Měření softwaru
Problémy používání SW metrik

Výsledek měření je metrika

- Bez měření nelze kvalitně řídit ani hodnotit kvalitu SW, atributy kvality mohou ale být různé
- Softwarová metrika je nějaký údaj, který lze nakonec převést na číselné hodnocení nějakého atributu softwaru
- DB softwarových metrik je paměť firmy a prostředek optimalizace softwarových procesů (procesů vývoje softwaru) a managementu (např. méně rizik při uzavírání smluv),
- Je součástí business intelligence SW firmy a přepoklad uplatnění moderních způsobů řízení, např. CMMI

Použití SW metrik

a) ***Výzkum:*** podklad pro **hledání metod realizace** softwarových produktů, které by přinesly podstatné zvýšení jeho kvality a snížení nákladů a doby vývoje a hlavně rozsahu prací při údržbě softwaru (výzkum metod a zákonitostí vývoje softwaru).

b) ***Normy:*** základ pro **stanovení technicko-ekonomických podkladů** pro řízení prací při tvorbě softwaru (normy pracnosti, odhady takových metrik, jako je pracnost či doba řešení) a uzavírání smluv (cena, termíny), předpoklad CMM

c) ***Kontrola kvality:*** prostředek **sledování spolehlivosti** softwaru při provozu a podklad pro řídící zásahy během údržby,procesy zajišťujíící kvalitu

d) **Operativa:**prostředek **sledování průběhu prací** při vývoji (dodržování termínů, procento testovaných komponent, trendy počtů chyb, počty nově zanesených chyb, komponenty s největším počtem chyb, atd.).

Použití SW metrik

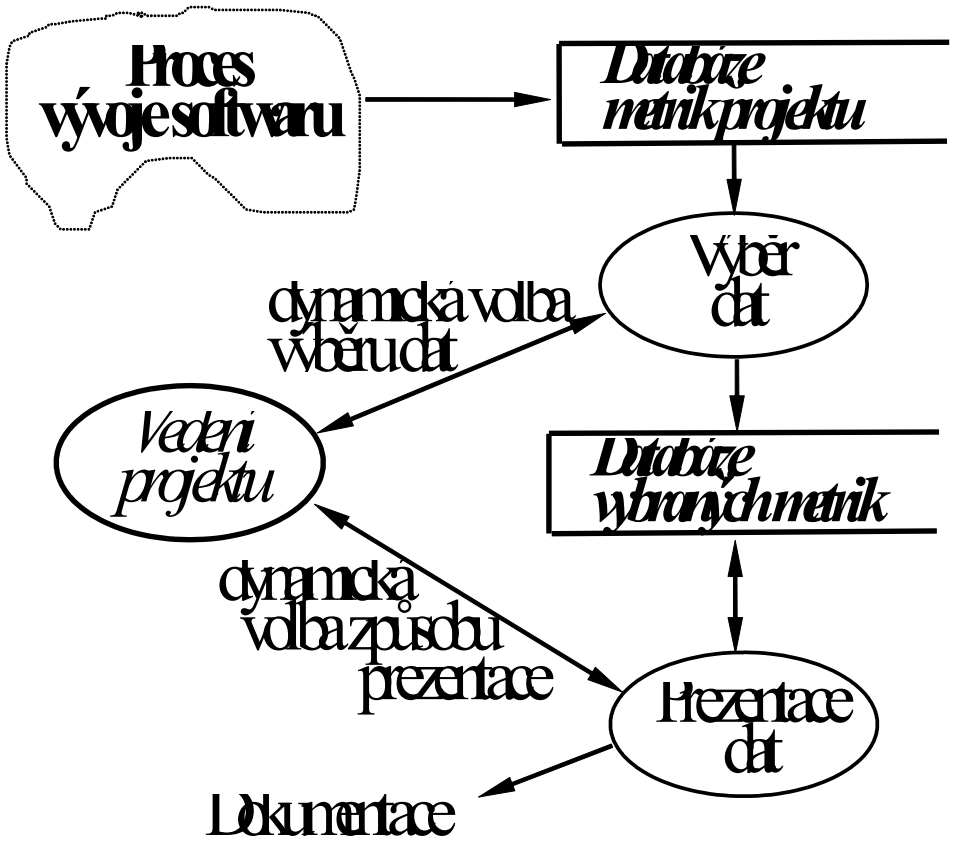
Výsledkem výzkumu SW metrik jsou metodiky vývoje (SOA, OO, strukturované programování, znalosti a vlivu kvality specifikací atd.) a metodiky odhadu pracnosti a doby řešení COCOMO, funkční body, a filosofií SOA ITIL

