SROVNÁNÍ METOD COCOMO A ANALÝZY FUNCTION POINTS

THE COMPARISON OF METHODS COCOMO AND FUNCTION POINTS ANALYSIS

Zdeněk Struska, Robert Pergl

Anotace:
Článek představuje a porovnává metody Cocomo a analýzu function points, které se používají v počátečních fázích pro odhad složitosti vývoje informačních systému. Článek se zaměřuje na rozdílné přístupy obou metod a na výhody a nevýhody jednotlivých postupů.

Klíčová slova
Bezporuchovost, dostupnost systému, metoda function points, Cocomo 1.1, Cocomo 2.0., použitelnost, rychlost odhadu, složitost.

Annotation:
The paper introduces and compares Cocomo and Function Points Analysis methods, which are used in the first phases for the estimation of information system complexity. As well it tries to advise of the differences in individual procedure.

Keywords:
Reliability, system accessability, funtion points method, Cocomo 1.1, Cocomo 2.0, reuseability, speed of estimation, complexity.

ÚVOD

Článek staví vedle sebe dvě metody, které se používají pro odhad složitosti softwaru v době, kdy je znalost tohoto údaje klíčová, jde o počáteční fáze vývoje IS.

Metoda IFPUG vzešla z dílny mezinárodní organizace International Function Point User Group a navázala na koncept A. Albrechta, který vyvinul základní koncept metody funtion points v laboratořích IBM na konci sedmdesatých let minulého století. Metoda se pokouší odhadovat pracnost vyvíjeného softwaru v počátečních fázích jeho vývoje prostřednictvím funkcí požadovaných od IS a podmínek, ve kterých je IS vytvářen.

Původní metoda Cocomo 1.1 (COnstructive COst MOdel) byla vyvinuta na přelomu sedmdesátých a osmdesátých let minulého století Hary Boehmem. V průběhu vývoje v oblasti tvorby informačních systémů byl původní ohad Cocomo nahrazen novou verzí, která je známa pod označením Cocomo 2.0. Nová verze odráží vývoj v oblasti IT ve vztahu s postupným přechodem na objektové prostředí.

CÍL

Příspěvek si klade za cíl krátké představení vybraných metod zaměřených na odhad složitosti softwaru. Záměrem je především porovnat přístupy a postupy obou metod v procesu odhadování. Vypíchnout jednotlivé přednosti a nedostatky každé z nich a v závěru porovnat jejich skutečnou použitelnost.

1. DVA PŘÍSTUPY K ODHADU SLOŽITOSTI SOFTWARU

Existuje několik kategorií, do kterých lze zahrnout vybrané metody použitelné pro odhad složitosti/pracnosti výoje informačních systémů. Z této množiny jsem si vybral kategorii Odhadů založených na modelech. Článek se dále soustředí na metodu function points, kterou považuji za jednu z klíčových metod, jenž svým způsobem ovlivňovala rozvoj v oblasti odhadu vývoje informačních systémů.

Jako druhou je zvolena metoda Cocomo (COnstructive COst MOdel), kterou považuji za jednu z nejznámějších. Původní koncept metody Cocomo nese označení 1.1 a v porovnání s metodou function points byla jeho použitelnost limitována faktorem znalosti počtu zdrojových řádků. Tento handicap by měla odstranit nová verze označovaná jako Cocomo 2.0.

Metoda function points

Metoda funkčních jednic (FP) nepředstavuje jediný ucelený návod, ale spíše soubor návodů, které se navzájem liší podle typu produktu, který bude vytvářen. V tomto směru nabízí metoda funkčních jednic prostor pro vznik různých modifikací.

Pro tento článek jsem si vybral metodu známou pod označením IFPUG (International Function Point Users Group). Důvodem je, že metoda je „nejčistčí“ verzí původního konceptu metody FP, která byla navržena v laboratořích IBM v sedmdesátých letech minulého století. Metoda funguje jako nástroj pro měření funkční hodnoty obchodních požadavků aplikačního softwaru viděného z pohledu uživatele softwaru.

Prostřednictvím uživatelského pohledu je vytvořen formální popis obchodních požadavků na vytvářený SW pomocí uživatelského jazyka. Aby mohly být uživatelské požadavky promítnuty do výsledného SW, musí je vývojový pracovníci přeložit do programovacího jazyka.

Přístup vychází z představy, že detailnější část funkcionality, která musí být vyvinuta, zůstává uživatelsko-obchodním potřebám skryta. Jednoduchým příkladem je, že pro budoucího uživatele není příliš podstatné, zda je požadovaná funkcionalita zajištěna využitím jednoho procesoru či distribuována na několik vzájemně spolupracujících procesorech. Uživatel vidí pouze výslednou funkcionalitu a jen jednu hodnotu bez ohledu na další implementační úvahy.

Z tohoto hlediska je funkcionalita softwaru v jednotlivých vrstvách pod aplikační úrovní pro „normálního uživatele“ neviditelná a další potřeba měnit její funkčnost je ignorována. Další podmínkou je, že dávka toku procesů, jedná se především o základní procesy, které jsou měřeny, zachycuje jen vstupy a výstupy přecházející vnější hranici, tzn. musí být viděny „normálním uživatelem“. Jakákoli další funkcionalita, která není rozeznatelnou částí těchto procesů, je ignorována.

