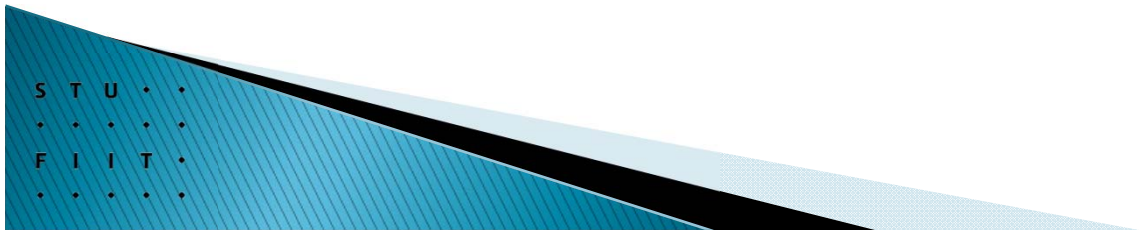
Odhady v softvérových projektoch

1. časť



Bol raz jeden podnik a v ňom štyria zamestnanci, ktorí sa volali *Každý*, *Niekto*, *Hocikto* a *Nikto*. Jedného dňa bolo treba splniť dôležitú úlohu a *Každý* si bol istý, že to *Niekdo* urobí. Mohol to urobiť *Hocikto*, ale *Nikto* to nerobil. *Niekto* sa nahneval, pretože to predsa bola práca pre *Každého*. *Každý* si myslel, že by to mohol urobiť *Hocikto*, ale *Nikto* si neuvedomil, že to *Každý* neurobí. Nakoniec *Každý* obviňoval *Niekoho*, že *Nikto* neurobil to, čo mohol urobiť *Hocikto*.

Odhady - úvod

- **ODHAD – najpravdepodobnejšia hodnota, ktorú bude nadobúdať nejaká (merateľná) veličina (ukazovateľ)**
- podstatná vec pri plánovaní
- aké úsilie (prácnosť) bude treb
  určitej činn
- ko
- ak

„A leading executive was once asked what single characteristic is most important for a project manager. His response: '…a person with the ability to know what will go wrong before it actually does…' We might add: '…and the courage to estimate when the future is cloudy…'"

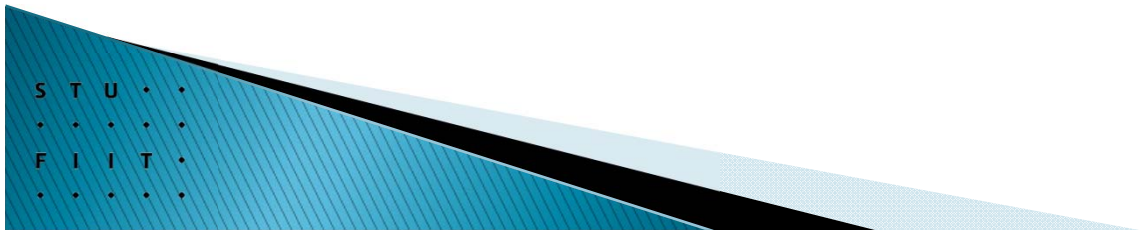
(Pressman, 1992)

Typy odhadov

- **empirický**
  - analýza údajov a stanovenie vzťahov
- **analogický**
  - použitie známych hodnôt z minulosti (z predošlých projektov)
- **teoretický**
  - na základe numerického modelu (overuje sa empiricky)
- **heuristický**
  - doplnenie predošlých metód, použitie expertov

Odhady

- náklady a zdroje treba poznať (odhadnúť) na začiatku projektu, ale presne ich určiť možno až na konci projektu
- odhad -> pravdepodobnosť, nepresnosť, chyba
  - dobrý odhad <= 20% nepresnosť
- cieľ odhadu – najpravdepodobnejšia budúca hodnota (trvania, zdrojov, nákladov)
- splnenie odhadu **NIE JE cieľom projektu**
- **odhad NIE JE hádanie**
  (teda má byť podložené faktami!)



