoch.*

Kľúčové slová: *odhad, softvérový projekt, COCOMO, expertné odhady*

Prečo a ako odhadovať?

Väčšina ľudských činností sa zakladá na plánovaní. Či už je to varenie čaju, pri ktorom si naplánujeme postupnosť jednotlivých krokov alebo vytvorenie väčšieho diela ako je softvérový produkt. Jedným z najdôležitejších prvkov plánovania je odhad. Pri plánovaní

je dôležité, ba priam až nutné, vedieť odhadnúť trvanie a zložitosť jednotlivých krokov. Tieto odhady sú východiskovými bodmi pre správny návrh plánu. Preto je dôležité dbať na presnosť odhadov. Čím presnejší je odhad, tým lepší bude plán, jednoduchšia realizácia a nakoniec samozrejme kvalitnejší produkt. Snád každý z nás by chcel vytvoriť presný odhad úsilia, ktoré musí vynaložiť na spracovanie projektu. Jednoduhšie je plánovať rozdelenie práce s presným odhadom ako s odhadom, ktorý je nadhodnotený alebo naopak podhodnotený. Avšak niečo ako presný odhad snáď ani neexistuje. Preto si ľudia vymysleli postupy ako svoje odhady čo najlepšie priblížiť k presnému odhadu projektu. Podľa [1] môžu byť postupy softvérových odhadov

1. empirické,
2. analogické,
3. teoretické,
4. heuristické.

Každý z týchto odhadov má svoje výhody a nevýhody. Nedá sa povedať, ktorý odhad je lepší a ktorý horší. Dokonca neexistuje ani univerzálny postup na vytvorenie správneho odhadu. Preto sa veľa prác venuje porovnávaniu rôznych typov odhadov. Skoro vo všetkých prácach je výsledok porovnania softvérových odhadov závislý na veľkosti a type množiny odhadovaných projektov. Ale ako to vyzerá pri odhadovaní softvérových projektov v malých firmách? Môžu si tieto firmy dovoliť odhadovať projekty presne tak ako väčšie už dávno zabehnuté firmy? Alebo existujú aj postupy odhadov softvérových projektov, ktoré sú vhodnejšie pre malé začínajúce firmy? Na tieto otázky sa snažím odpovedať v tejto eseji. Taktiež priblížim fungovanie jednotlivých postupov softvérových odhadov a navzájom ich porovnám. Z porovnania ich výhod a nevýhod (teoretických aj prakticky odskúšaných) môže čitateľ sám vydedukovať, ktorý variant bude pre určenú firmu zaujímavejší.

Odhadovaná veľkosť projektu

Veľkosť projektu je jeden z dvoch hlavných bodov pri odhadovaní softvérového projektu. Každý projektový manažér najskôr odhadne veľkosť projektu a až potom úsilie a zdroje, ktoré na zrealizovanie projektu treba vynaložiť. Existuje viacero spôsobov, ako odhadnúť veľkosť softvérového projektu. Najobľúbenejšia a najpoužívanejšia je metóda počtu riadkov kódu (z angl. line of code, LOC) a metóda funkčných bodov. Metóda počtu riadkov kódu je založená na počte riadkov zdrojového kódu. Metóda funkčných bodov ohodnotí hlavné funkčné body projektu a na ich základe odhadne jeho veľkosť. Obidve techniky podrobnejšie popíšem v nasledujúcich kapitolách.

Metóda počtu riadkov kódu

Metóda počtu riadkov kódu (angl. skr. LOC) sa používa v praxi veľmi často. Množstvo ľudí si to ani neuvedomuje, ale podvedome používajú túto metódu na odhad zložitosti softvérového projektu. Väčšina programátorov si pri odhade veľkosti nejakej časti softvérového projektu hneď začne predstavovať množstvo kódu, ktoré musí naprogramovať. Ak sa ma môj šéf opýta, či je pridanie určitej funkcionality zložité a o

koľko sa zväčší projekt, začnem rozmýšľať nad tým, ako by som túto funkcionalitu naprogramoval. A to je presne odhad softvérového projektu pomocou metódy počtu riadkov. Myslím si, že táto metóda je jedna z najintuitívnejších metód odhadu veľkosti softvérového projektu.

