Odhadovanie v softvérových projektoch

Keď na veľkosti záleží

JOZEF BEŇO

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

Abstrakt. Odhadovanie softvérových projektov je oblasť, kde je softvérové inžinierstvo stále veľmi neisté. Pri veľkých projektoch je skôr výnimkou, ak sa odhad čo len približne podobá výslednému úsiliu. Pri menších projektoch je situácia lepšia. V práci uvádzam niektoré možnosti zlepšenia odhadovania pre projekty, ktorých členmi sú aj menej skúsení riešitelia. Ide o použitie kontrolného zoznamu, ktorý slúži na pripomenutie dôležitých častí produktu a aktivít projektu, aby neboli vynechané pri tvorbe odhadu rozsahu a potrebného úsilia. Ako druhú spomeniem techniku Deplhi, ktorej cieľom je uľahčiť zdieľanie znalostí medzi členmi tímu, a tak prispieť ku kvalitnejšiemu odhadu.

Úvod

Ako inak začať úvahu o odhadovaní v softvérových projektoch, než hrozivými číslami zo široko citovaných správ Standish Group. Od roku 1995, keď sa podľa ich výskumu v USA zrušilo pred dokončením až vyše 31% softvérových projektov, sa situácia síce zlepšila, ale aj podľa reportu z tretieho kvartálu roku 2004 ich stále zlyhalo nepríjemných 18%. V tabuľke (Tab. 1) sú uvedené štatistiky z niekoľkých rokov (zo zdrojov na stránkach Standish Group: www.standishgroup.com).

(v %)	1995	2000	2004
Úspešné	16	28	29
Problémové	53	49	53
Zrušené	31	23	18

Tab. 1. Vývoj úspešnosti softvérových projektov

Za úspešné projekty sa akceptujú len tie, ktoré boli dokončené načas, za plánovaný rozpočet a obsahujú všetky funkčnosti, ktoré boli požadované. Problémové sú tie projekty, ktoré boli dokončené a odovzdané do používania, ale nesplnili minimálne jedno kritérium úspechu, t.j. boli odovzdané neskôr a/alebo ich vývoj stál viac než bol rozpočet a/alebo neobsahovali všetky pôvodne požadované funkcie.

Manažment v softvérovom inžinierstve, december 2005, s. 1-7.

Zrušené projekty sú tie, ktoré boli zastavené pred dokončením výsledku alebo boli vyradené z prevádzky krátko po nasadení.

V roku 2001 uskutočnila britská počítačová spoločnosť podobný výskum na vzorke viac ako 1000 IT projektov zo Spojeného kráľovstva. Zistili, že až 90% úspešných projektov malo trvanie do 12 mesiacov a 47% trvalo menej ako 6 mesiacov. Ide však o celkové trvanie a nie o rozsah. K podobnému záveru dospeli aj v Standish Group, keď ako optimálne odporúčajú projekt štyroch ľudí počas štyroch mesiacov, t.j. celkovo 16 človeko-mesiacov (ČM).

Aj tieto zistenia podnietili vznik a rozšírenie agilných metód vývoja softvéru. Jedným z ich nosných princípov je inkrementálny vývoj, keď sa funkčnosť do výsledku dopĺňa postupne po menších častiach. Každá jedna časť je vedená ako samostatný projekt, ktorý je lepšie manažovateľný a, ako ukázali štatistiky, aj spoľahlivejší.

Aj pri malých projektoch je však možné zlepšiť odhadovanie. Najviac problémov s odhadom veľkosti projektu a potrebným úsilím majú menej skúsení členovia tímu. Pre zlepšenie ich orientácie v projekte, ktorá vedie aj k zlepšeniu odhadov, sa používajú aktivity, z ktorých uvediem techniku Delphi a použitie kontrolného zoznamu.

Čo je odhad?

Magne Jörgensen v krátkom článku [1] upozorňuje na skutočnosť, že pojem odhadovania sa používa nejasne. Na príklade s plánovaním rozpočtu dovolenky identifikoval tri rôzne typy odhadov:

- Najpravdepodobnejší
- Minimalizujúci riziko
- Minimalizujúci náklady

Najpravdepodobnejší odhad vyjadruje predpokladanú mieru úsilia potrebného na vykonanie projektu. Môže sa získať pomocou rôznych techník alebo analógiou s predchádzajúcimi podobnými projektmi na základe skúseností.