Procesy odhadu

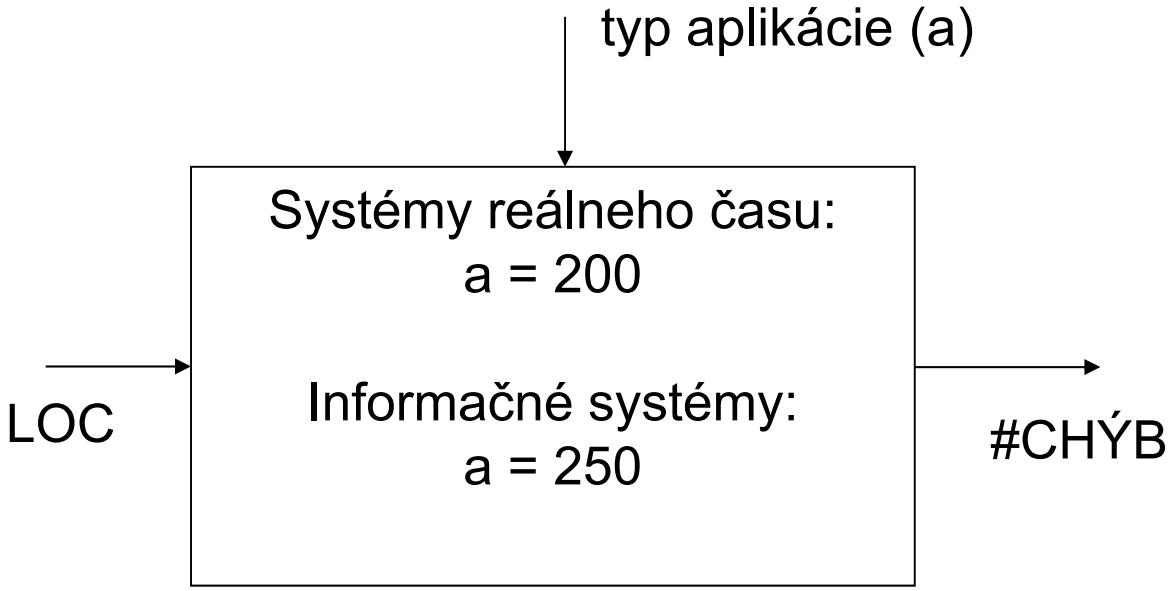
- **proces predpovede (odhadu) zahŕňa**
  1. výber veličín (charakteristík), ktoré sa budú odhadovať
  2. vykonanie odhadu (predpovede)
  3. vyhodnotenie správnosti odhadu (spätná väzba)
- **metódy odhadu**
  - **zhora nadol**
    - jednoduchšia
    - riziko podcenenia drobných problémov („implementačné detaily“)
  - **zdola nahor**
    - zložitejšia – treba už mať nejakú dekomponovanú štruktúru
    - riziko zanedbania prierezových (integračných) činností

Presnosť odhadu

- ovplyvňuje
  - **veľkosť projektu** (úsilie, prácnosť)
    - čím väčší projekt, tým nepresnejší odhad
  - **zložitosť projektu**
    - čím zložitejší, tým nepresnejší odhad
  - **štruktúrovanosť projektu**
    - čím štruktúrovanejší, tým presnejší

Techniky odhadu

- **dekompozícia**
- **modelovanie**
  - parametrické modely
  - neparametrické modely (NeuNet, strojové učenie)

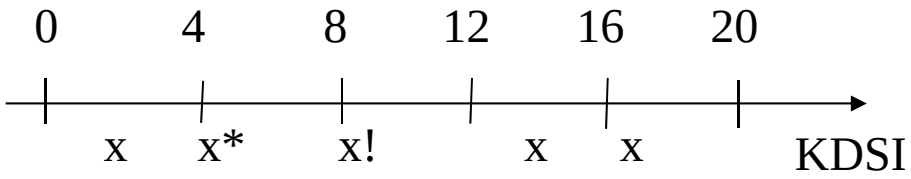


#CHÝB = LOC/a

Techniky

- **posúdenie exp**
  - použitie analóg
  - Delphi metóda
  1. Experti dosta
  2. *Diskusia* o pr
  3. Vytvorenie in
  4. Zápis odhado (identifikujú s
  5. Distribuovani
  6. Odhad veľkos Prehodnoteni kým nedôjde