Metoda COCOMO

Cocomo 1.1

COCOMO (COnstructive COst MOdel) je jedním z nejznámějších a nejpopulárnějších modelů pro odhad nákladů. Metoda COCOMO byla vyvinuta na přelomu sedmdesátých a osmdesátých let minulého století Barry Boehmem. Jeho první model se skládal z hierarchie tří modelů s rostoucím významem detailu – základní model COCOMO, střední model COCOMO a pokročilý model COCOMO. Tyto modely byly vyvinuty pro odhad projektů

zákaznického softwaru a softwaru stavěného na míru. Rovnice pro výpočet základního a středního modelu jsou k dispozici v tabulce 2 a 3.

Cocomo 2.0

COCOMO 2.0 se objevilo v polovině 90. let kvůli neschopnosti původního modelu COCOMO (COCOMO 1.1) přesně odhadovat objektově-orientovaný software, evoluční modely (evolutionary models) a software vyvinutý pro komerční účely na sklad (commercial-off-the-shelf software).

Několik základních rozdílů mezi COCOMO 1.1 a 2.0:

Zatímco COCOMO 1.1 musí znát rozsah softwaru v KSLOC (thousand source lines of code) jako vstup, COCOMO 2.0 poskytuje rozdílný model odhadu pracnosti založený na vývojových etapách projektu

Zatímco COCOMO 1.1 poskytovalo bodové odhady pracnosti a časový plán, COCOMO 2.0 poskytuje pravděpodobný rozsah odhadů reprezentující jednu standardní odchylku blížící se nejpravděpodobnějšímu odhadu. (viz. tabulky 2. a 3.)

COCOMO 2.0 je nastaveno pro znovupoužití softwaru a reinženýring, jsou zde užity automatické nástroje pro překlad existujícího softwaru. COCOMO 1.1 poskytovalo malou přizpůsobitelnost pro tyto faktory.

COCOMO 2.0 také počítá s požadavky proměnlivosti ve svých odhadech.

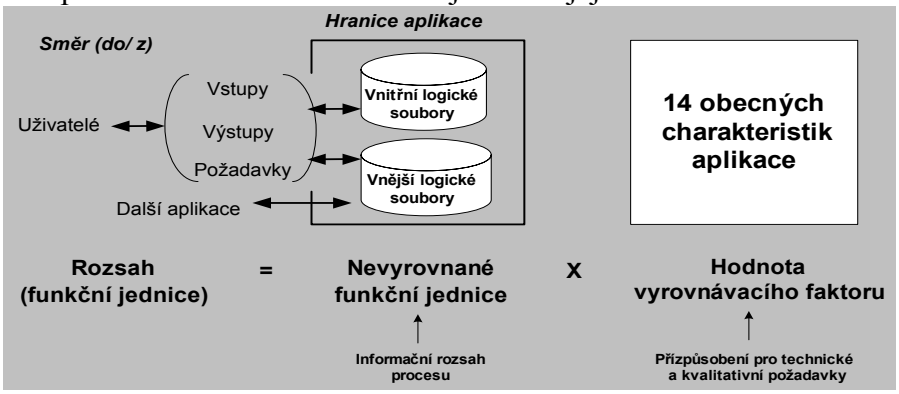
2. ZÁKLADNÍ KONCEPTY METOD

2.1 Koncept metody IFPUG

Jak už bylo v článku jednou naznačeno, je metoda IFPUG nejbližší modifikací původní metody function points vyvinuté v druhé polovině sedmdesátých let. Měřený software nebo-li jeho FUR (Functional User Requirements) jsou rozděleny do třech typů elementárních procesů (vstupy, výstupy a požadavky) a dvou typů souborů (vnitřní a vnější logické soubory). Každá komponenta je poté klasifikována dle stupnice

jednoduchá – průměrná – složitá

a je ohodnocena počtem funkčních bodů závisejících na její složitosti.



Obr. 1: Schéma výpočtu metody IFPUG

V prvním kroku dojde k ohodnocení složitosti funkčních souborů v ordinální stupnici: jednoduchý, průměrný a složitý. Komponenty se na počátku rozpadnou do dvou kategorií datových (EI, EO, EQ) a transakčních (ILF, EIF) funkcí. Po rozdělení každé funkce do jedné ze třech kategorií, přidělíme každé z nich stanovenou hodnotu (viz. tab. 1.), počet funkcí v kategorii roznásobíme s přidělenou jakousi váhou, sečteme v řádcích a následně v celém sloupci. Výsledkem je celkový počet nevyrovnaných funkčních jednic (UFP).

