Odhady pracnosti

- důležité kvůli plánování
- příliš krátký odhad - projekt špatně naplánován, výsledná cena projektu se zvýší => nespokojenost zákazníků, vývojářů i managementu
- příliš dlouhý odhad - skutečná cena se také zvýší, protože do projektu budou zahrnuty věci, které nejsou nutné

- jsou obtížné – na začátku jen mlhavá představa toho, co máme naprogramovat, teprve v čase podrobnosti a konkrétní rozhodnutí (všechny problémy a všechna rozhodnutí známe až po naprogramování systému☺)

-> odhady jsou nutně na začátku nepřesné a teprve postupně se zlepšují

Např.

- bude zákazník chtít, aby program uměl X?
- jak složitá bude programová podoba X? (obtížnost implementace se může lišit i 1:10)
- jak kvalitně bude X naprogramováno? (počet chyb se může lišit 1:10)
- jak dlouho bude trvat oprava chyb v X? (dokonce mezi programátory se stejnou zkušeností je opět rozdíl 1:10 ve schopnosti najít chyby)
- jak dlouho bude trvat zaintegrovat vlastnost X do zbytku systému?

V různých stadiích projektu přece jen lze predikovat přesnost odhadu ceny (tj. pracnosti) i doby trvání:

Odhad pracnosti a velikosti produktu a délky vývoje

Fáze	**Pracnost a velikost min**	**Pracnost a velikost max**	**Délka vývoje min**	**Délka vývoje max**
úvodní koncept	0.25	4.00	0.60	1.60
schválený koncept	0.50	2.00	0.80	1.25
specifikace požadavků	0.67	1.50	0.85	1.15
specifikace návrhu	0.80	1.25	0.90	1.10
podrobný návrh	0.90	1.10	0.95	1.05

- odhady by měly začínat od velkého rozpětí a postupně konvergovat (násobení optimistickým koeficientem -> optimistický odhad, násobení pesimistickým koeficientem -> pesimistický odhad)

- koeficienty pro pracnost a odhad doby trvání se liší - předpokládá se, že napřed si odhadneme pracnost a z ní odvodíme dobu trvání

- odhady nemohou být lepší, než udává výše uvedená tabulka -> zákazník by se měl dozvědět, kdy budeme mít k dispozici lepší odhady, máme dobrou vůli – řekneme mu to, jakmile to budeme sami vědět)

„Build to budget“ (vlastnosti produktu vs. cena)

- většina zákazníků na začátku chce víc, než kolik si může dovolit -> budou muset buď omezit svoje přání (= rozsah produktu) nebo na něj budou muset uvolnit víc prostředků

- otázka - jak přizpůsobit vlastnosti sw dostupným prostředkům? -> způsob "build to budget"
  - podstatně se liší od "programování pro zábavu", kdy programujeme věci, které nám připadá zajímavé naprogramovat
  - kdykoli máme volbu nějakou vlastnost buď naprogramovat nebo vynechat, musíme ji vynechat
  - kdykoli se rozhodujeme, jestli vlastnost naprogramujete lépe nebo levněji, musíme ji naprogramovat levněji
  - jakmile občas naprogramujete lepší vlastnost za cenu času, není to "build to budget" ale normální vývoj SW
  - je zapotřebí, aby zákazník předem věděl, co to pro něj bude znamenat

Odhad délky vývoje - 3 kroky:

- odhad rozsahu produktu (počet nekomentářových řádek kódu nebo počet funkčních bodů – umělá metrika)
- odhad pracnosti (člověkoměsíce) - nejlepší je vycházet z projektů obdobného rozsahu, které jste už realizovali
- odhad doby trvání (kalendářní měsíce) - lze odvodit z odhadu pracnosti; podle tabulky získáme rozsah, který můžeme zveřejnit

Odhad rozsahu produktu - několik přístupů:

- % velikosti podobné součásti starého systému; nejlépe po menších částech - čím menší části, tím přesnější odhad
- algoritmický přístup - odhaduje velikost programu z požadovaných vlastností; většinou je zapotřebí použitý algoritmus nebo software kalibrovat na způsob práce v daném prostředí (nutné vycházet z nasbíraných historických dat)
- pro velké projekty je pro odhady výhodné používat SW nástroje -> výsledky jsou přesnější

Pozn:

- algoritmický přístup lepší na začátku projektu (o projektu toho moc nevíme).
- přibližně od prostředku fáze návrhu do konce projektu lepší odhady vytvořené součtem odhadů jednotlivých částí projektu.