Odhad veľkosti vytváraného systému
Projekt: bankový softvérový systém
Expert: DC
Kolo: 1
Dátum: 2.8. 2008



Kľúč: x … odhad
x* … váš odhad
x! … stredný odhad

- **dodávateľské možnosti**
  - „nákladová“ metóda – podľa dostupnosti zdrojov

Techniky odhadu III

- **„najlepšia“ technika**

– neexistuje!

  - len viac alebo menej vhodná, ich kombinácia

Náklady na softvérový projekt

- náklady na softvér a hardvér (+ údržba, upgr)
  - OS, programovacie prostredie, CASE, podpora, ...
- cestovné, školenia, tréningy
- **úsilie** (vlastná práca pracovníkov) – prácnosť
  - platy a odmeny
  - réžia – prevádzka, priestory, podporný personál, komunikácia, doplnkové služby (knižnica, copy-service, ...), poistenie, ODVODY
    - **réžia** = vysoká položka! : až 2-násobok platov

Náklady, cena

- Na náklady vplýva
  - veľkosť (zložitosť) projektu
  - schopnosti a skúsenosti tímu
  - používanie podporných (softvérových, hardvérových) prostriedkov
  - proces (postup) vývoja
  - ohraničenia a požiadavky na výrobok (softvér)
    - spoľahlivosť, rýchlosť, pamäťové nároky, ...

- **CENA =**

  náklady + zisk (strata?)

Cena

- vplýva na ňu
  - podmienky zmluvy (zdrojový kód, záruky, údržba,...)
  - vytvorenie príležitosti na trhu (preniknutie na trh)
  - neurčitosť odhadu nákladov („bezp. prirážka“?)
  - pravdepodobnosť zmeny požiadaviek (zákazníka)
  - finančná situácia a stratégia dodávateľa (dumping)
  - finančné možnosti zákazníka
    - podľa toho sa prispôsobí rozsah a funkčnosť výsledného výrobku (špecifikácie)

Android Code 2012: prezentácie pre developerov na STU v Bratislave (24.október 2012)

17. OKTÓBER 2012 | **MOJANDROID.SK**

Vrámci pomoci a propagácie našej súťaže Android Code 2012 sme sa rozhodli spolupracovať aj s univerzitam
Prvé stretnutie bude v priestoroch Slovenskej Technickej Univerzity v Bratislave. Pripravili sme pre vás zaujímav
program a možnosť prísť si vypočuť inšpiratívne prednášky. Stretneme sa v stredu **24.októbra 2012 od 14,30**
aule BC300.


CODE 2012
Hlavný organizátor
MÔJ ANDROID.SK
V spolupráci s
STU
Súťaž o najlepšie slovenské Android aplikácie
12:03
12:45
12:45
Vyhraj tieto telefóny a finančné odmeny za tvoju aplikáciu
Príď si vypočuť zaujímavé prezentácie od úspešných developerov a manažérov:
Rasťo Kulich
Google Slovensko
Michal Štencl
Sygic
Martin Adámek
Android developer
Streda 24.október 2012 o 15,00 hod
Aula prof. Ľudovíta Kneppa FEI STU v Bratislave, Ilkovičova 3 (BC300)
Hlavní partneri
Partneri
IBM
SmartAds
WebSupport

STU FIIT

Produktivita

- P = množstvo vyprodukovaného výstupu na jednotku času
- dá sa merať ako
  - veľkosť výstupu (počet riadkov zdrojových textov)
  - množstvo funkčnosti (funkčné body, prípady použitia)
- za deň/mesiac

- produktivita vs. kvalita
  - treba zohľadniť celý životný cyklus – testovanie, dokumentácia

Som produktívny, ak vyprodukujem veľa softvéru?

