Měření v softwarovém projektu

Autor: Jitka Kreslíková

© 2020

Ústav informačních systémů

Fakulta informačních technologií

Vysoké učení technické v Brně

Řízení projektů

Měření v softwarovém projektu

- Co jsou metriky?
- Měření
- Oblasti uplatnění softwarových metrik
- Metody a prostředky měření
- Obvykle používané metriky
- Metody odhadu pracnosti, softwarové metriky

Měření v softwarovém projektu

Motto: "nemůžeme řídit to, co nemůžeme měřit" - Tom DeMarco.

Pro úspěch projektu je důležitá dohoda na kriteriích akceptování projektu:

př.: operátor musí být schopen začít pracovat s libovolnou funkcí do 30 sekund od chvíle, kdy se posadil za terminál.

- Co jsou metriky:
  - můžeme je definovat jako kriteria určující atributy softwarového projektu,
  - umožňují hodnocení vytvořeného výrobku a procesu, který se použil při jeho tvorbě.

Měření v softwarovém projektu

- Měření
  - je proces objektivního přiřazování symbolů entitám (obvykle čísel) za účelem charakteristiky určitého atributu dané entity,
  - Entita - objekt nebo událost, jejíž charakteristika se měří.
  - Atribut - rys, charakteristika, která nás zajímá.
  - Objektivita - skutečnost, že vlastní proces měření musí vycházet z přesně definovaného pravidla, t.j. měření nezávisí na tom, kdo ho provádí.
  - Různé interpretace naměřených hodnot:

    př.: počet chyb nalezených za jednotku času reprezentuje kvalitu testování nebo (ne)spolehlivost programu..

Měření v softwarovém projektu

Rozlišujeme měření:

- přímé - přímé získání hodnoty sledovaného atributu

  př.: počet řádků programu.

- nepřímé - odvození z jiných atributů, které nelze měřit přímo,

  př.: udržovatelnost je možné měřit časem na odstranění chyby.

Oblasti uplatnění softwarových metrik

- Definice uživatelských požadavků ve formě měřitelných atributů produktu umožňuje předcházet potenciálním nedorozuměním,
- poskytování kvantifikovatelného přehledu pro podporu rozhodování,

  př.: hodnocení efektivnosti strategií testování, identifikace potenciálně kritických a chybových modulů.

- vytváření přesnějších odhadů (nákladů a doby trvání projektů),

Oblasti uplatnění softwarových metrik

- sledování kvality softwaru i procesu jeho tvorby,
- přehlednější vývoj softwaru a dřívější identifikace potenciálních problémů na základě naměřených hodnot vybraných atributů charakterizujících projekt,
- vyhodnocování vlivu nových metod a podpůrných prostředků na produktivitu práce a kvalitu,
- zlepšování procesu vývoje a produktu.

Metody a prostředky měření

Metody měření závisí na charakteristikách:

- **jaké entity** se měří (výrobky, procesy),
- **proč** se měří (vyhodnocení, řízení, předpověď, zlepšení),
- **koho** zajímají výsledky měření (návrhář, manažer),
- **které** vlastnosti se měří,
- **v jakém prostředí** se měří (lidé, technologie, jiné zdroje).

Typy měření

Aby bylo možné výsledek měření interpretovat, je nutné ho reprezentovat na stupnici.

5 různých typů měření:

- **nominální** (klasifikace) - stanoví pouze, zda se měřená veličina "rovná" či "nerovná",

  př.: typ programovacího jazyka.

- **ordinální** (+ uspořádání) - uspořádání nad hodnotami umožňuje změřit úroveň atributu a porovnání. Není však řečeno jaká je diference mezi hodnotami.

  př.: méně modulární program.

Typy měření

- **intervalové** (+ vzdálenost mezi prvky stupnice).

  př.: rozdíl v modularitě mezi programem X a Y je stejný jako mezi programem Y a Z.

- **poměrové** (+ absolutní nula) - umožňuje porovnání poměrů. Absolutní nula znamená, že existuje hodnota udávající absenci zkoumané vlastnosti,

  př.: program X je dvakrát tak modulární jako program Y.

- **absolutní** - všechny matematické operace jsou smysluplné,

  př.: počet řádků textu programu, počet chyb.