ISO 9126, ISO 250xx, ISO 12207, ISO 20000

ISO 250XX, **Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Planning and management**

ISO 270XX Information Security Management System

Infrastruktura sběru metrik



Problémy se systémem měření

Hlavním cílem je systém měření umožňující snadný přístup k metrikám a efektivní metody vyhledávání a vyhodnocování informací.

Cílem měření není pouze každodenní operativní dohled a kontrola.

- Systém orientovaný na kontrolu a dohled má tendenci ustrnout a nevyvíjet se. Navíc hrozí, že se nevyužijí možnosti, které systém nabízí pro vrcholové řízení.

Problémy se systémem měření

- Systém měření nemá vyvolávat odpor, jinak je neefektivní
- Odpor může být způsoben nevhodnými opatřeními managementu.
  - Zviditelnění dobrých výsledků může vést management k rozhodnutí nadměrně redukovat zdroje pro daný úkol.
  - Zviditelnění nevýkonnosti může vést k posílení zdrojů, později však i k postihům. Pozdní posílení může být kontraproduktivní
- Je důležité, aby většina cítila, že jsou metriky užitečné a poctivé zvýhodňují a zvyšují šance na úspěch.

Problémy se systémem měření

- Využívání metrik může být ztíženo krátkozrakostí a profesionálními předsudky (nekritické přeceňování účetnictví a finanční operativy atd.).
- Informace a rozhodnutí mohou složitým způsobem záviset na několika metrikách. Snaha o zjednodušení může vést k nesprávným závěrům. Management by měl své závěry konzultovat s členy týmu.
  - Chybovost modulu může být náhoda
- Efektivnost využití systému sběru a vyhodnocování metrik závisí na tom, do jaké míry bude všemi akceptován a kvalifikovaně užíván. Při tom lze využívat matematickou statistiku

Vlastnosti systému měření

- Jasně definované cíle měření
  - U všech sbíraných dat musí být jasné, že jsou potenciálně užitečné pro všechny, byť se hned nevyužívají,
  - Jinak je sběr metrik chápán jako šikana
- Otevřenost, modifikovatelnost
  - Znalosti o metrikách (state of the art) se mění, mění se i potřeby managementu a standardy

Vlastnosti systému měření

- Vychází z potřeb managementu
  - Integrální součást řízení
- Měření odděleno od vyhodnocování
- Lidé by měli cítit systém měření jako podporu jejich práce a ne jako hrozbu. Neměli by být dominováni systémem

Kvalita dat, opakování

- V managementu se musí používat data, která nejsou zcela spolehlivá a relevantní
- Podpora managementu se stává hlavním úkolem informatiky. Operativa je už do značné míry vyřešeným úkolem (neplatí pro zábavu)

Kvalita dat

Kategorie	**Dimenze**
Vnitřní, intrinsická (Intrinsic)	Přesnost (Accuracy) Objektivnost (Objectivity) Důvěryhodnost (Believability) Reputace (Reputation)
Dostupnost (Accessibility)	Dostupnost (Accessibility, též Availability) Bezpečnost přístupu (Access security)
Kontextuální (Contextual)	Relevantnost (Relevancy) Přínos (Value added) Včasnost (Timeliness) Úplnost (Completeness) Rozsah (Amount of data)
Reprezentační (Representational)	Interoperabilita (Interoperability) Srozumitelnost (Easy of Understanding) Výstižná a stručná reprezentace (Concise representation) Konsistentní reprezentace (Consistent representation)