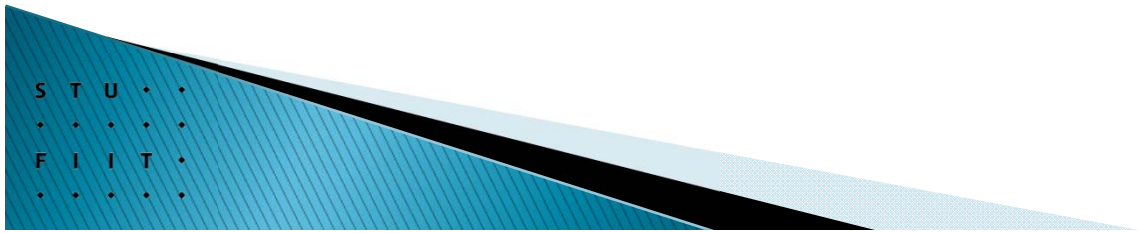
Produktivita

- produktivitu jednotlivých pracovníkov („sv-ing-ov“) **ovplyvňujú**
  - znalosti, skúsenosti (relevantné pre daný projekt)
  - kvalita nastavených procesov vývoja softvéru – review, model životného cyklu)
  - výpočtové prostredie – spoľahlivosť, výkonnosť
  - problematika – veľkosť, zložitosť a novosť
  - zdroje – technológie, nástroje (CASE, ...)
- **rozdiely v produktivite**
  - **až 10násobné**
- menšia produktivita ?= kvalitnejší a lepšie udržiavateľný kód?
- **produktivita je významne záležitosť manažmentu**

Odhad rozsahu (veľkosti) softvéru

- **manažéri ľahšie (presnejšie) odhadnú veľkosť úsilia (prácnosť) ako rozsah softvéru**

- metriky rozsahu (veľkosti) softvéru
  - dĺžka (zdrojového) textu programu/produktu
  - funkčné body (function points)
  - body prípadov použitia (use case points)

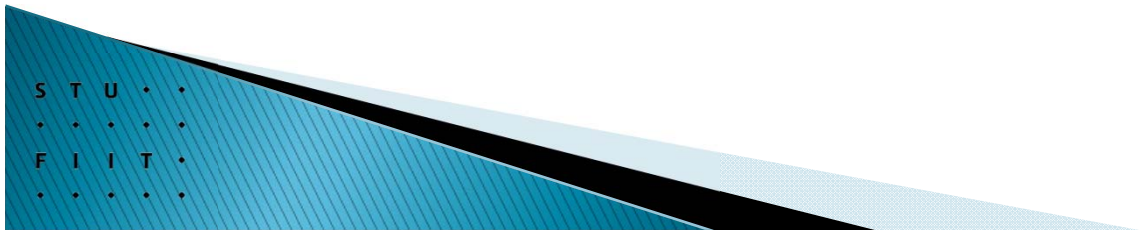


Metriky

- CLOC
- ELOC
- LOC
- LLOC
- SLOC
- DSI. KDSI
- NCLOC
- NOP
- NOC
- NOM

Dĺžka textu programu

- LOC – Lines of Code – počet riadkov kódu (bez komentárov a prázdnych riadkov)
- SLOC – Source Line of Code (logický, fyzický)
- DSI, KDSI – Delivered Source Instructions
- počet oddeľovačov ;
- počet znakov
- počet bytov (súboru s uloženým programom)
- počet strán výpisu



Dĺžka textu programu – SLOC

Príklad

```
for (i=0; i<100; ++i) printf("hello"); /* How many lines of code is this? */
```

```
for (i=0; i<100; ++i)
{
printf("hello");
} /* How many source lines of code is this? */
```