Riadky zdrojového kódu sú esenciálnou zložkou softvérového projektu. Preto si opodstatnene môžme položiť otázku: „Ako inak zmerať veľkosť programu a námahu na neho vynaloženú ako počtom riadkov v kóde?“ Úvaha o zmeraní veľkosti softvérového projektu iba pomocou metódy LOC má jeden háčik. Metóda LOC sa dá celkom presne použiť na odhadovanie menších a jednoduchších projektov. Problém pri zložitejších projektoch je ten, že nie vždy sa dá veľkosť softvérového projektu zmerať iba počtom riadkov v kóde. Okrem počtu riadkov, ktoré naprogramujeme, sa v projekte nachádzajú interné alebo externé dáta. A tieto dáta musíme samozrejme tiež započítať do výslednej veľkosti programu. Preto by bolo optimálne túto metódu používať spolu s inou metódou na odhadnutie veľkosti softvérového projektu. Aj keď je metóda LOC závislá od programovacieho jazyka, stále patrí k najpoužívanejším metódam na odhad veľkosti softvérových projektov.

Ak sa malé softvérové firmy zaoberajú menšími softvérovými projektmi (1-2 ľudia pracujú na projekte najviac 6 mesiacov), postačuje im metóda LOC. Ak sa však obzerajú po väčších projektoch, odporúčam im zvoliť inú metódu alebo kombináciu inej metódu spolu s LOC.

Funkčné body

Metóda funkčných bodov je založená na funkcionalite programu. Funkčné body reprezentujú jednotlivé funkcie programu, ktoré sa dajú rozdeliť do 5 tried [2]. Každej skupine je následne priradená zložitosť od 1 po 3 (1 je najľahšia, 3 je najzložitejšia). Nakoniec sa spočítajú všetky zložitosti funkčných bodov. Oproti LOC má metóda funkčných bodov výhodu, kedže táto metóda priamo vychádza zo špecifikácie softvérového projektu. Na rozdiel od metódy LOC je nezávislá od programovacieho jazyka. Fakt, že metóda funkčných bodov je nezávislá od programovacieho jazyka, môže byť výhoda pre projektových manažérov, ktorý nepoznajú do detailov implementačný programovací jazyk. Projektový manažér nemusí poznať všetky funkcie programovacieho jazyka. Vo väčšine programovacích jazykov stále pribúdajú nové knižnice, ktoré poskytujú väčšiu funkcionalitu na menej riadkov kódu. Projektový manažér nemusí poznať najnovšie knižnice implementačného programovacieho jazyka. Ak však programátor, na rozdiel od projektového manažéra, tieto novšie knižnice pozná, stáva sa odhad veľkosti softvérového projektu nepresný. Vďaka metóde funkčných bodov sa nemusí projektový manažér zamýšlať nad detailnejším naprogramovaním softvérového projektu. Určite je to podstatná výhoda pri tvorbe odhadov väčších projektov.

Ďalší spôsob odhadu podľa funkčných bodov je **výpočet neupravených funkčných bodov (UFC).** Tento výpočet sa realizuje podľa rovnice (1).

$$UFC = \sum_{i=1}^{5} \sum_{j=1}^{3} N_{ij} W_{ij} \tag{1}$$

N znamená počet, teda odhadnutá hodnota funkčných bodov, *W* je váha funkčných bodov, *i* je typ charakteristiky, sumou prechádzame cez všekých 5 typov. Zložitosť funkčného bodu je *j*. Pre výpočet UFC si najskôr zostrojíme tabuľku váh jednotlivých skupín funkčných bodov. Konečný počet funkčných bodov vypočítame prenásobením UFC a faktora zložitosti projektu. Táto metóda je rozšírením predchádzajúcej metódy. Myslím si, že na využitie tohto odhadu sú potrebné určité skúsenosti s odhadovaním softvérových projektov. Menšie softvérové firmy nie sú schopné zaplatiť skúsenejších projektových manažérov, ktorí by vedeli odhadovať pomocou UFC. Pre menej skúseného projektového manažéra bude ľahšie odhadovať veľkosť projektu pomocou metódy funkčných bodov.

Funkčné body použité v praxi

Existujú pravidlá, podľa ktorých sa dajú previesť funkčné body na počet riadkov kódu. Softvérová spoločnosť Quantitative Software Management (QSM) buduje databázu softvérových projektov už od roku 1978. Nachádza sa v nej vyše 7200 kompletných softvérových projektov. Vďaka tomuto veľkému počtu projektov, dokázala spoločnosť QSM odhadnúť vzťah medzi funkčnými bodmi a riadkami zdrojového kódu. V tabuľke 2 sa nachádzajú prevody niektorých jazykov.