Problémy s kvalitou dat pro řízeni

- Relevantnost a včasnost závisí na frekvenci zjišťování nebo na tom, jak je časově náročné data vytvořit (např. data rozvrhu)
- Kvalita dat může implikovat vytvoření datového úložiště v SOA, aby management mohl ovlivňovat chod systému

Problémy s kvalitou dat pro řízeni

- Kvalita dat může implikovat filosofii řešení
  - Kritická cesta a kritický řetězec
- Kvalitu je nutno měřit či odhadovat
- Kvalitu dat můžeme zlepšovat
  - Okrajová data, odstranění
  - Chybějící data pro parametry, pro regresi, doplnit
  - Opakovaná data, odstranit
  - Rozsah dat, měl by být dostatečný

Přesnost SW metrik je malá

- Hodnoty metrik závisí
  - Na typu projektu, projekty se málo opakují (výjimka: SAP atp.)
    - Velikost projektu a ostrost termínů
    - Typu úlohy RT*dávka bez přímých následků
  - Na stavu oboru
    - Výkon HW, sítě, vývojová prostředí,
    - paradigmata

Přesnost SW metrik je malá

- Hodnoty metrik závisí
  - Dosažitelné technologii
  - Kvalitě programátorů (produktivita 1:20)
    - Kvalitnější vývojáři píší lepší programy (rychlost 1:10 a to s méně chybami)
    - Programátoři jsou schopni silně ovlivnit určitou SW metriku, je-li jim dovoleno nebrat ohled na jiné metriky (rychlost a úkor délky)
  - Do jaké míry se řeší podobné problémy (viz minulý semestr)

Přesnost SW metrik je malá (velký rozptyl)

- Výhrady k přesnosti metrik neplatí pro metriky sledující vlastnosti systému za provozu
  - Frekvence chyb/střední doba mezi poruchami
  - Počet stížností
  - Frekvence nalézání problémů a detekce defektů

Pracnost závisí zejména na

– druh softwaru: míra interakce od dávkových až po tvrdé systémy reálného času, závažnost

- důsledků selhání, od nevýznamných škod, přes ekonomické ztráty až k ohrožení životů. V neposlední řadě

- velikosti systému;

– ostré až obtížně splnitelné termíny realizace;

– použití moderních projekčních technik a technik vývoje;

– kvalita zúčastněných;

– omezení hardwaru a softwaru. Pokud systém využívá zdroje jako je rychlost a paměť více než na dvě třetiny, vzrůstá značně pracnost řešení.

Interní a externí metriky (ISO 9126, ISO 250XX)

- Též implicitní a explicitní metriky (in process, after process)
- Interní – známo jen týmu během vývoje na základě sledování vývojových procesů: např. spotřeba práce, a také doba řešení
- Externí: dají se zjistit na hotovém produktu, např. délka
- Některé metriky je možné chápat obojím způsobem (počet selhání při testování, externí – zjištěno uživateli)

Datové typy SW metrik

- Příslušnost k třídě (id. - číslo tramvaje)
  - Trendy počtů objektů s daným id, operace rovnost =)
- Fuzzy (id hodnocení, dobrý, lepší, nejlepší, známky ve škole)
  - Operace = test na rovnost, < test na „velikost“. Obvykle se převádí na číselné hodnocení. Př. COCOMO, známky ve škole (tam se používají přímo fuzzy hodnoty ve formě čísel)

Datové typy SW metrik

- Interval
  - Čas, teplota
  - Je možná lineární transformace, =, <
  - Využitelný je rozdíl jako číselná metrika
- Číselná metrika
  - Délka programu, doba řešení
  - Jsou smysluplné všechny operace pro reálná čísla

ISO 9126 (4 části)

- Stovky metrik, rozumně použitelné většinou jen u velkých firem
- Není jasné, k čemu se to všechno použije
- Jádro (principy a některé metriky) použitelné
- Modernizace norme - sada norem 250xx
  - Základní principy a základní metriky stejné

Externí metriky

- *Del* – délka produktu v řádcích. U programů se nepočítají komentáře. *Del* programů se někdy udává v lexikálních atomech. Do metriky *Del* se někdy nezahrnují deklarace proměnných a záhlaví podprogramů.
- *Srnd* – rozsah slovníku operandů. Tato metrika se týká programů. Operand je bud' konstanta (např. celé číslo 10, nebo řetězec znaků „ xyz“, v terminologii programování literál), nebo proměnná, např. *x*. *Srnd* je pak počet logicky odlišných operandů vyskytujících se v programu.