Měření v softwarovém projektu

Příklad metriky:

Entita : zdrojový text programu

Atribut: modularita

1: nízká	**málo velkých komponent**
2: střední	**procedurální komponenty, které zahrnují základní funkcionalitu**
3: vysoká	**funkcionální dekompozice, použití abstraktních typů dat**

Obvykle používané metriky

- Metriky produktu
- Metriky pro proces vývoje softwarového produktu
- Metriky pro zdroje
- Testové metriky

Metriky produktu

- velikost, rozsah - používá se na odhad času, nákladů a měření produktivity:
  - počet řádků textu programu (LOC),
  - počet funkčních bodů (FP),
  - počet modulů,
  - průměrný počet LOC na modul,
  - rozsah dokumentace (počet stran, slov).
- modularita
  - svázanost modulů - počet toků dat a řízení mezi moduly a počet globálních struktur dat.
  - soudržnost modulů - vztah mezi funkcemi modulu a jejich vztah ke zbytku programu.

Metriky produktu

- spolehlivost (PDMV) průměrná doba mezi výpadky systému:
  - (PDNV) průměrná doba do následujícího výpadku,
  - (PDO) průměrná doba opravy,

  PDMV = PDNV + PDO,
- dostupnost (D) - pravděpodobnost, že v daném čase program pracuje správně podle specifikace:
  - D = 100 x PDNV / PDMV [%]

Metriky produktu

- složitost
  - počet souborů,
  - počet příkazů,
  - počet větvení,
  - hloubka zanoření řídících struktur,
  - počet cyklů,
  - cyklomatické číslo:
    - počet nezávislých cest v grafu řízení programu mínus počet rozhodovacích příkazů + 1
  - paměťové nároky, nároky na procesor.

Metriky produktu

- chyby
  - počet chyb v programu,
  - chybovost : počet chyb / FP,
  - klasifikace chyb a frekvence jejich výskytu,
  - chyby v dokumentaci,
- udržovatelnost
  - střední doba potřebná na opravu chyby,
  - střední doba na pochopení logiky modulu,
  - střední doba na zpřístupnění příslušné informace v dokumentaci.

Metriky pro proces vývoje softwarového produktu

- úsilí
  - čas vynaložený na vývoj systému (člověko-měsíce).
- změny požadavků (odráží kvalitu specifikace požadavků),
  - počet změn požadavků,
  - střední doba od ukončení specifikace do oznámení požadavku na změnu,
- náklady a čas
  - začátky a konce činností,
  - trvání činností,
  - náklady na provedení jednotlivých činností.

Metriky pro proces vývoje softwarového produktu

Metoda SSPI (Statistical software process improvement)

- chyby a defekty jsou kategorizovány podle původu (chyba ve specifikaci, v logice, ..)
- je stanovena cena za opravu chyby
- sečte se počet chyb podle kategorie
- je stanovena celková cena chyb podle kategorie
- analyzují se kategorie s nejdražšími chybami
- snaha modifikovat proces, aby se eliminovala frekvence výskytu chyb v této kategorii

Metriky pro zdroje

- charakteristiky personálu
  - produktivita,
  - velikost týmu, rozsah komunikace,
  - zkušenosti,
- charakteristiky výpočetních systémů a softwarových prostředků
  - kapacita pamětí, taktovací frekvence procesoru,
  - dostupnost, použitelnost vývojových softwarových prostředků.

Testové metriky

Doporučuje se vytvořit informační systém výsledků testů.
Při vyhodnocování takto získaných dat je vhodné sledovat:

- trendy v počtech zjištěných selhání systému,
- doby potřebné pro odstranění příčin selhání.

Testování:

- u výrobce - alfa testování,
- u vybraných zákazníků - beta testování.

Testové metriky

Doporučuje se vést záznamy, které obsahují:

- počet modulů (komponent) modifikovaných při vývoji/změně,
- počet chyb odstraněných v dané etapě,
- průměr chyb na modul,
- počet změněných příkazů,
- průměrnou dobu na lokalizaci a odstranění chyby.

