ODHADY ÚSILIA V PLÁNOVANÍ SOFTVÉROVÝCH PROJEKTOV

Cieľ bez plánu je len prianie. (Antoine de Saint-Exupery)

Lukáš Zboroň

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

Abstrakt. *Pre úspech akéhokoľvek softvérového projektu je veľmi dôležité, aby bol správne plánovaný. Efektívne plánovanie vyžaduje poznanie jednotlivých častí projektu a ich nárokov na zdroje. Najdôležitejším zdrojom vo väčšine softvérových projektov sú kvalifikovaní pracovníci, a preto treba všetky časti projektu medzi nich vhodne rozdeľovať. Pre tento účel je potrebné na začiatku, ale aj v priebehu celého projektu dokázať odhadnúť približnú dobu riešenia s dostupnými zdrojmi. V súčasnosti existuje viacero rôznych metód, na základe ktorých je možné takéto odhady vykonávať. Aké sú ich výhody a nevýhody? Kedy je vhodné ich použiť? V tejto eseji sa pozriem na niekoľko odlišných prístupov, ktoré je možné použiť pre odhad vyžadovaného úsilia a pokúsim sa zhodnotiť ich prínos pre plánovanie projektu.*

Kľúčové slová: *úsilie, odhad úsilia, plánovanie, rozsah projektu, SLOC, COCOMO, body prípadov použitia, odhad na základe analógie*

Úvod

Takmer každá, aspoň trochu zložitejšia ľudská činnosť, ktorá trvá nejakú dobu, podlieha určitému stupňu plánovania. Plánovanie pomáha znížiť čas potrebný na vykonanie danej činnosti, zvyšuje jej kvalitu, zmenšuje potrebnú námahu a podobne. Pri relatívne jednoduchých činnostiach, na ktorých sa nepodieľa príliš veľa ľudí, je plánovanie zvyčajne skôr neuvedomelé a intuitívne. Ak však ide o skutočne komplexnejšiu činnosť, ktorá bude prebiehať nejakú dlhšiu a najmä ak do nej bude zainteresovaná väčšia skupina ľudí, je potrebné hľadať sofistikovanejšie metódy plánovania.

Väčšina softvérových projektov všetky tieto tri podmienky spĺňa, a preto ich plánovanie musí byť nejakým spôsobom organizované. Nutnou podmienkou každého úspešného plánovania je potrebné čo najpresnejšie poznať ciele, ktoré chceme dosiahnuť, a tiež vedieť, aké kroky nás na ceste za ich dosiahnutím čakajú. Ďalej je potrebné vedieť aké majú jednotlivé kroky nároky na zdroje a aké zdroje máme k dispozícii, resp. aké vieme potenciálne získať.

Prvým a možno najväčším problémom, s ktorým sa pri plánovaní projektu stretávame je teda odhad potrebných zdrojov a úsilia, ktoré budú potrebné na realizáciu projektu. V tejto eseji sa pozriem na niekoľko spôsobov, ktoré sa na tento účel využívajú. Každý z nich má svoje výhody aj nevýhody a výber vhodného spôsobu záleží na type a rozsahu projektu a skúsenostiach a vedomostiach ľudí, ktorí ho realizujú.

V prvej časti som opísal dôvody, pre ktoré sú plánovanie a odhady nevyhnutné, na čo sa zameriavajú a aké sú s nimi spojené problémy. V ďalšej časti som stručne opísal všeobecný proces odhadovania úsilia a následne som opísal aj niekoľko metód používaných pre tieto odhady. Koniec eseje je venovaný môjmu názoru na odhady úsilia v softvérových projektoch a niektoré metódy používané pre tieto odhady.

Potreba plánovania a odhadov

Ak za úspešný softvérový projekt pokladáme projekt, ktorý je dokončený načas pri dodržaní stanoveného rozpočtu a v požadovanej kvalite, softvérové projekty sú úspešné len málokedy. Podľa výskumu spĺňa tieto kritériá len 16,2% projektov. Jednou z najčastejších príčin týchto zlyhaní je zlé alebo nedostatočné plánovanie [3].