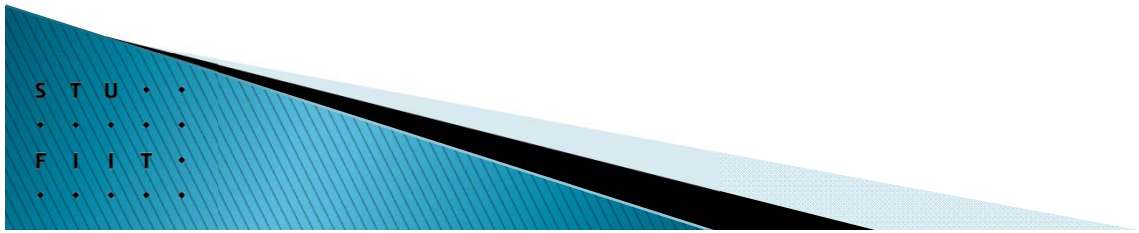
softvér na spočítanie (normalizáciu) SLOC

Dĺžka textu ako metrika

- výhody
  - ľahko a jednoducho sa spočíta a používa
  - možnosť použitia množstva historických dát
  - možnosť automatického merania a sledovania
- nevýhody
  - špecifická pre rôzne jazyky
  - ignoruje kvalitu kódu
  - závisí od progr. štýlu resp. predpísaných (projektových) štandardov a odporúčaní
  - kód – výrobok (ako výsledok projektu) môže predstavovať iba menšiu časť nákladov na výrobok (30%)
  - GUI programming tools – programátor priamo žiaden kód nepíše, ale „vykliká ho“ s použitím výkonného a mohutného nástroja/prostredia s bohatými knižnicami

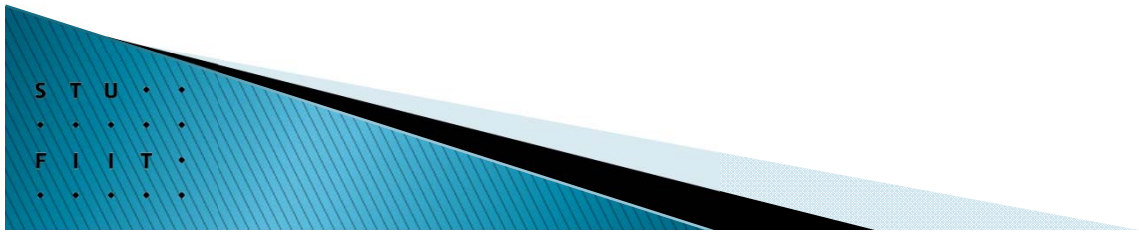
Funkčné body

- špecifikácia požiadaviek (GUI aplikácie)
  - počet a rozsah služieb (funkcií) prístupných (viditeľných) používateľovi interaktívne
  - sú pomerne presne známe už na začiatku projektu
- používa sa na odhad zložitosti (veľkosti) softvéru pre interaktívne (db) aplikácie
- nezávisí od programovacieho jazyka



Funkčné body – postup – krok 1

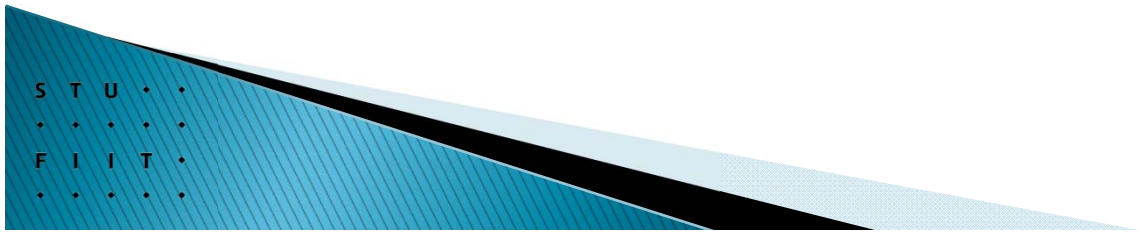
1. **stanovenie charakteristík pre produkt**
   A. počet externých vstupov (od používateľa)
   B. počet externých výstupov (zostavy, obrazovky)
   C. počet logických externých súborov (db tabuľka, súbor, logické zoskupenie údajov)
   D. počet externých rozhraní (objekty na komunikáciu s okolím – konfiguračné súbory, ...)
   E. počet externých dopytov – vyžadujú db transakcie, hľadanie v texte, db query, ...