V druhom prípade je najdôležitejšie vziať do úvahy všetky možné riziká spojené s projektom, aby aj v prípade, že došlo k ich uskutočneniu, nebol odhadnutý rozpočet prekročený.

Tretí typ je spojený s cieľom minimalizovať náklady tým, že sa obmedzí rozpočet a účastníci projektu ho musia pri riešení jednotlivých častí brať do úvahy, t.j. musia si „strážiť“ náklady.

Nejednoznačnosť v odhadovaní môže spôsobovať problémy. Často sa stáva, že odhad, ktorý je vyhotovený ako najpravdepodobnejší, chápe zákazník ako odhad, ktorý už má započítané všetky riziká, a z toho usudzuje, že celkové náklady neprekročia poskytnutý odhad.

Problémy s nejednoznačnosťou sa však týkajú aj porovnávania projektov. Ak sa vyhodnocujú odhady vytvorené s rôznymi cieľmi (najpravdepodobnejší odhad,

bezpečný odhad), tak výsledky sú nutne nepresné a neposkytujú relevantné údaje pre ďalší výskum a nevedú ani k spresneniu nových odhadov. Problém sa, podľa Magneho, týka aj najznámejšieho výskumu, vyššie spomenutého reportu Chaos od Standish Group, keď vo vstupných formulároch nie je uvedené, aký typ odhadu majú prispievatelia zadávať. Možno aj preto bolo v ich správe z roku 1995 priemerné prekročenie počiatočného odhadu až 89%.

Odhadovanie pre veľké projekty

Pre veľké systémy boli vypracované rôzne metodiky, ktoré sa odvádzali na základe skúseností z predchádzajúcich projektov. Pomocou určenia veľkosti systému na základe informácií, ktoré sú k dispozícii po fáze špecifikácie a návrhu, sa snažia ďalej odhadnúť koľko úsilia, vyjadreného v človeko-mesiacoch, je potrebné na realizáciu daného projektu.

Tieto metodiky majú definovaný vzťah medzi úsilím a veľkosťou projektu v tvare

$$\acute{U} := a + bV^{c} \tag{1}$$

Kde Ú je veľkosť úsilia a V je veľkosť systému, čo môže byť údaj v počte riadkov kódu (LOC) alebo veľkosť môže byť určená pomocou funkčných bodov (FP).

Hodnoty premenných a, b a c je pre každú metodiku rozdielna. V tabuľke sú uvedené niektoré metódy a hodnoty ich parametrov (Tab. 2).

Model	Faktor b	Faktor c	Veľkosť
Walston-Felix	5.2	0.91	LOC
COCOMO (organic)	2.4	1.05	LOC
Herd		1.06	LOC
COCOMO (semi-detached)	3.0	1.12	LOC
Bailey-Basili	5.5	1.16	LOC
Frederic		1.18	LOC
COCOMO (embedded)	3.6	1.20	LOC
Jones		1.40	LOC
Halstead	0.7	1.50	LOC
Schneider		1.83	LOC
COCOMO II	2.9	1.10	LOC
Albrecht and Gaffney	-13.39	1	FP
Kemerer	60.62	3	FP

Tab. 2. Porovnanie jednotlivých modelov odhadovania [6]

Rozdiely, ktoré sú medzi jednotlivými metódami, vyplývajú z množiny dát, z ktorých sa štatistickými metódami získavali hodnoty parametrov a, b a c. Tento vzťah je ovplyvňovaný množstvom faktorom a je takmer pre každú firmu alebo dokonca projekt osobitý. Určenie veľkosti projektu je zložité a citlivé na nastavenia. Napríklad COCOMO má až 15 „driverov“, ktorými sa opisuje vývojové prostredie.

V metóde Use Case Points [8] sa používa 13 faktorov opisujúcich technickú zložitosť. Je zrejmé, že chybným určením niektorého parametra sa výrazne zmení odhad veľkosti výsledku a teda aj odhad potrebného úsilia. Podľa niektorých výskumov boli pri používaní COCOMO evidované priemerné odchýlky odhadov od uskutočneného úsilia až v rádoch stoviek percent.

Malé projekty

Kathleen Peters z firmy „Software Productivity Center" uvádza v [5], že za malý projekt sa považuje ten, ktorý vykonáva tým veľkosti jednej až dvoch osôb a je naplánovaný na dobu do šesť mesiacov. Malý projekt je tak najviac v rozsahu 12 človeko-mesiacov.