Plánovanie je dnes už nevyhnutnou súčasťou všetkých softvérových projektov. Prečo to tak je? Aký je jeho účel a prínos? V prvej fáze projektu ešte pred jeho samotnou realizáciou je najdôležitejšie, že vďaka určitému plánovaniu a odhadom môžu byť so zákazníkom dohodnuté také dôležité otázky ako termíny dodania, cena projektu a čo všetko bude do projektu zahrnuté. Na základe týchto dohôd a možností môže výrobca vyčleniť príslušné zdroje. V ďalších fázach sú plánovanie a odhady dôležité najmä kvôli koordinácii jednotlivých činností a presúvaniu zdrojov medzi nimi. Ak napríklad jeden tím ľudí má pracovať na niečom, čo v istom bode bude potrebovať výsledky iného tímu, ktoré majú byť hotové v istom čase, je potrebné to pri plánovaní zohľadniť. Na to potrebujeme vedieť, kedy približne budú tieto výsledky k dispozícii a kedy budú potrebné. Bez toho by sa mohlo stať, že jeden tím bude zbytočne musieť čakať na výsledky druhého tímu. Týchto ľudí by bolo potrebné dočasne presunúť na inú činnosť a neskôr vrátiť späť. Takéto presuny samozrejme vždy vyžadujú nejaký čas, kým sa tím adaptuje, a to spôsobuje zbytočnú stratu času a v konečnom dôsledku aj zisku [3].

Základným problémom je teda hľadanie odpovede na nasledujúcu otázku: ako rýchlo sme schopní vykonať požadované úlohy so zdrojmi, ktoré máme k dispozícii? Túto otázku nikdy nevieme zodpovedať úplne presne vzhľadom na to, že každý softvérový projekt je jedinečný a kladie na ľudí a zdroje iné požiadavky [4]. Tieto špecifiká a ich dopad na trvanie a výsledky projektu je zvyčajne veľmi zložité a niekedy dokonca takmer nemožné predpovedať v úvodných fázach projektu, kedy robíme prvé odhady, ktoré majú za účel určiť očakávaný rozsah projektu. Na základe týchto odhadov sa môže určovať cena a čas dodania systému a tak môžu mať veľký vplyv na celý projekt. Nesprávny odhad môže

mať za následok stanovenie nereálnych požiadaviek na vyvíjaný systém a nesplniteľné termíny, čo môže spôsobiť veľké škody nielen pre dodávateľa systému, ale aj pre zákazníka. Takýmto situáciám sa prirodzene každá spoločnosť snaží predísť, a preto sa neustále hľadajú a optimalizujú rôzne metódy pre odhad potrebného úsilia.

V súčasnosti zvyčajne nepredstavuje veľký problém získanie potrebných technických zariadení alebo softvéru, ale hlavne zabezpečenie dostatočného počtu kvalifikovaných pracovníkov [5]. Takýto ľudia zvyčajne predstavujú tie najdôležitejšie zdroje, na ktoré treba pri odhadoch najviac prihliadať, pretože ich nie je možné jednoducho podľa potreby pridávať, presúvať alebo vyradzovať tak ako napríklad rôzne technické zariadenia.

Okrem toho pri pracovníkoch vznikajú pri plánovaní dodatočné komplikácie, pretože nie každý pracovník je rovnako výkonný vo všetkých oblastiach ako ostatní. Hoci v jednej oblasti môže vynikať, v ostatných môže byť len priemerný, či dokonca podpriemerný. Štandardne používané človeko-dni, resp. hodiny alebo mesiace, teda nie je jednoznačná konštantná hodnota, ale závisí aj od konkrétnych ľudí. V malom projekte s niekoľkými ľuďmi je možné uvažovať aj o konkrétnych osobách, ktoré budú projekt realizovať. To sa však komplikuje v prípade, že na projekte bude pracovať niekoľko desiatok ľudí a je to v podstate nemožné pre projekty so stovkami alebo dokonca tisíckami ľudí. Schopnosti týchto ľudí tiež nemusia byť presne známe pri plánovaní, obzvlášť v prípade, že sa na projekte budú podieľať noví zamestnanci [1]. Ak však aj poznáme schopnosti jednotlivých pracovníkov na začiatku projektu, počas jeho trvania sa môžu výrazne meniť. Už táto skutočnosť vylučuje vytvorenie úplne presného plánu. Ak totiž vytvárame nejaký odhad v určitých jednotkách, nemôže byť takýto odhad presnejší ako použité jednotky. Keďže jednotky, ktoré používame, sú vo svojej podstate nepresné a skresľujúce, nemôže byť samozrejme úplne presný ani odhad.