Funkčné body – postup– krok 2

2. **klasifikácia zložitosti** každej z charakteristík do 3 úrovní
   - jednoduchá
   - stredná
   - zložitá

   určenie zložitosti – rôzne metódy, odhady, pomôcky, podklady – napr. frekvencia prístupov, typy záznamov



Funkčné body – postup– krok 3

3. výpočet UFC – neupravený funkčný bod (Unadjusted Function Count)

$$UFC = \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} c_{ij}$$

- i – typ charakteristiky (A, B, C, D, E)
- j – zložitosť (jednoduchá, stredná, zložitá)
- cij – počet (odhadnutá, nameraná hodnota)
- wij – váha z tabuľky

	jednoduchá	stredná	zložitá
A	3	4	6
B	4	5	7
C	7	10	15
D	5	7	10
E	3	4	6

Funkčné body – postup– krok 4

4. Úprava funkčných bodov

- obodovanie 14 technických charakteristík (technickej zložitosti) celého projektu (body od 0 do 5) – podľa tabuľky
- výpočet hodnoty faktora zložitosti CAF – Complexity Adjustment Factor

$$CAF = 0.65 + \left(0.01 * \sum_{i=1}^{14} v_i \right)$$

Funkčné body – krok 4

1. Vyžaduje sa dátová komunikácia? (batch mód, nepriamy vstup údajov, interaktívny mód)
2. Sú v systéme distribuované funkcie?
3. Je výkonnosť systému kritická? (odozva a priepustnosť systému)
4. Vyžaduje systém spoľahlivé zálohovanie a obnovu?
5. Bude sa program vykonávať v existujúcom a používanom operačnom prostredí?
6. Vyžaduje systém on-line vstup údajov?
7. Vyžaduje on-line vstup údajov zložité transakcie (viaceré obrazovky a operácie)?
8. Umožňuje sa on-line modifikácia údajov?
9. Je vnútorné spracovanie zložité?
10. Sú moduly navrhnuté pre znovupoužitie?
11. Bude treba viacnásobnú inštaláciu v rôznych organizáciách?
12. Uvažuje sa jednoduchosť inštalácie?
13. Uvažuje sa jednoduchosť použitia?
14. Uvažuje sa zapracovanie budúcich zmien?

Funkčné body – postup– krok 5

5. Výpočet DFP (Delivered Function Points)

DFP = CAF * UFC

voliteľne – doplniť optimistické a pesimistické odhady (z predošlých projektov)

DFP sa vezme ako najpravdepodobnejšia hodnota (m), a výsledok (odhad) sa vypočíta ako

$$E = \frac{p + 4*m + o}{6}$$

kde o – optimistický odhad,

p – pesimistický odhad

Funkčné body– čo ďalej

- Function points II
  - počítajú sa na základe transakcií v aplikácií
  - doplnenie charakteristík pre iné druhy (než GUI interaktívne) aplikácií (sw)
- Feature Points
  - modifikácia pre neiteraktívne systémy (OS, vnorené systémy)
  - 6. charakteristika – algoritmus (stredná váha 3)
  - logické interné systémy – váha z 10 na 7

Funkčné body – pros vs. cons

- Výhody
  - dá sa odvodiť zo špecifikácie (hrubej)
  - nezávislosť od konkrétneho progr. jazyka (tabuľky prevodu FP na LOC)
- Nevýhody
  - spolieha sa príliš na prvotné používateľské špecifikácie rozsahu (vôbec nemusia byť spoľahlivé)
  - subjektívnosť určovania charakteristík a ich váhovania (stanovenia zložitosti)
  - nedá sa automatizovať
  - znovupoužiteľnosť softvérových súčiastok (kódu) – nevyjadrí skutočnú produktivitu (náklady)

Konverzná tabuľka LOC – FP

Jazyk	LVL	LOC/FP
Assembler	1	320
ArityPROLOG	5	64
C	2,5	128
C++	6	53
HTML	22	15

Jazyk	LVL	LOC/FP
Excel	57	6
SQL	25	13
Java	6	53
Prirodzený jazyk	0,1	3200
Visual C++	9,5	34