Literatúry, ktorá by sa venovala projektom uvedeného rozsahu, nie je veľa. Čiastočne to môže vyplývať aj z toho, že malé projekty majú lepšie výsledky a je pre ne ťažké uskutočniť dostatočne presné merania. Podrobné zdôvodňovanie odhadov a meranie činností v projekte by malo vplyv na celkové vynaložené úsilie, a teda by nastal, fyzikom dobre známy, efekt, keď meranie ovplyvňuje namerané hodnoty.

V univerzitnom prostredí je možné venovať sa aj malým projektom. Ak projekt trvá jeden školský rok a zúčastní sa ho šesť študentov, je celková veľkosť projektu približne 200 človeko-dní, t.j. je už dostatočne veľký, aby malo zmysel odhadovať jeho veľkosť, resp. jeho náklady. Projektom v tomto rozsahu sa venovali práce [2], [3] a [4], z ktorých budem čerpať niektoré postrehy.

Odhadovanie pre malé projekty

Pre malé týmy je nepraktické strácať čas komplikovanými definíciami rôznych premenných a následnými výpočtami údajov, ku ktorým nemajú dôvod, v konečnom dôsledku, mať dôveru.

Ak by chceli na odhad prácnosti použiť napr. metódu „Use Case Points", ktorú navrhol Gustal Karner, museli by definovať množstvo informácii [8]:

- počet a zložitosť prípadov použitia
- počet a zložitosť hráčov so systémom
- rôzne nefunkcionálne požiadavky (*prenosnosť*, *udržiavateľnosť* atď.)
- prostredie, v ktorom sa projekt vyvíja (*použitý jazyk*, *motivácia tímu* atď.)

Pre menšie projekty by bolo priradenie váhy každému parametru vstupujúcemu do výpočtov veľmi náročné a, v končenom dôsledku, by výsledná hodnota nemusela byť ani približne správna. Chybné určenie váhy pre niektorý parameter (napr. *jednoduchosť používania* alebo *zložitosť výpočtov*) môže výrazne zmeniť odhad veľkosti, a teda aj potrebného úsilia.

Najvhodnejším spôsobom ostáva ten najpriamočiarejší, a to nechať odhad na členov tímu, aby každý zvážil koľko úsilia bude potrebovať na realizáciu projektu.

Pomocou metód pre veľké projekty nie je možné nasimulovať vlastnosti malého projektu a členov riešiteľského tímu. Expertný odhad je pre malé projekty aj najviac odporúčaným spôsobom v literatúre.

Problémy môžu mať menej skúsení vývojári. Najmä pre nich je vhodné použitie niektorých jednoduchých techník, z ktorých sa budem krátko venovať použitiu kontrolných zoznamov a technike Delphi.

Kontrolný zoznam

Kontrolné zoznamy pomáhajú členom tímu nezabúdať na niektorú oblasť riešenia. Zvyčajne sú odvodené z Breakdown Structure a obsahujú najpodstatnejšie aspekty projektu. Zoznam môže byť definovaný produktovo alebo procesne, t.j. môže obsahovať to, čo sa má dodať (*triedy*, *obrazovky*, *dokumentácia*) ale aj aké aktivity sú spojené s vývojom (napr. *tréning používateľov*, *testovanie*).

Najmä pre neskúsených vývojárov je kontrolný zoznam užitočný pri odhadovaní potrebného úsilia, keďže je pre nich charakteristické, že sa sústredia len fázu tvorby kódu (konštrukčná fáza v Unified Process) a na ostatné si nerezervujú čas.

Podľa výsledkov výskumu na univerzite Bournemouth [3] študentov najprv nechali odhadnúť veľkosť a potrebné úsilie. V druhom kole odhadovania im poskytli kontrolné zoznamy pre veľkosť a aj pre proces. Pri procesom zozname použili aktivity definované v Rational Unified Proces.

Výsledok porovnania odhadov v prvom a druhom kole potvrdil predpoklad, že neskúsení, resp. menej skúsení riešitelia zabudnú na niektorú časť produktu alebo procesu. Väčšina študentov zväčšila svoj odhad veľkosti a takmer všetci zvýšili odhad potrebného úsilia, čo svedčí o tom, že v prvom kole venovali málo pozornosti iným aktivitám ako samotné programovanie.

Skúsenejší členovia tímu sú potrební na to, aby zadefinovali obsah kontrolného zoznamu, resp. na to, aby schválili použitie existujúceho zoznamu z podobného projektu.