Napriek tomu, že presný výsledok nie je možný, nemôžeme odhady úsilia celkom zavrhnúť. Aj približný odhad je podstatne lepší ako žiadny. Hľadaniu metódy odhadu, ktorá by umožňovala čo najviac sa priblížiť k realite, sa venuje veľké množstvo výskumov. Výsledkom týchto výskumov je množstvo rôznych metód pre odhady v softvérových projektoch, ktoré sa navzájom viac alebo menej líšia. V niektorých aspektoch sa však všetky tieto metódy podobajú. Asi najdôležitejšou vlastnosťou, ktorá im je všetkým spoločná je fakt, že akýkoľvek spôsob odhadu sa zvolí, je nevyhnutné vždy brať do úvahy, že odhady nikdy nie sú a ani nemôžu byť nemenné, konečné čísla. Ako sa postupne projekt vyvíja, musia sa postupne vyvíjať aj odhady, aby zachytávali skutočný a aktuálny obraz projektu, a na základe nich sa musí neustále upravovať aj celý plán [5]. Nakoľko totiž odhad nikdy nie je presný, jednotlivé časti projektu sú dokončované skôr, resp. neskôr ako je plánované. Pokiaľ by tieto odchýlky neboli zapracované do plánu, mohlo by dôjsť v lepšom prípade k neefektívnemu využitiu zdrojov a v horšom až k úplnému zlyhaniu plánu, čo by mohlo viesť až k celkovému zlyhaniu projektu.

Odhady v softvérových projektoch všeobecne

Pri odhadoch nielen v softvérových projektoch môžeme postupovať zhora nadol alebo zdola nahor [3]. Postup zhora nadol znamená, že sa robí najskôr odhad trvania riešenia celého problému a ten sa potom delí na jednotlivé časti. Takýto prístup vyžaduje veľké skúsenosti s dosť podobnými projektmi, inak môže byť veľmi nepresný. Zvyčajne sa

používa najmä v skorých fázach projektu, napríklad pri vytváraní ponuky pre zákazníka. Pri odhade zdola nahor sa začína odhadom úsilia pre jednotlivé časti a tie sa následne spočítajú a určia úsilie pre celok. Tento prístup sa používa najviac na plánovanie na úrovni jednotlivých úloh. Pri odhade zdola nahor je potrebné najskôr vytvoriť nejakú hierarchiu úloh, ktoré je potrebné splniť. Členovia tímu potom určia zložitosť komponentov a úloh, za ktoré sú zodpovední, a produktivitu pracovníkov na základe ich skúseností a schopností [3].

Proces odhadovania zvyčajne pozostáva z niekoľkých častí. Na začiatku vždy musia byť stanovené určité predpoklady, z ktorých sa dá vychádzať. Hlavne požiadavky na rozsah a kvalitu systému, termíny a spôsob odovzdania, ale aj dostupné finančné a ľudské zdroje. Na všetky tieto a iné okolnosti musí byť neustále prihliadané a každý vytvorený plán ich musí rešpektovať. Samozrejme nie vždy musia byť konštantné. Neskôr sa môžu upraviť a prispôsobiť požiadavkám projektu, čo je prakticky takmer vždy vo väčšej alebo menšej miere nevyhnutné. Na základe daných požiadaviek sa na začiatku vytvára odhad rozsahu systému, ktorý môže slúžiť ako základ pre odhad úsilia potrebného pre projekt zohľadňujúc dostupné zdroje. Na základe porovnania odhadov so skutočným vynaloženým úsilím sa môžu analyzovať metódy odhadu a tak sa postupne zlepšovať neskoršie odhady [3].

Metódy odhadov

Prvým dôležitým odhadom, z ktorého vychádzajú všetky ďalšie je odhad veľkosti softvérového projektu. Pre vyjadrenie veľkosti projektu je možné použiť niekoľko rôznych metód. Tradične sa používajú napríklad tieto tri [5]:

- Počet riadkov zdrojového kódu
- Funkčné body
- Objektové body

Určenie veľkosti projektu týmto spôsobom však môže byť pomerne náročné a najmä v prípade použitia počtu riadkov zdrojového kódu značne závisí od použitého programovacieho jazyka. Pre elimináciu týchto problémov bola vytvorená metóda *Bodov prípadov použitia.* Táto metóda rozširuje metódu funkčných bodov a využíva veľký nárast popularity jazyka UML. Keďže jazyk UML je dnes používaný pri analýze mnohých softvérových projektov, je táto metóda aplikovateľná v širokej škále projektov [5].

Pri tejto metóde sa v diagramoch prípadov ohodnotia zložitosti jednotlivých hráčov a prípadov použitia ako:

- Jednoduché
- Priemerné
- Zložité

Pre každý prípad použitia, resp. hráča, sú podľa zložitosti priradené konkrétne hmotnostné faktory. Ich súčet určuje neupravené body prípadov použitia. Tieto body sú následne upravené prenásobením technickými a environmentálnymi faktormi. Technické a environmentálne faktory vyjadrujú technickú zložitosť projektu, nefunkcionálne

požiadavky a podobne. Existuje trinásť technologických a osem environmentálnych parametrov, ktoré treba zohľadniť. Každý z nich má určitú váhu, ktorá určuje jeho vplyv na odhad vzhľadom k ostatným. Z takto upravených bodov prípadov použitia je možné vypočítať odhad úsilia v človeko-hodinách. V literatúre sa zvyčajne odporúča počítať s 20 až 36 človeko-hodinami pre jeden bod prípadov použitia [5].