Tab. 2 Funkčné body/LOC[5]

Programovací jazyk	Priemer	Medián	Najmenej	Najviac
Access	35	38	15	47
C	148	104	9	704
C++	60	53	29	178
C#	59	59	51	66
HTML	43	42	35	53
Java	60	59	14	97
Javascript	56	54	44	65
PL/SQL	46	31	14	110
SQL	39	35	15	143
Visual Basic	50	42	14	276

Ak dokážeme odhadnúť počet riadkov kódu (LOC) a funkčné body, môžme si jednoducho overiť, či sme v jednej alebo druhej metóde niečo nezabudli. Ak prepočítame odhadované funkčné body na riadky kódu, môžeme toto číslo porovnať s našim počtom odhadovaných riadkov kódu. Ak sa tieto čísla veľmi líšia, mali by sme pouvažovať nad obomi odhadmi. Je viac ako pravdepodobné, že sme v jednom z odhadov niečo vynechali. Samozrejme tento prevod nebude platiť pre každý projekt. Môžu sa nájsť aj projekty, ktoré nebudú spĺňať hodnoty prevodu s číslom v stĺpci najviac alebo naopak s číslom v stĺpci najmenej. Potom sa ale treba zamyslieť nad tým, či sa nedá projekt naprogramovať na menej riadkov.

Odhadované náklady na projekt

Odhady nákladov na softvérový projekt vychádzajú z veľkosti projektu a vynaloženého úsilia. Existuje viacero spôsobov ako odhadnúť náklady na softvérový projekt. Môžme ich odhadnúť analogicky, rozhodnutím expertov alebo pomocou modelov. Do nákladov na softvérový projekt sa započítavajú všetky zdroje potrebné na vyhotovenie tohto projektu. Preto je tento odhad pre firmy najdôležitejší zo všetkých. Hlavnými zdrojomi potrebnými na vyhotovenie projektu sú ľudské zdroje. Každá firma chce vyťažiť svojich pracovníkov na 100 percent. Spoločnou úlohou pre projektového manažéra a tím lídra je optimálne zaradiť programátorov na projekty. Vo väčšine firiem prebieha súčasne viac projektov. Tieto projekty majú rôzne termíny začiatku ako aj termíny odovzdania. Preto je veľmi dôležité odhadnúť množstvo ľudských zdrojov na projekte, aby firma vedela, či sa môže zaujímať o ďalší projekt alebo sú jej kapacity už vyčerpané. Pri nedostatočnom odhade ľudských zdrojov môže projektový manažér zistiť, že pre úspešné dokončenie projektu bude potrebovať viac programátorov. Následne vzniká problém, kde ďalšieho programátora nájsť. Aj keď má firma ešte voľného programátora, ktorý by sa mohol venovať tomuto zle odhadnutému projektu, odhadovaná cena projektu sa navýši, za čo samozrejme zaplatí firma. Preto je dôležité spraviť na začiatku dobrý odhad , z ktorého sa následne vypracuje cenová ponuka. Pri dobrom odhade firma neprerobí, pri zlom odhade sa jej to môže stať. V ďalších kapitolách sa budem venovať jednotlivým odhadom nákladov na softvérových projektoch.

Analógie

Analogické odhady sa zakladajú na porovnávaní už dokončených projektov (alebo ich jednotlivých súčastí) s novým odhadovaním projektom. Na analogické odhady je potrebné mať dostatočný počet už ukončených projektov. Najväčšou výhodou analogických odhadov je, že ukončené projekty už majú presne stanovené náklady, ktoré na ne boli vynaložené. Náš odhad teda vychádza z presnej hodnoty a nie ďalšieho odhadu. Dôležité je vedieť odhadnúť do akej miery je náš projekt totožný z iným, už ukončeným projektom. Určite nenájdeme úplne totožný projekt. Avšak pri väčšej vzorke projektov dokážeme nájisť podobné časti projektov. Aj keď sa tento spôsob zdá celkom dobrý, bude len ťažko realizovateľný pre malú firmu, ktorá zatiaľ pracovala iba na rádovo desiatkách projektov. Riešením by mohlo byť využívanie databázy s väčším (aj keď nie vlastným) množstvom projektov. Pri analogických odhadoch je dôležité určiť podobnosť odhadovanej časti a časti už hotového projektu. Existuje viacej spôsobov ako tento odhad vykonať [4]. Jednou z metód je použitie experta na určenie podobnosti oboch častí projektu. Ak je vo firme veľa ukončených projektov, je táto práca časovo dosť náročná. Medzi programátormi je známe, že sa ani jednému nechce písať dokumentácie k softvérovym projektom. Práve v tomto prípade by sa hodila prehľadná dokumentácia, v ktorej by expert na analogické odhady našiel všetky potrebné informácie. Tiež sa môže využiť interný systém vo firme, v ktorom by sa uchovávali kľúčové informácie o projekte a jeho jednotlivých častiach. Expert by zadal do systému kľúčové informácie o novom projekte a systém by mu odporučil podobné projekty.