Metóda Deplhi

Metóda Deplhi je založená na tímovej diskusii, kde si jednotliví členovia tímu vymieňajú názory týkajúce sa projektu. Význam majú najmä poukazovania na možné riziká, kde sa spoločnou diskusiou môže zlepšiť odhad jeho miery a času potrebného na jeho riešenie.

Metóda Delphi je iteratívna metóda a je zameraná na eliminovanie skupinového efektu tým, že odhady sú vykonávané anonymne. Zistilo sa, že je to vhodná metóda na minimalizovanie podľahnutiu väčšinovému názoru. Debata členov je otvorená a nie je v nej vyžadovaný všeobecný súhlas diskutujúcich.

V prípade výskumu [3] nie je jednoznačne možné určiť vplyv metódy Deplhi na odhadovanú veľkosť resp. potrebné úsilie. V niektorých prípadoch novo získané vedomosti umožnili znížiť odhad, keďže niektoré predtým problematické otázky sa v diskusii ukázali byť jednoduchšie na riešenie. Na druhej strane sa niekedy ukázalo, že študenti zabudli na niektoré aktivity alebo časti produktu, a preto výsledkom diskusie bolo zvýšenie odhadov.

Metóda Delphi však mala pozitívny efekt v tom, že riešitelia po diskusii mali bližšie odhady celkového úsilia potrebného na riešenie projektu. Tiež sa zvýšila aj dôvera k vytvorenému odhadu.

Záver

Pre odhadovanie malých projektov je najčastejšou a najkvalitnejšou metódou expertný odhad. Malým tímom, kde ani jeden člen nemá dosť skúseností s odhadovaním, nepomôžu žiadne metodiky, keďže na ich zvládnutie je potrebné mať veľa znalostí a ich výstupy sú veľmi citlivé na vstupné parametre, a teda prakticky nepresné.

V prípade, že sa použije kontrolný zoznam a metóda Deplhi, dôjde k zlepšeniu odhadu, keďže dochádza k zdieľaniu poznatkov. K presnosti odhadov však prispieva aj Parkinsov princíp, podľa ktorého sa práca vždy rozširuje až dovtedy, kým sa nenaplní čas, ktorý je pre ňu vyhradený. Takže, ak expert pri odhadoch aplikuje bežné riadenie rizika tak, že pôvodný odhad vynásobí osvedčenou konštantou, je veľká pravdepodobnosť, že projekt bude ukončený včas a za odhadnuté náklady.

Uvedený spôsob odhadovanie projektov je na Slovensku dominantný a používa sa pri všetkých projektoch, od malých až po tie najväčšie. Súvisí to s bežnou praxou vo firmách, ktoré podceňujú význam úvodných fáz projektu a snažia sa čím skôr dostať k implementácii. Bez dôkladne zvládnutej špecifikácie požiadaviek nie je možné kvalitne odhadovať, keďže ako vraví múdrosť štatistikov: Zo zlých údajov sa zlou metódou môžu získať dobré výsledky, ale ak máme dokonalú metódu, tak zlé vstupy nutne vedú k zlým výstupom.

Použitá literatúra

1. Jörgensen, M.: How Much Does a Vacation Cost? Or What is a Software Cost Estimate. *ACM Software Engineering Notes,* Vol. 28, Issue 6 (Nov. 2003) 30.
2. Johanson, L., Lungdren, M: *Software Project Planning Light: Applying project planning techniques on small software projects.* Växjö, 1999.
3. Passing, U., Shepperd, M.: An Experiment on Software Project Size and Effort Estimation. International Symposium on Empirical Software Engineering 2003 (ISESE'03).
4. Jörgensen, M, Sjöberg, D.I.: Impact of Effort Estimates on Software Project Work.
5. Peters, K.: *Software Project Estimation.* Kathleen Peters, Software Productivity Center Inc., 1999 (www.spc.ca)
6. Johnson, K.: *Software Cost Estimation: Metrics and Models.* Department of Computer Science, University of Calgary.
7. Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. *IEEE Transactions On Software Engineering*, Vol. 23, No. 12 (Nov. 1997) 736 – 743.

8. Cohn, M.: Estimating With Use Case Points. *Methods & Tools*, Vol. 13, No. 3 (2005) 3-13

Annotation

Estimating in Software Projects

For estimating software projects great differences between small ones and huge ones exist. From several studies it is known that huge projects are more likely to fail than small ones. But there are still opportunities for improving estimates. Effort and size estimate accuracy depends on team members' capabilities and therefore less experienced people tend to underestimate their estimate. In this essay I am briefly introducing two techniques, check list and Delphi, which could help them improve their estimates.