Typ komponenty	Složitost komponent			
	Nízký	Průměrný	Vysoký	Celkem
External Inputs	__ x 3 = __	__ x 4 = __	__ x 6 = __	
External Outputs	__ x 4 = __	__ x 5 = __	__ x 7 = __	
External Inquiries	__ x 3 = __	__ x 4 = __	__ x 6 = __	
Internal Logical Files	__ x 7 = __	__ x 10 = __	__ x 15 = __	
External Interface Files	__ x 5 = __	__ x 7 = __	__ x 10 = __	
	Celkový počet nevyrovnaných funkčních jednic (UFP)			
	Vynásobíme hodnotou vyrovnávacího faktoru (VAF)			
	Upravené funkční jednice celkem (FP)			

Tab. 1: Váhy složitosti jednotlivých funkcí

Dále je vyvíjený informační systém ohodnocen z pohledu podmínek, ve kterých je vyvíjen. Pracuje se se 14 předdefinovanými technickými a kvalitativními charakteristikami, každá z charakteristik se ocení váhou 0 až 5 dle vlivu dané charakteristiky na vyvíjený IS (0 – faktor nemá žádný vliv, 5 – faktor má velmi významný vliv). Poté se přidělené faktory roznásobí váhou každé z charakteristik a po jejich sečtení se dosazením do stanoveného vzorce získá hodnota vyrovnávacího faktoru (VAF).

Suma hodnot UFP všech jejích komponent dává nevyrovnanou hodnotu části softwaru. Tato hodnota je dále roznásobena s faktorem VAF, který je určen příspěvky 14 technických a kvalitativních charakteristik poskytujících vyrovnávající část celkové hodnoty function points. Výsledkem je suma vyrovnaných funtion points.

2.2 Koncept metody Cocomo

COCOMO 2.0 Application Composition model (viz. tabulka 2, 3. řádek) užívá object points pro provedení odhadu. Model přejímá užití integrovaných CASE nástrojů. Objekty zahrnují obrazovky, reporty a moduly v programovacích jazycích třetí generace. Object points nejsou nutně navázany na objekty v objektově orientovaném programování. Počet nezpracovaných objektů je odhadnutý, složitost každého objektu je odhadnuta a je vypočítán vážený součet (Object-Point count). Procento užití a předpokládaná produktivita jsou odhadnuty také. S touto informací, může být spočítán i odhad pracnosti (viz. tabulka 2.)

Model Name	Effort Equation	Parameters
Basic COCOMO	Effort = a(KSLOC)^b	Organic: a=2,4, b=1,05 Semidetached: a=3,0, b=1,12 Embedded: a=3,6, b=1,20
Intermediate COCOMO	Effort = EAF a (KSLOC) ^{b}	EAF is the product of 15 cost driver attributes Organic: a=3,2, b=1,05 Semidetached: a=3,0, b=1,12 Embedded: a=2,8, b=1,20

Model Name	Effort Equation	Parameters
COCOMO 2.0 Application Composition Model	$Effort = \frac{NOP}{PROD}$ $NOP = OP\frac{(100 - \%reuse]}{100}$	NOP is New Object Points PROD is Productivity Rate 4 – very low 7 – low PROD = 13 – normal 25 – high 50 – very high
COCOMO 2.0 Early Design and Post Architecture Model	$Effort = \frac{ASLOC(\frac{AT}{100})}{ATPROD} + a[size(1 + \frac{BRAK}{100})]^b \pi EM$ $b = 1{,}01 + \frac{1}{100}\sum SF_j$ $Size = KSLOC + KASLOC(\frac{100 - AT}{100})AAM$	a = 2,5 SF_j = scale factor EM_i = effort multiplier BRAK = percentagecode discarded due to requirements volatility ASLOC = size of adapted components AT = percent of components adapted ATPROD = Automatic Translation Productivity AAM = Adaptation Adjustment Multiplier

Tab. 2: Vzorce pro výpčet odhadů pracnosti

Function points utvářejí rozsah vstupů pro model COCOMO 2.0 Model Early Design.

Model Name	Schedule Equation	Parameters
Basic and Intermediate COCOMO	$Schedule = 2{,}5Effort^c$	Organic: c = 0,38 Semidetached c = 0,35 Embedded c = 0,32
COCOMO 2.0	$Schedule = c[Effort^{0{,}33+0{,}2(b-1{,}01}]\frac{SCED\%}{100}$ $b = 1{,}01 + \frac{1}{100}\sum SF_j$	c = 3, 0 SFj = scale factor SCED% = schedulE compression/expansion parameter

Tab. 3: Vzorce pro výpočet časových plánů

DISKUSE

Function Points jsou pravděpodobně jednodušeji odhadnutelné než prvotní KSLOC v životním cyklu. Metoda Cocomo2.0 pracuje s function points jako se vstupním údajem.

ZÁVĚR

Pro praktickou aplikaci představených metod bych doporučovat použít kombinaci výše představených metod.

LITERATURA:

[1] Vaníček, J.: Měření a hodnocení jakosti informačních systémů. PEF ČZU. Praha 2004.
[2] International Function Point Users Group: IT Measurement. Practical Advice from the Experts. Addison-Wesley. Boston 2002
[3] McGibbon, T.: Modern Empirical Cost andSchedule Estimation Tools. www.dacs.dtic.mil/tech/estimation/costtools.pdf

Adresa autora:

Ing. Zdeněk Struska, Ing. Robert Pergl, Katedra informačního inženýrství, PEF ČZU Praha, struska@pef.czu.cz