Úroveň jazyka v. produktivita – ilustratívny empirický príklad

Úroveň jazyka	Produktivita (FP za 1 mesiac)
1–3	5 – 10
4–8	10 – 20
9–15	16 – 23
16–23	20 – 32
24–55	30 – 50
nad 55	40 – 100

Body prípadov použitia (Use Case Points – UCP)

- veľkosť (zložitosť) systému sa odhadne na základe odhadu zložitosti prípadov použitia
- analogický postup výpočtu ako pri vypočte rozsahu pomocou funkčných bodov
- používa sa v OO aplikáciách, ktoré majú (roz/s)pracované prípady použitia – biznis logiku
- Gustav Karner - 1993

Body prípadov použtia - postup

1. Klasifikácia hráčov podľa zložitosti
2. Klasifikácia prípadov použitia podľa zložitosti
3. Výpočet neupravených bodov prípadov použitia
4. Úprava na základe charakteristík projektu

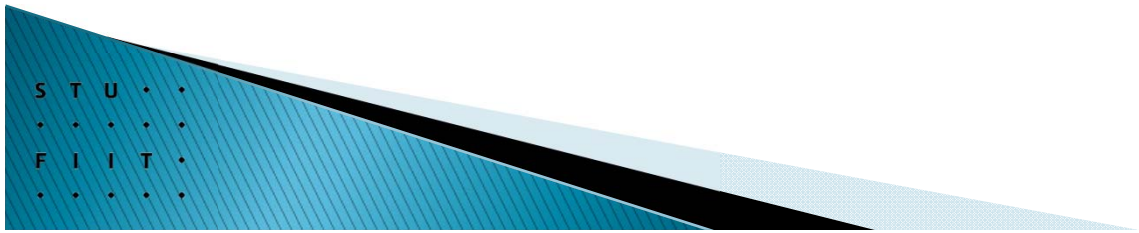
UCP = UUCP * TCF * ECF *(* PF)*

UUCP – neupravené UCP (Unadjusted UCP)

TCF – faktor technickej zložitosti (Technical Complexity Factor)

ECF – faktor zložitosti prostredia / projektu (Environment Complexity Factor)

PF – faktor produktivity (Productivity Factor)



UUCP – neupravené UC body

- **UUCP = UUCW + UAW**
- **UUCW -** ***Unadjusted Use Case Weight*** **–** neupravená váha prípadov použitia
- **UAW -** ***Unadjusted Actor Weight*** **–** neupravená váha hráčov - aktérov (actors)

UUCW - Unadjusted Use Case Weight

Typ prípadu použitia	*Opis typu prípadu použitia*	*Váha*
Jednoduchý	Jednoduché rozhranie pracujúce s jednou db-entitou; scenár má najviac 3 kroky (3 transakcie), implementácia - menej než 5 analytických tried	5 (1)
Stredný	Zložitejšie rozhranie pracujúce s 2 db-entitami; scenár má 4-7 krokov (4-7 transakcií), implementácia – 5-10 tried	10 (2)
Zložitý	Komplexné rozhranie pracujúce s 3 a viac db-entitami; scenár má viac než 7 krokov (transakcií), implementácia - viac než 10 tried	15 (3)

UAW - Unadjusted Actor Weight

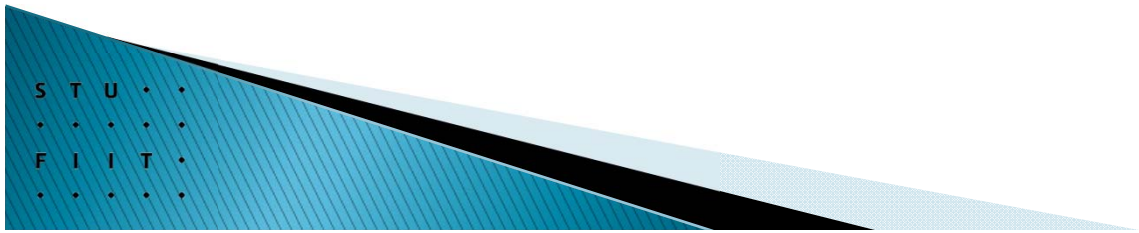
Typ hráča - aktéra (actor)	***Opis typu hráča - aktéra (actor)***	***Váha***
Jednoduchý	Aktér predstavuje iný systém komunikujúci prostredníctvom API	1
Stredný	Aktér predstavuje iný systém komunikujúci prostredníctvom protokolu (napr. TCP/IP)	2
Zložitý	Aktér je osoba (človek) interagujúci cez špecifický interface (GUI, web)	3