Odhady úsilia na projekte pomocou modelov

Matematické modely odhadov sa väčšinou zaoberajú odhadom úsilia potrebného na úspešné dokončenie softvérového projektu. Úsilie softvérových projektov sa zvyčajne meria v človeko-mesiacoch alebo človeko-dňoch. Modely sú vlastne matematické funkcie, ktoré podľa zadaných vstupných parametrov dokážu odhadnúť úsilie. V práci [2] sú popísané vstupné parametre modelov (pomenované cost factors). Modely sa delia na tri skupiny

- Lineárne modely
- Multiplikačné modely (z angl. Multiplicative models)
- Exponenciálne modely (z angl. Power function models)

Súhlasím s výrokom Barryho Boehma „existuje tak veľa nelineárnych vzťahov pri softvérovom vývoji, že nie je možné aby lineárne modely dokázali tieto vzťahy spracovať". Tým pádom majú lineárne modely veľkú nevýhodu oproti multiplikačným a exponenciálnym modelom. V praxi sa asi najviac používajú exponenciálne modely, ktoré sú dobre zmapované a majú podporu vo viacerých softvérových aplikáciach na odhad projektov. Ďalej sa budem venovať exponenciálnemu modelu COCOMO.

COCOMO model

Model COCOMO bol navrhnutý Barrym Boehmom, významným americkým softvérovým inžinierom. Tento model sa zakladá na exponenciálnej funkcii (2), v ktorej a a b sú funkciami vstupných (nákladových) parametrov a S je veľkosť kódu.

$$Úsilie = a \times S^b \tag{2}$$

Existujú dve verzie COCOMO modelu. Základný model, ktorý bol navrhnutý v roku 1981 a COCOMO 2, ktorý je vylepšením starého modelu COCOMO. Pre lepšie značenie a odlíšenie týchto dvoch modelov sa základný model premenoval na COCOMO 81. Ak budete hladať informácie o COCOMO modeloch, tak v literatúre publikovanej pred 1995 (alebo v literatúre kde autori používajú starú konvenciu) nájdete základný model COCOMO. Ak nájdete v literatúre publikovanej po 1995 názov modelu iba COCOMO s najväčšou pravdepodobnosťou ide o COCOMO 2.

Aj keď bol základný model COCOMO veľmi úspešný, preukazoval neskôr nedostatky v odhadovaní novších softvérových projektov. Tieto nedostatky boli prepracované v modeli COCOMO 2. Preto jednoznačne odporúčam používať novší COCOMO model na odhadovanie úsilia softvérového projektu. Novším model COCOMO 2 berie do úvahy aj to, či sa daný odhad deje v analýze projektu, skorom návrhu projektu alebo postarchitektonickom návrhu. Tento model je komplexnejší a berie do úvahy rôzne stavy rozpracovania projektu.

Dôležitým krokom pri používaní tohto modelu je správne odhadnutie (nastavenie) vstupných parametrov. Nie všetci projektoví manažéri musia mať skúsenosti s odhadovaním pomocou COCOMO 2 modelu. Každý má svoju overenú techniku, ktorú používa pri odhadovaní softvérových projektov. Problém nastane vtedy, ak používaná technika zjavne nie je vhodná na odhadovanie špecifického softvérového projektu.

Zoberme si situáciu keď máme techniku odhadov, ktorá produkuje dobré odhady na menšie projekty, ale my sa práve zaujímame o väčší softvérový projekt. Treba si spraviť dobrý odhad tohto projektu, aby sme vedeli vypracovať kvalitnú ponuku. Teraz by sme boli radi, keby sme vedeli odhadovať napríklad pomocu COCOMO 2 modelu, ktorý by nám vypočítal úsilie potrebné na realizáciu projektu. Nasledujúcu situáciu môže projektový manažér vyriešiť dvomi spôsobmi.