Testové metriky

- druhy a frekvence selhání systému způsobené příčinami podle členění:
  - chyba specifikací,
  - chyba návrhu,
  - chyba kódování,
  - selhání hardware,
  - chyba v reakci softwaru na selhání hardwaru.
- výčet modulů s největším / nejmenším počtem defektů,
- výčet modulů, které jsou nejsložitější

  př.: stanovená metrika nabývá extrémních hodnot nebo překračuje stanovenou hodnotu.

Pozn.: Je vhodné zavést automatické generování těchto metrik, aby se omezil vliv chybných nebo zastaralých údajů.

Metody odhadu pracnosti, softwarové metriky

Metrika - vyjádření vztahů mezi charakteristikami projektu a jeho náročností
Metrologická charakteristika – charakteristika, která může ovlivnit výsledky měření.
(ČSN EN ISO 9000 – Systémy managementu kvality – Základní principy a slovník) *Březen 2016*

- projektové metriky:
  - měří veličiny spojené s projektem jako celkem,
  - jedna projektová metrika má jednu hodnotu pro jeden projekt, hodnota se ale může v průběhu projektu měnit,

Metody odhadu pracnosti, softwarové metriky

- metriky aktivit, činností:
  - jsou spojeny s dílčími činnostmi pracovního postupu,
  - každá činnost může mít svoji metriku (vzorec).
- význam metrik - solidní podklady pro:
  - tvorbu plánu,
  - zdůvodnění rezistence vůči krácení harmonogramu.

Členění metod odhadu

- Empirické metody na základě analyzovaného množství projektů:
  - Funkční body (Albrecht 1979, IFPUG 1994),
  - COCOMO (Boehm 1981, 1997)
- Metody založené na zkušenosti a vlastních historických datech:
  - Lines of Code (LOC),
  - Wideband Delphi (Boehm 1981),
  - PROBE (Proxy Based Estimation - Humprey 1996),
  - Vážený průměr (WAVE).

Function Point Analysis (FPA)

- "top-down" model pro odhad pracnosti,
- měří na základě komplexnosti a velikosti vyvíjeného systému,
- hodnoty jsou počítány dle funkčních a technických charakteristik a podle vývojového prostředí,
- k reálnému využití odhadu pracnosti jsou nutná historická data,
- existuje více variant výpočtu:
  - Albrecht
  - Mark II
- existuje mezinárodní sdružení uživatelů metody FPA.

  The International Function Point Users' Group (IFPUG)

Function Point Analysis (FPA)

- Co je FPA?
  - Je to metoda objektivního měření velikosti vyvíjeného IS na základě jeho:
    - rozsahu,
    - složitosti,
    - specifických vlastností.
- Vlastnosti FPA:
  - nezávislost výsledku na implementačních podmínkách,
  - nezávislost výsledku na podmínkách vývoje systému (schopnosti lidí, týmu, specifické podmínky a omezení projektu atd.)

Function Point Analysis (FPA)

- možnost porovnání výsledku s výsledky jiných projektů,
- možnost kontinuálního zlepšování odhadů na základě svých i cizích zkušeností.

- K čemu slouží FPA?
  - odhadování pracnosti při plánování projektu,
  - sledování a předvídání změn v průběhu projektu,
  - následné zjišťování produktivity práce v projektu (porovnání s jinými),
  - následné zjišťování efektivnosti použitých technik a nástrojů atd.

Function Point Analysis (FPA)

Postup výpočtu:

1. výpočet hrubých funkčních bodů - podle typů a složitosti funkcí a datových sad systému,
2. úprava hrubých funkčních bodů - na základě vah, stanovených pro jednotlivé typy funkcí a datových sad a jejich složitost,
3. zjištění stupňů vlivu - jednotlivých specifických vlastností vyvíjeného IS,
4. výpočet **Faktoru úpravy hodnoty** - úpravou součtu jednotlivých stupňů vlivu,
5. výpočet celkového počtu funkčních bodů - pomocí **Faktoru úpravy hodnoty**.