Jednou z nevýhod tohto modelu je to, že jednotlivé faktory sa uvažujú nad celým projektom, napriek tomu, že sa môžu dotýkať len niektorých prípadov použitia, resp. hráčov a tým sa do odhadu zavádza chyba [5]. Zrejmou nevýhodou tohto modelu je aj to, že je potrebné už pred odhadom mať dobre spracované diagramy prípadov použitia a taktiež pomerne dobrú predstavu o architektúre systému a jeho implementácii, aby bolo možné správne nastaviť všetky faktory a predpokladám, že najmä pri niektorých menších projektoch môžu tieto analýzy predstavovať pomerne veľkú časť celkového úsilia [1].

Iným veľmi známym a používaným modelom, pomocou ktorého je možné vypočítať úsilie potrebné pre projekt je COCOMO (Constructive Cost Model) vytvorený Barrym Boehmom v sedemdesiatych rokoch alebo jeho upravená verzia COCOMO II. Počítajú pravdepodobné úsilie na základe odhadu veľkosti projektu vyjadreného v počte riadkov kódu, funkčných alebo objektových bodoch [1]. Tieto modely predpokladajú inkrementálny vývoj softvéru a teda musia poskytovať aj možnosti pre výpočet úsilia, ktoré je potrebné vyvinúť medzi jednotlivými verziami a na údržbu systému. COCOMO II taktiež zahŕňa model pre odhad znovu použitia zdrojového kódu, kde zohľadňuje množstvo kódu, ktoré je prevzaté, koľko a ako je ho potrebné upraviť, ako dobre vývojári poznajú existujúci projekt, ako sa vyznajú v zdrojovom kóde a ďalšie faktory [5].

Obidve popísané metódy sú podľa môjho názoru dostatočne všeobecné a je možné ich aplikovať na širokú škálu projektov a pokiaľ dokážu experti podieľajúci sa na projekte dostatočne presne odhadnúť väčšinu vstupných parametrov, najmä tých s väčšou váhou, odhady by mohli byť relatívne presné. Samozrejme nie je možné ich výsledky použiť ako definitívne záväzné termíny, ale môžu dať ľuďom zodpovedným za projekt aspoň nejakú predstavu o tom, čo môžu čakať. Ich výhodou je aj to, že nutne nevyžadujú, aby mal tím pracujúci na projekte skúsenosti s podobnými projektmi. Podľa môjho názoru je vhodné tieto parametrické modely využívať najmä v takýchto prípadoch.

Ak však ľudia v tíme majú skúsenosti s podobným projektom, myslím, že je pre nich výhodnejšie použiť odlišný prístup k odhadu. Môžu sa pokúsiť odhadnúť potrebné úsilie na základe analógie s predchádzajúcimi projektmi. Ak už daní ľudia v minulosti pracovali na podobných projektoch, nielen budú schopní urobiť túto prácu rýchlejšie a lepšie, ale čo je z hľadiska plánovania najdôležitejšie, budú pravdepodobne schopní pomerne presne odhadnúť, ako dlho bude trvať urobiť ekvivalentnú prácu v novom projekte.

Odhad na základe analógie teda vychádza z predošlých skúseností s podobnými projektmi. Žiadne dva projekty však nie sú úplne rovnaké, vždy existujú určité rozdiely, ktoré môžu mať výrazný dopad na výslednú náročnosť projektu [2]. Pokiaľ by tieto odlišnosti neboli brané do úvahy, mohli by mať za následok výrazné nepresnosti v odhade. Preto je potrebné nájsť v obidvoch projektoch vlastnosti a časti, ktoré majú podobné a je ich možné použiť pre odhad [4]. Existuje viacero výskumov, ktoré sa zaoberajú hľadaním spôsobov ako zvoliť správnu podmnožinu vlastností projektu, ktoré je možné použiť ako základ pre odhad založený na analógii. V týchto výskumoch sú na tento

účel používané rôzne metódy ako napríklad umelá inteligencia, neurónové siete, heuristiky a iné [2].