Prvá možnosť vychádza trocha z analógií. Projektový manažér si pozrie odhady a požiadavky strarých projektov a snaží sa pomocou nastavenia vstupných parametrov COCOMO 2 modelu získať veľmi podobné odhady ako odhady, ktoré boli spravené na základe inej techniky.

Ak by však odhady pomocu COCOMO 2 modelu robil projektový manažér súčasne s odhadmi pomocou iných techník, priebežne by sa učil pracovať s COCOMO 2 modelom. Projektový manažér, ktorý si priebežne natrénuje techniku odhadu pomocu COCOMO 2 modelu, nebude zaskočený, ak dostane projekt, v ktorom nemôže použiť svoju obľúbenú techniku odhadu. Myslím si, že ak sa projektový manažér naučí odhadovať pomocou modelov (napríklad COCOMO), získal univerzálny prostriedok na odhadovanie softvérových projektov.

Expertný odhad

Expertné odhady nákladov na softvérovom projekte sú jednou z najpoužívanejších techník. Každý, kto vyvíjal softvér, si vie zhruba predstaviť koľko úsilia treba vynaložiť na spracovanie určitého softvérového projektu. Samozrejme, nie každý programátor vie správne odhadnúť veľkosť a úsilie potrebné na dokončenie projektu. Preto tieto faktory odhadujú experti, ktorí majú s podobnými odhadmi bohaté skúsenosti. Týmto som chcel poukázať na to, že v menších firmách môžu byť mladí menej skúsenejší experti na odhady softvérových projektov. Títo menej skúsení experti nemusia poznať žiadnu z techník odhadovania úsilia na softvérovom projekte. Pri odhadovaní veľkosti softvérového projektu som napísal, že metóda počtu riadkov kódu je jedna z najintuitívnejších metód odhadu veľkosti softvérového projektu. Presne to isté si myslím o metóde expertných odhadov vzhľadom na určenie úsilia potrebného na dokončenie softvérové projektu. Každý, kto má za sebou viac softvérových projektov, by mal na základe týchto skúseností vedieť spraviť odhad na iný softvérový projekt.

Expertné odhady sa dajú robiť dvomi spôsobmi. Pomocou kontrolných zoznamov (z angl. checklist) alebo pomocou metódy DELPHI. Ak si spravíme kontrolný zoznam jednotlivých vecí, na ktoré nemôžeme pri odhade zabudnúť, môže sa nám stať, že niektoré prvky zle odhadneme, ale nie že by sme na niečo úplne zabudli. Preto je dobré tento spôsob odhadu kombinovať s iným spôsobom. Samozrejme, že sa to nedá kombinovať s analogickým alebo modelovým odhadom. Jednoducho sa to dá skombinovať s iným expertným odhadom, napríklad pomocou metódy DELPHI.

Pri metóde DELPHI sa stretne viac expertov, ktorí diskutujú o odhade na softvérový projekt. Po diskusii každý napíše svoj subjektívny odhad. Potom sa diskutuje o jednotlivých odhadoch, ktoré experti spravili a vytvárajú sa ďalšie odhady. Toto pokračuje až kým sa experti nedohodnú na spoločnom odhade.

Pokus na expertných odhadoch

Ursula Passing a Martin Shepperd spravili pokus na študentoch z Bournemouthskej univerzity, ktorých najskôr rozdelili do štyroch skupín [4]. Študenti mali za úlohu odhadovať veľkosť (v riadkoch kódu) a úsilie potrebné na dokončenie fiktívneho softvérového projektu. Odhady robili v troch kolách. V prvom kole spravili samostatne odhady na projekte, len na základe podkladov k tomuto projektu. V ďalšom kole boli študentom k dispozícií kontrolné zoznamy. Študenti opravili svoje odhady na základe kontrolných zoznamov a odovzdali ich. V poslednom kole použili študenti techniku DELPHI a vytvorili nové odhady. Experiment bol sice vykonaný na študentoch, ktorí majú ešte ďaleko od expertov, ale myslím si, že aj tak do určitej miery vystihol svoju podstatu.