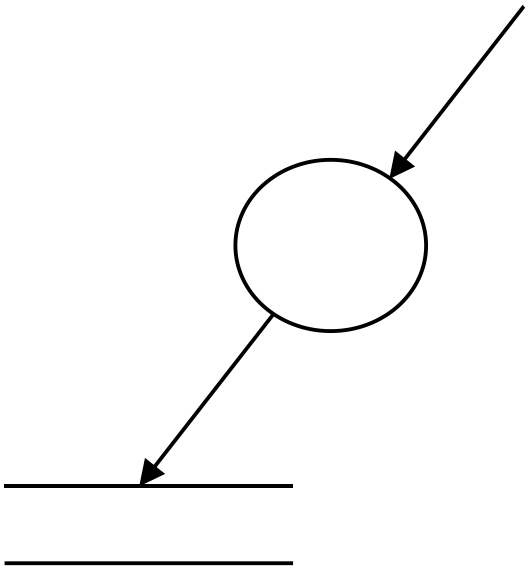
1. Výpočet hrubých funkčních bodů

a) vstupní funkce,
b) výstupní funkce,
c) dotazovací funkce,
d) interní logická datová sada,
e) externí vazební datová sada.

1. Výpočet hrubých funkčních bodů

a) vstupní funkce:

- každý přesun dat z vnějšku do systému:



1. Výpočet hrubých funkčních bodů

a) vstupní funkce:

- vstupní formuláře:
  - terminálové obrazovky,
  - přesuny dat z jiných aplikací,
  - čtení z datové základny jiné aplikace,
- jde o data, která:
  - nejsou produktem systému,
  - jsou vzájemně unikátní (buď jsou různého formátu, nebo jsou různě zpracovávána)

1. Výpočet hrubých funkčních bodů

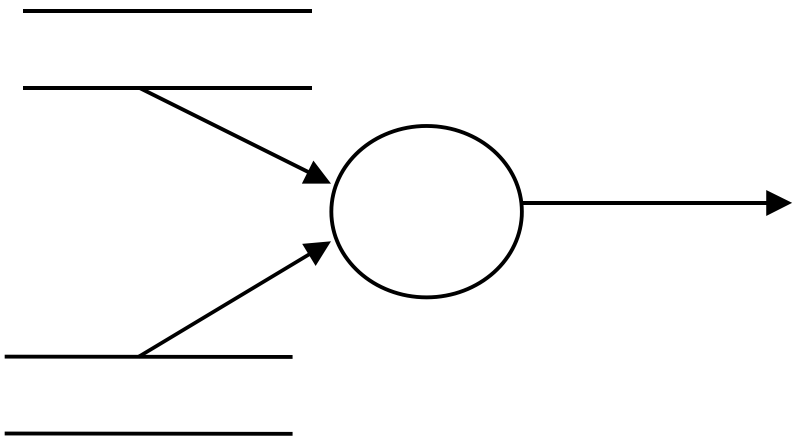
a) vstupní funkce:

složitost vstupní funkce				
Pa(j), Pa(p), Pa(s)		počet položek		
		1 - 4	5 - 15	16 a více
počet datových sad	0-1	jedn.	jedn.	prům.
	2	jedn.	prům.	slož.
	3 a více	prům.	slož.	slož.

1. Výpočet hrubých funkčních bodů

b) výstupní funkce:

- každý výstup dat ze systému,



1. Výpočet hrubých funkčních bodů

b) výstupní funkce:

- tiskové výstupy (reporty)
  - výstupy na obrazovku
  - zprávy obsluze
  - přesuny dat do jiných aplikací
- jde o data, která:
  - jsou produktem systému,
  - jsou vzájemně unikátní (buď jsou různého formátu, nebo jsou různě zpracovávána).

1. Výpočet hrubých funkčních bodů

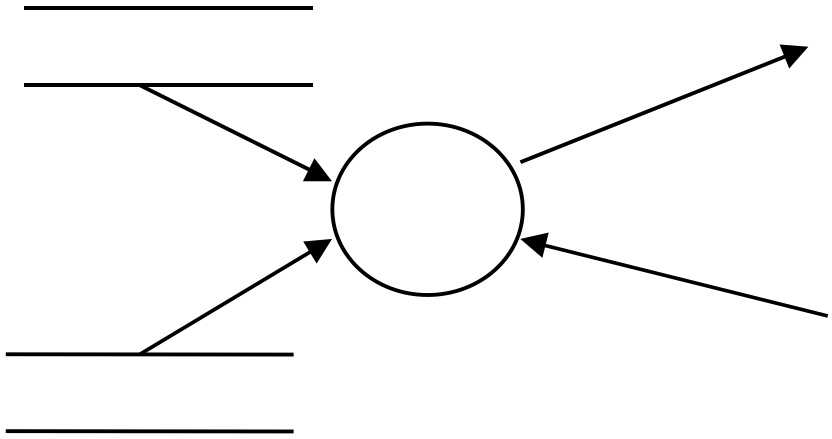
b) výstupní funkce:

složitost výstupní funkce				
Pb(j), Pb(p), Pb(s)		počet položek		
		1 - 5	6 - 19	20 a více
počet datových sad	0-1	jedn.	jedn.	prům.
	2	jedn.	prům.	slož.
	3 a více	prům.	slož.	slož.

1. Výpočet hrubých funkčních bodů

c) dotazovací funkce:

- každá vstupně - výstupní kombinace v systému, která na základě požadavku produkuje výstup dat ze systému.



1. Výpočet hrubých funkčních bodů

c) dotazovací funkce:

- vstupující data slouží pouze ke specifikaci dotazu, nejsou nijak v systému ukládána ani zpracovávána,
- vystupují data, která:
  - jsou produktem systému,
  - jsou reakcí systému na vzájemně unikátní dotazy.

1. Výpočet hrubých funkčních bodů

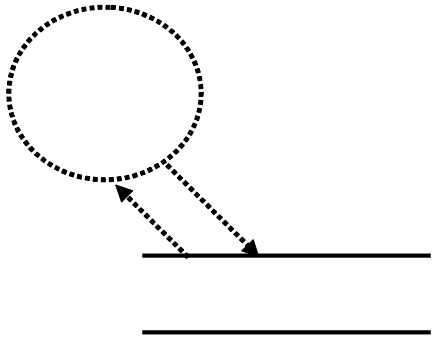
c) dotazovací funkce:

složitost dotazovací funkce				
Pc(j), Pc(p), Pc(s)		počet položek		
		1 - 4	5 - 15	16 a více
počet datových sad	0-1	jedn.	jedn.	prům.
	2	jedn.	prům.	slož.
	3 a více	prům.	slož.	slož.

1. Výpočet hrubých funkčních bodů

d) interní logická datová sada:

- každá vnitřní entita systému, t.j. entita, která je v systému vytvářena, aktualizována a rušena,



1. Výpočet hrubých funkčních bodů

d) interní logická datová sada:

- každý vnitřní datový sklad systému, tedy takový, který je systémem vytvářen, aktualizován a rušen,
- nepatří sem technologické a implementační datové sady:
  - např. pomocné soubory pro třídění dat pro výstup,
  - sady, tvořící rozhraní k jiným systémům.

1. Výpočet hrubých funkčních bodů

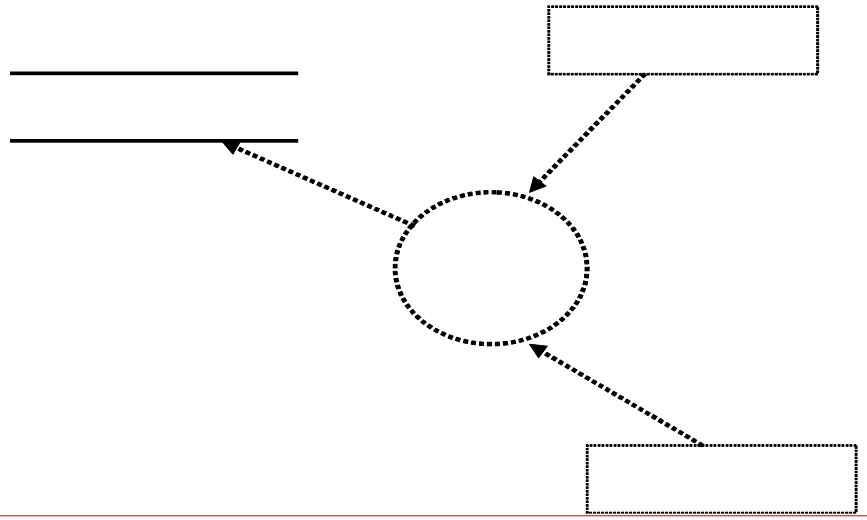
d) interní logická datová sada:

složitost interní logické datové sady				
Pd(j), Pd(p), Pd(s)		počet položek		
		1 - 19	20 - 50	51 a více
počet vazeb na jiné sady	1	jedn.	jedn.	prům.
	2 - 5	jedn.	prům.	slož.
	6 a více	prům.	slož.	slož.