TCF – tabuľka charakteristík technickej zložitosti + váhy

1	Distributed system	2
2	Performance	1
3	End User Efficiency	1
4	Complex internal Processing	1
5	Reusability	1
6	Easy to install	0,5
7	Easy to use	0,5
8	Portable	2
9	Easy to change	1
10	Concurrent	1
11	Special security features	1
12	Provides direct access for third parties	1
13	Special user training facilities are required	1

TCF (Faktor technickej zložitosti) - výpočet

- každej charakteristike sa odhadne relevantnosť v projekte na škále od 0 (nerelevantné) do 5 (max. závažné)
- spočíta sa váhovaná celková hodnota cez všetky faktory – získa sa TF (TotalFactor – súhrnný faktor zložitosti)
- **TCF = 0,6 + (0,01*TotalFactor)**



ECF – faktory zložitosti prostredia / projektu

1.	Familiarity with UML	1,5
2.	Application Experience	0,5
3.	Object Oriented Experience	1
4.	Lead analyst capability	0,5
5.	Motivation	1
6.	Stable Requirements	2
7.	Part-time workers	-1
8.	Difficult Programming language	2
	Total Factor	*TotalFactor*

ECF = 1.4 + (-0.03*TotalFactor)

PF – faktor produktivity

- PF – priemerný počet človeko-hodín potrebných na realizáciu jedného prípadu použitia
  - v tom je zahrnutá nielen implementácia (kódovanie), ale aj návrh, testovanie, review, dokumentácia
- určí sa na základe údajov z predošlých projektov
- priemerná hodnota je 15-30 (čh/1UC)
- typická hodnota (default) je 20
- *vypočítané UCP potom vyjde v človeko-hodinách*

$$UCP = TCP * ECF * UUCP * PF$$

Body prípadov použitia – problémy a úskalia

- väčšinou vyjdú nadhodnotené čísla (v porovnaní s expertnými odhadmi)
- zvyčajne je nadhodnotený
  - počet krokov v scenári (transakcií)
  - počet hráčov-aktérov (zovšeobecniť do super-hráča)
- faktor produktivity môže byť ako-tak spoľahlivý iba na základe údajov z viacerých reálnych projektov
- UC (prípady použitia) musia byť vypracované (až 10-20% celkovej prácnosti projektu)
- pre detailné praktické plánovanie sú UC ako jednotky plánovania práce príliš veľké

Body prípadov použitia - výhody

- celý proces (postup výpočtu) sa dá automatizovať
- dobré oddelenie metriky rozsahu od veľkosti tímu
- možnosť stanovenie faktora produktivity pre danú organizáciu (vnútroorganizačná „norma“ pre implementáciu jedného UC)

Odhady nákladov

- cieľ
  - zistiť tie charakteristiky, ktoré vplývajú na náklady
  - nájsť vzťah medzi nimi a nákladmi
- **odhad vynaloženého úsilia (prácnosti)**

$$p * V$$ [človeko-mesiac]

kde p – koeficient produktivity (čím vyššia, tým menší)
V – veľkosť systému

Odhady nákladov

- Jednoduchšie projekty

$$úsilie = C * VM \quad * M_1 + M_2$$

- zložitejšie projekty

$$úsilie = C * VM^{b} * M$$

  - C – zložitosť systému
  - VM – metrika (veľkosť, rozsah)
  - b – exponenciálny koeficient (konštanta v daných vývojových podmienkach)
  - M – konšt. korekčný koeficient pre dané podmienky