Zhodnotenie

Plánovanie je veľmi komplikovanou a nedokonalou, no zároveň nevyhnutnou súčasťou každého softvérového projektu. Činnosti ako analýza, implementácia a testovanie sú také komplexné, že nie je možné jednoducho povedať ako dlho budú trvať. Ovplyvňuje ich totiž obrovské množstvo okolností, z ktorých mnohé nie sú ľahko alebo vôbec predvídateľné. Keďže však je potrebné ich trvanie dopredu poznať, museli sa nájsť nejaké spôsoby pre približný odhad. Týmto spôsobom sa hovorí odhad úsilia.

Nejde o nijaké exaktné postupy alebo výpočty, ale skôr len o nejaké hrubé orientačné údaje. Napríklad pri parametrických modeloch ako sú body prípadov použitia, COCOMO a podobné, ide podľa môjho názoru často len o nejaké formálne odhady, ktorých úlohou je podložiť odhady nejakou sofistikovanou metodikou, ktorá má navodiť dojem exaktnosti a systematického prístupu. Pred zákazníkom je totiž lepšie povedať, že všetky odhady, na základe ktorých je určovaná cena a termíny dodania, je vypracovaná na základe nejakej metodiky zodpovedajúcej štandardom, ako mu povedať pravdu, že v skutočnosti nevieme ako dlho bude projekt skutočne trvať a predkladané čísla sú len nejaké neurčité termíny, o ktorých by sa dalo predpokladať, že budú realizovateľné. Alebo dokonca, že ide o termíny navrhnuté len v snahe získať projekt bez príliš veľkého ohľadu na skutočné možnosti a zdroje.

Domnievam sa, že odhad na základe analógie je lepším riešením. Opiera sa o reálne skúsenosti s reálnymi projektmi a tak prináša do odhadu istú mieru presnosti. Najmä ak majú ľudia skúsenosti s viacerými podobnými projektmi a poznajú teda požiadavky na systém, majú dobrú znalosť používaných technológií a podobne. Nevýhodou je, že táto metóda nie je použiteľná, ak ide o projekt odlišného typu než na aké sú zodpovední experti zvyknutí. Aj vtedy sa však zvyčajne dajú nájsť nejaké podobné časti, ktoré je možné použiť.

Jedným z najväčších problémov je podľa môjho názoru rozdielna výkonnosť ľudí. Odhady sa spravidla vyjadrujú v človeko-hodinách alebo človeko-mesiacoch, ktoré však vychádzajú z toho, že všetci ľudia majú rovnakú produktivitu, čo samozrejme nie je správny predpoklad. Myslím si, že by bolo vhodnejšie používať nejakú inú normovanú jednotku, ktorá by sa priamo nevzťahovala na jedného človeka, ale skôr na nejakú jednotku práce. Takouto jednotkou by bolo možné vyjadrovať aj produktivitu zamestnancov. Samozrejme by to bolo značne problematické najmä pri väčších tímoch, no mohlo by to pomôcť spresniť odhady najmä na nižších úrovniach pri rozdeľovaní práce v menších tímoch.

Použitá literatúra

1. Asundi, J.: The Need for Effort Estimation Models for Open Source Software Projects. Proceedings of the fifth workshop on Open source software engineering, 2005, s. 1-3

2. Azzeh, M., Neagu, D., Cowling, P.: Improving Analogy Software Effort Estimation using Fuzzy Feature Subset Selection Algorithm. Proceedings of the 4th international workshop on Predictor models in software engineering, 2008, s. 71-78
3. Bajaj, N., Tyagi. A., Agarwal. R.: Software Estimation – A Fuzzy Approach. ACM SIGSOFT Software Engineering Notes, Volume 31, Issue 3, ACM, New York, 2006.
4. Li, J., Ruhe, G.: A Comparative Study of Attribute Weighting Heuristics for Effort Estimation by Analogy. Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, 2006, s. 66-74
5. Mohagheghi, P., Anda, B., Conradi, R.: Effort Estimation of Use Cases for Incremental Large-Scale Software Development. Proceedings of the 27th international conference on Software engineering, 2005, s. 303-311

Annotation

Effort estimation in planning of software projects

For success of any software project it is very important, it is well planned. Effective planning requires knowledge of its parts and their resource requirements. The most important resource in the most of the software projects are qualified workers and therefore all the parts of the project must be well assigned to them. For this purpose it is very important to be able to estimate time required to solve the problem with available resources not only in the beginning, but also during the entire project. In the present there are several methods, based on which it is possible to make these estimations. What are there advantages and disadvantages? When is it appropriate to use them? In this essay I want to take a look at several different approaches, which can be used to estimate required effort and I will try to evaluate their asset for project planning.