1. Výpočet hrubých funkčních bodů

e) externí vazební datová sada:

- každá datová sada, vytvářená systémem pro potřebu rozhraní na jinou aplikaci,
- každá datová sada, vytvářená jinou aplikací pro potřebu rozhraní na systém.



1. Výpočet hrubých funkčních bodů

e) externí vazební datová sada:

- patří sem:
  - tzv. "externí datastore" (coby rozhraní od jiné aplikace),
  - technologické datové sady, požadované jinými aplikacemi ve smyslu rozhraní.

1. Výpočet hrubých funkčních bodů

e) externí vazební datová sada:

složitost externí vazební datové sady				
Pe(j), Pe(p), Pe(s)		počet položek		
		1 - 19	20 - 50	51 a více
počet vazeb na jiné sady	1	jedn.	jedn.	prům.
	2 - 5	jedn.	prům.	slož.
	6 a více	prům.	slož.	slož.

2. Úprava hrubých funkčních bodů

Váhy jednotlivých funkcí a datových sad a - e (j, p, s)

	jednoduchá	průměrná	složitá
Vstupní funkce a	**3**	**4**	**6**
Výstupní funkce b	**4**	**5**	**7**
Dotazovací funkce c	**3**	**4**	**6**
Interní logická datová sada d	**7**	**10**	**15**
Externí vazební datová sada e	**5**	**7**	**10**

Součet hrubých funkčních bodů (SHFB)

SHFB = 3 x Pa(j) + 4 x Pa(p) + 6 x Pa(s) +

+ ..... obd. dle tabulky pro Pb, Pc, Pd, Pe.

3. Zjištění stupňů vlivu

Stupně vlivu jednotlivých specifických faktorů	
neexistuje nebo nemá vliv	0
nepodstatný vliv	1
mírný vliv	2
průměrný vliv	3
významný vliv	4
velmi silný vliv	5

3. Zjištění stupňů vlivu

	Specifické faktory aplikace	Váha (0-5)
1	Data používaná aplikací jsou přenášena pomocí telekomunikačních nástrojů	
2	Součástí aplikace je distribuce funkcí	
3	U aplikace je důležitý výkon (ve smyslu doby odezvy, nebo průchodnosti aplikace)	
4	Je požadováno (očekáváno) značné přizpůsobení konfigurace	
5	Aplikace je charakteristická vysokým stupněm transakcí	
6	Přímý vstup dat	
7	Míra využití uživatelem (složité dotazy)	

3. Zjištění stupňů vlivu

	Specifické faktory aplikace	Váha (0-5)
8	Přímé opravy dat	
9	Složitost zpracování	
10	Je kladen důraz na znovupoužitelnost	
11	Snadná instalace	
12	Snadná provozovatelnost	
13	Nasazení pro více organizací (přizpůsobení spec. podmínkám)	
14	Přizpůsobivost uživateli	

3. Zjištění stupňů vlivu

	Specifické faktory aplikace	Váha (0-5)
15	Budování a udržování rozhraní na jiné aplikace	
16	Audit a ochrana dat systému	
17	Umožnění přístupu k datům pro cizí systémy	
18	Vývoj školících prostředků uživatele	
19	Zvýšené nároky na dokumentaci	
	celkem	SFV

Součet vah všech faktorů vlivu

4., 5. Výpočty

- Výpočet Faktoru úpravy hodnoty:

  FUH = 0.65 x 0.01 x SFV

- Výpočet celkového počtu funkčních bodů :

  FB = SHFB x FUH

- Vynaložená práce v pracovních hodinách (WH):

  WH = 54 x FP - 13.390

COCOMO Constructive Cost Model **Boehm 1981, 1996**

http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html
http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/cocomo81.html

Verze 1

- Definuje potřebné úsilí (pracnost) a čas.
- Vychází z velikosti programu v tisících řádků.
- Má tři úrovně podle toho, kolik podrobností o projektu máme k dispozici:
  - základní (hrubé odhady),
  - střední (bere v úvahu atributy projektu),
  - podrobná (rozdělení úsilí mezi jednotlivé etapy).