Odhad pracnosti, tj. počtu člověkoměsíců

- důležitý sběr dat z předchozích projektů a jejich použití (odhad pracnosti po paměti ->podcenění projektu o cca 20 až 30%).

- nejlépe použít nasbíraná data z vlastní organizace (trvání projektů obdobné velikosti)

- existují algoritmy nebo programy, které např. z počtu řádků a typu SW vytvoří odhady pracnosti (př. systémový program o 10 000 řádcích má nejmenší(!) možnou pracnost cca 25 člověkoměsíců, stejný počet řádek pro obchodní aplikaci cca 5 člověkoměsíců

- pozor na opomenutí úkolů, jako je konverze dat, instalace, komunikace v rámci beta-testování, demonstrace programu zákazníkovi, opravy chyb v jiných programech, integrace, dovolená, nemoc, schůze, školení apod.

Pozn. Počet řádků kódu

- vytvořit program s 1000 řádkami kódu je přibližně stejně obtížné v kterémkoli imperativním programovacím jazyku

- na 1000 řádkách ale můžeme mít různou funkčnost (1000 řádkový program v assembleru bude mít pouze cca 350 řádek v C, tj. vlastní kódování bude cca 3x rychlejší)

- počet řádek/funkční bod (zdroj: www.qsm.com/FPGearing.html):

- nemá smysl řádek/funkční bod přeceňovat (kódování tvoří jen malou část celé tvorby produktu): pokud se kódování omezí o 50% a přitom tvoří 30% celkové práce, pak se efektivita zvýší o pouhých 15%

Přepočet trvání projektu z odhadu pracnosti

Např. rovnice:

kalendářní měsíce = 3.0 * člověkoměsíce ^ (1/3)

- tj. např. 25 člověkoměsíců bude v optimálním případě trvat 8.7 kalendářních měsíců (37 týdnů) (pokud už máme požadavky, bude rozptyl: 7.4 až 10 měsíců)

- průměrná velikost týmu - vydělení pracnosti (člověkoměsíců) trváním projektu (kalendářní měsíce); (tj. pro př. výše 3 lidi)

- Pro malý projekt tedy cca

architektura a návrh	10% -> 4 týdny (po zaokrouhlení)
podrobný návrh	20% -> 8
kódování a ladění	25% -> 9
testování jednotek	20% -> 7
integrace	15% -> 5
testování systému	10% -> 4

- Když ve skutečnosti návrh trvá 5 týdnů -> správně je vynásobit celkový odhad měřítkem zpoždění, tj. (5/4)*8.7 = 10 kalendářních měsíců.

Ale: pokud je odhad podstatně delší než je požadováno, zákazník nebo manažer odhad jednoduše ignoruje

Jedno z největších umění programátorů

pokud máte správně udělaný odhad – stát si za svým (nenechat se zmanipulovat)

pokud je projekt špatně naplánován, doba trvání projektu se tím ještě zvýší

- pod velkým tlakem člověk dělá více chyb (ideální míra stresu – lepší než žádný stres) -> další zpoždění -> větší tlak -> uzavřený kruh
- pokud termín není principielně možné stihnout, motivace rapidně klesá, hrozí syndrom vyhoření (jinak postihuje zejména pomáhající profese – zdravotníci, hasiči, učitelé), odchod za lepším místem

Poznámka (příklad špatně naplánovaného projektu a jeho důsledky)

Microsoft Word for Windows 1.0:

- na začátku byl požadavek "vytvořit nejlepší wordprocesor, nejlépe do roka"
- na základě požadavků byla správná doba pro projekt 22 měsíců
- protože nebylo dobře naplánováno (12 vs. 22 měsíců), špatný návrh i implementace, mnoho vývojářů odešlo
- vývoj celkem trval 5 let, z toho poslední rok "stabilizace", tj. vychytávání chyb