Výsledok experimetu bol, že väčšina študentov pri odhadovaní veľkosti softvérového projektu s pomocou kontrolného zoznamu svoje odhady navýšila. Z toho celého vyplýva, že neskúsenejší experti majú tendenciu zanedbať určité celky v prvotnom odhadovaní veľkosti softvérového projektu. Prínos kontrolných listov je jednoznačný. Myslím si, že sa oplatí použiť drahocenný čas na vytvorenie kontrolného zoznamu. Lebo ako sa hovorí, čas sú peniaze. Ale v tomto prípade je lepšie obetovať nejaký čas a spraviť lepší odhad, z ktorého budeme môcť neskôr vychádzať.

Po treťom kole bol skoro rovnaký počet študentov, ktorí svoje odhady navýšili alebo naopak znížili. Metóda DELPHI je tiež prínosom. Experti si navzájom porovnajú svoje odhady a každý z nich sa môže na nový odhad pozerať obohatený o myšlienky ďalšieho experta. Každý z nás sme iný, rozdielne rozmýšlame a preto sme jednoznačným prínosom do väčšieho celku.

Pokus preukázal, že odhad úsilia mal podobný priebeh ako odhad veľkosti softvérového projektu. Tento fakt len potvrdzuje myšlienky, ktoré som napísal ohľadne používania kontrolných zoznamov a metóde DELPHI.

Záver

Na koniec by som chcel odpovedať na otázky, ktoré som si položil na začiatku. Odhadovanie projektov v malých firmách je obmedzené kvôli rozpočtu a skúsenostiam odborníkov. Preto, ako som už spomínal v texte, odporúčam pre tieto firmy jednoduchšie metódy pre vytváranie odhadov. Tiež je pravda, že kvalitnejší odhad dostaneme použitím viacerých techník zároveň. Napríklad pri technike počtu riadkov kódu môžeme zabudnúť na nejakú podstatnú časť aplikácie. To sa nám nemôže stať ak spojíme túto techniku spolu s odhadovaním pomocou funkčných bodov. Pri odhadovaní pomocou funkčných bodov vidíme jednotlivé funkcie projektu, ktoré vychádzajú zo špecifikácie a preto nemôžeme na nič podstatné zabudnúť.

Odhadovanie nákladov na softvérový projekt je záverečná fáza nášho odhadu. Tu vidím ako jednu z najlepších metód odhadovanie pomocou analógií. Problém je v tom, že toto odhadovanie nie je vždy možné. Pre malú začínajúcu firmu je to priam nevhodné. Takáto firma nepracovala asi na veľkom počte projektov, takže nemá s čím porovnávať. Samozrejme ak pracuje na podobnom projekte ako v minulosti, nie je sa nad čím zamýšlať. Určite treba použiť metódu analógie na určenie odhadu. Metóda analógie sa dá tiež kombinovať s ostatnými metódami. Pri používaní matematických modelov, ako je

COCOMO, vidím problém v tom, že tieto modely sú zložité a často sa ich musí expert najskôr naučiť používať. Z predchádzajúcich úvah mi vyšlo, že v malej firme sa najlepšie uplatní metóda expertných odhadov. Kombinácia kontrolného listu spolu s DELHI metódou môžu byť silný nástroj aj v rukách začínajúceho projektového manažéra.

Použitá literatúra

1. Bieliková, M.: Softvérové inžinierstvo, 2000.
2. Leung, H., Fan, Z.: Software Cost Estimation, In: *Handbook of Software Engineering and Knowledge Engineering*, Vol. 2, S. K. Chang, World Scientific Publishing Company, 2002.
3. Passing, U., Shepperd, M.: An experiment on software project size and effort estimation, pp.120, 2003 International Symposium on Empirical Software Engineering (ISESE'03), 2003.
4. Shepperd, M., Schofield, C.:. Estimating Software Project Effort Using Analogies. *IEEE Trans. Softw. Eng.* 23,(1997), 736-743.
5. Quantitative Software Management. Dostupné na internete http://www.qsm.com/ [cit: 2009-október]

Annotation

Estimation software projects in small companies

For new and small company is not really easy mission to win recognition in enlarging software market. The key moment for this company is good estimation of software project, from which they will be able to device interesting offering for clients. For this reason is really important for every single successful project manager to know how to create as much real estimation of software project as possible. In this work session I dwell on estimation of software projects, their clasification and comparing, whether of teoretical or real examinated projects. In this esay I evode the method of calculating of code lines, functional points, models COCOMO and expert guesses (checklist, DELPHI method) for software projects. Further I evode to problems and differences in software projects in small companies (small software projects, small working team) and bigger software projects.