COCOMO Constructive Cost Model

Uvažuje 3 módy vyjadřující stupeň složitosti systému:

- organický mód - relativně malý softwarový tým pracuje na aplikaci ze známé problémové domény, ve známém prostředí, předpokládá se:
  - malé náklady na komunikaci,
  - stabilní hardware,
  - známé algoritmy,
  - možnost dekompozice problému na menší ucelené části,
  - velikost do 50 KLOC.

COCOMO Constructive Cost Model

- přechodný mód, předpokládá se:
  - vyšší požadavky na komunikaci, čas,
  - velikost do 300 KLOC.
- vázaný mód, předpokládá se:
  - různá ohraničení projektu,
  - krátké termíny dodávky,
  - neustálé změny v požadavcích,
  - zásahy zákazníka,
  - neznámá problémová oblast,
  - aplikace mají ostré požadavky na časovou odezvu,
  - často jde o interaktivní, řídící a operační systémy.

COCOMO Constructive Cost Model

Verze 2

- bere v úvahu změněné podmínky při tvorbě software:
  - uvažuje různé modely životního cyklu softwaru,
  - různé přístupy k vývoji software,
- Výsledkem je nové dělení úrovní COCOMO:
  - analýza aplikace (počet objektů),
  - návrh aplikace (počet funkčních bodů),
  - implementace (řádky textu programu).
- Model se vyvíjí, konkrétní tvary vztahů a koeficientů se mění.

Lines of Code (LOC)

- metrika absolutní velikosti SW
- varianty: KLOC (kilo-LOC), SDLOC (source delivered LOC)
- různé způsoby měření v závislosti na tom, co se počítá jako řádka; např. COCOMO-II:
  - výkonné příkazy, hranice bloků,
  - programované ručně, konvertované,
  - převzaté z dřívější práce, knihoven a jiných komponent.

Lines of Code (LOC)

- deklarace, direktivy překladače, komentáře,
- generované automaticky,
- v zakoupeném sw, knihovnách dodaných se systémem,
- výhody:
  - přesnost,
  - jednoznačnost,
  - snadná spočitatelnost
- problémy:
  - použitelné pozdě v životním cyklu projektu, bez vazby na produktivitu,
  - jazyková závislost (např.).

  př.: tentýž program má v JSA 10KLOC, v Pascalu 3KLOC.

(Wideband) Delphi

Skupinová metoda odhadu týmem expertů s koordinátorem:

1. předložení specifikace a tabulky pro odhad expertům,
2. skupinová diskuse (vč. koordinátora) nad okolnostmi ovlivňujícími odhad,
3. anonymní vyplnění tabulek experty,
4. sumarizace výsledků koordinátorem:
   - vyznačení odhadů a mediánu,
5. skupinová diskuse nad původem největších rozdílů v odhadech,
6. iterace předchozích třech bodů dle potřeby.

(Wideband) Delphi - charakteristika

- skupinová:
  - odstranění osobních předpojatostí a preferencí,
- iterativní, zpětná vazba:
  - odstranění vlivu výrazně odlišných odhadů,
- anonymní:
  - odstranění vlivu silných jedinců nebo politických tlaků,
- diskuse:
  - vyjasnění problému, odstranění vlivu neznalosti projektu.

Vážený průměr (Weighted AVErage WAVE)

- metoda váženého průměru slouží k odhadu pracnosti, resp. trvání projektu,
- vážený průměr je vhodná metoda pro počáteční fáze odhadu projektu,
- WAVE se počítá na základě kombinace různých odhadů jednoho projektu,
- WAVE je velice snadná metoda,
- existuje více vzorců pro výpočet.

Vážený průměr (Weighted AVErage WAVE)

Princip výpočtu:

Proměnné: A - optimistický odhad
B - realistický odhad
C - pesimistický odhad

Vzorce:

M (výsledná hodnota) = (A + 3B + C) / 5

S (standardní odchylka) = (C - A) / 5

D (spodní hodnota intervalu rozptylu) = M - 2 x S

E (horní hodnota intervalu rozptylu) = M + 2 x S

Platí: je 95% pravděpodobnost, že výsledek bude v intervalu (D ; E).

Pozn.: Hodnota S by neměla být větší než M / 20 (tzv. pravidlo palce), jinak je třeba odhad zpřesňovat zpodrobňováním rozkladu činností.

Programová podpora odhadování pracnosti softwarových produktů - Optimize

Popis produktu:

- nástroj Optimize plně podporuje metodiku Object Metrix,
- ObjectMetrix pracují s následujícími faktory, které ovlivňují délku trvání projektu a náklady na projekt vynaložené:
  - Rozsah projektu
    - odhad je ovlivňován počtem subsystémů, tříd, typových úloh, komponent a rozhraní, ze kterých se systém skládá,
    - každému prvku lze přiřadit atributy a jejich hodnoty jako je složitost, obecnost, možnost využití stávajících komponent apod.

Programová podpora odhadování pracnosti softwarových produktů - Optimize

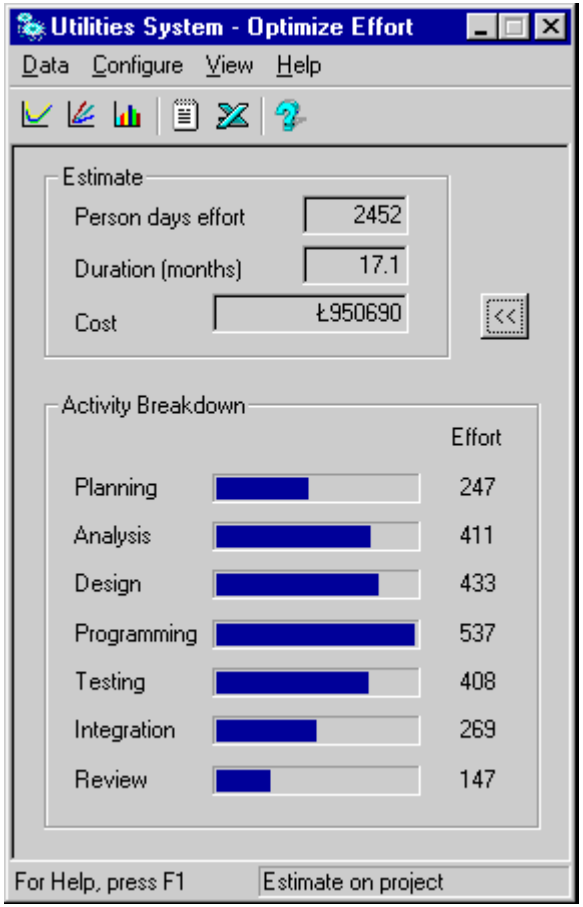
- Vývojový tým
  - je potřeba sledovat počet týmů a počet členů těchto týmů, role pracovníků a úroveň jejich dovedností,
- Technologie
  - nástroj pracuje s různými typy technologií (jako například vývojové prostředí) a jedním ze sledovaných prvků projektu je míra vlivu technologie na projekt,
- Předpokládaný průměrný počet pracovních dní v měsíci (nebo dní, kdy jsou zdroje skutečně zapojeny do práce).

Programová podpora odhadování pracnosti softwarových produktů - Optimize

- Předpokládaný průměrný počet hodin za den, kdy zdroje na projektu skutečně pracují.
- Hodinová sazba interních a externích zdrojů.

- Optimize umožňuje vytvářet odhad za celý projekt nebo za jednotlivé vybrané prvky.
- Výsledný odhad se člení podle fází projektu na plánování, analýzu, návrh, implementaci, testování a integraci řešení.
- Uživatelé mají možnost měnit metriky a parametry odhadu dle svých požadavků a zkušeností.

Programová podpora odhadování pracnosti softwarových produktů - Optimize



Programová podpora odhadování pracnosti softwarových produktů - Optimize

- Informace o rozsáhlých projektech mohou být zadávány ručně nebo s využitím některých CASE nástrojů, jako například nástroje Rational Rose.
- Také je možné kombinovat obě metody vkládání informací.
- Samotné odhady jsou zobrazeny v nástroji Optimize, nebo mohou být vygenerovány do připravených reportů, eventuelně do produktu Microsoft Excel.

Měření v softwarovém projektu