Externí metriky

- *Nrnd* – počet výskytů operandů v programech
- *Noper* – Počet výskytů delimiterů a znaků operací a podprogramů v programech.
- *Soper* – rozsah slovníku operací, podprogramů a delimiterů. Tato metrika udává, kolik program obsahuje významem různých znaků operací (*, C, atd.), jmen podprogramů (sin, tan, put), delimiterů (: **if begin real** atd.).

Externí metriky

Users – Maximální počet uživatelů, pro které je systém plánován.

McCabe – počet podmíněných příkazů (if), příkazů cyklu (for, while, . . .) a přepínačů (case) v programu. Metrika *McCabe* je dobrým indikátorem složitosti programů. Jejím nedostatkem je, že není citlivá na hloubku a způsob vloženosti podmíněných příkazů (tvar rozhodovacího stromu (případně lesa).

Externí metriky

In, Out, Qer, File, Filee – složitost příkazů vstupu, výstupu, dotazů na terminál a operací se soubory interními a se soubory společnými s jinými aplikacemi. U databází složitost SQL dotazů. Tyto metriky se používají v odhadech pracnosti a doby řešení v metodě funkčních bodů.

FanIn, FanOut – míry indikující složitost rozhraní tříd, modulů a aplikací. *FanIn* udává počet logicky různých typů dat vstupujících do dané entity (např. modulu). *FanOut* je obdoba *FanIn* pro vystupující datové toky.

Často se používá metrika $Fan1 = \sum_{modul\ i} (FanIn_i * FanOut_i)^{2.}$

Problém metrik pro SOA

- Obtíže s měřením složitosti komunikace,
- Jak měřit složitost hrubozrnných zpráv
-

Interní metriky

- *Prac* – spotřeba práce, obvykle v člověkoměsících
- *Doba* – doba provádění
- *Prod* – jednotky délky za člověkoměsíc (řádky za měsíc)
- *Fail* – počet selhání za týden (den)

Interní metriky

- *Prod* – produktivita, počet jednotek délky vytvořených za člověkoměsíc.
- *team(t)* – velikost týmu (počet osob) v čase *t*, měřeno od začátku prací. Tato metrika umožňuje postupné zpřesňování odhadů pracnosti a doby řešení během vývoje projektu.
- *Team* – průměrná velikost týmu.

Interní metriky

Fail(t,p) – počet selhání systému /části *p* detekovaných při testování či provozu v čase *t*. Při provozu hlásí selhání zákazníci. Obvykle se udává po dnech nebo týdnech. Tato metrika je důležitou mírou kvality. Při inspekcích je hodnota metriky *Fail* dána počtem chyb zjištěných při inspekci.

Interní metriky

Defect(t,p) – počet míst v programech, která bylo nutno opravit pro odstranění selhání nebo pro nápravu selhání v čase *t* (den, týden) v části *p*, normalizovaný pro 1 000 řádků (1000 _ počet defektů_délka).

Defect1(e1,e2,t,p) – počet defektů v části *p* vzniklých v etapě řešení *e*1 a zjištěných v etapě *e*2 v době *t*. Tato metrika a metriky z ní odvozené jsou účinnou mírou efektivnosti inspekcí a testů. Důležitá externí metrika pro vyhodnocování kvality produktu

Interní metriky

Satisf(t) – průměrná míra spokojenosti zákazníků včase *t*. Spokojenost se udává ve stupnici 1 až 5 (nejlepší). Lze vyhodnocovat trendy. Existují metody odhadu vývoje úspěšnosti produktu na trhu z aktuálních hodnot metrik *Satisf* (Babich, 1992).

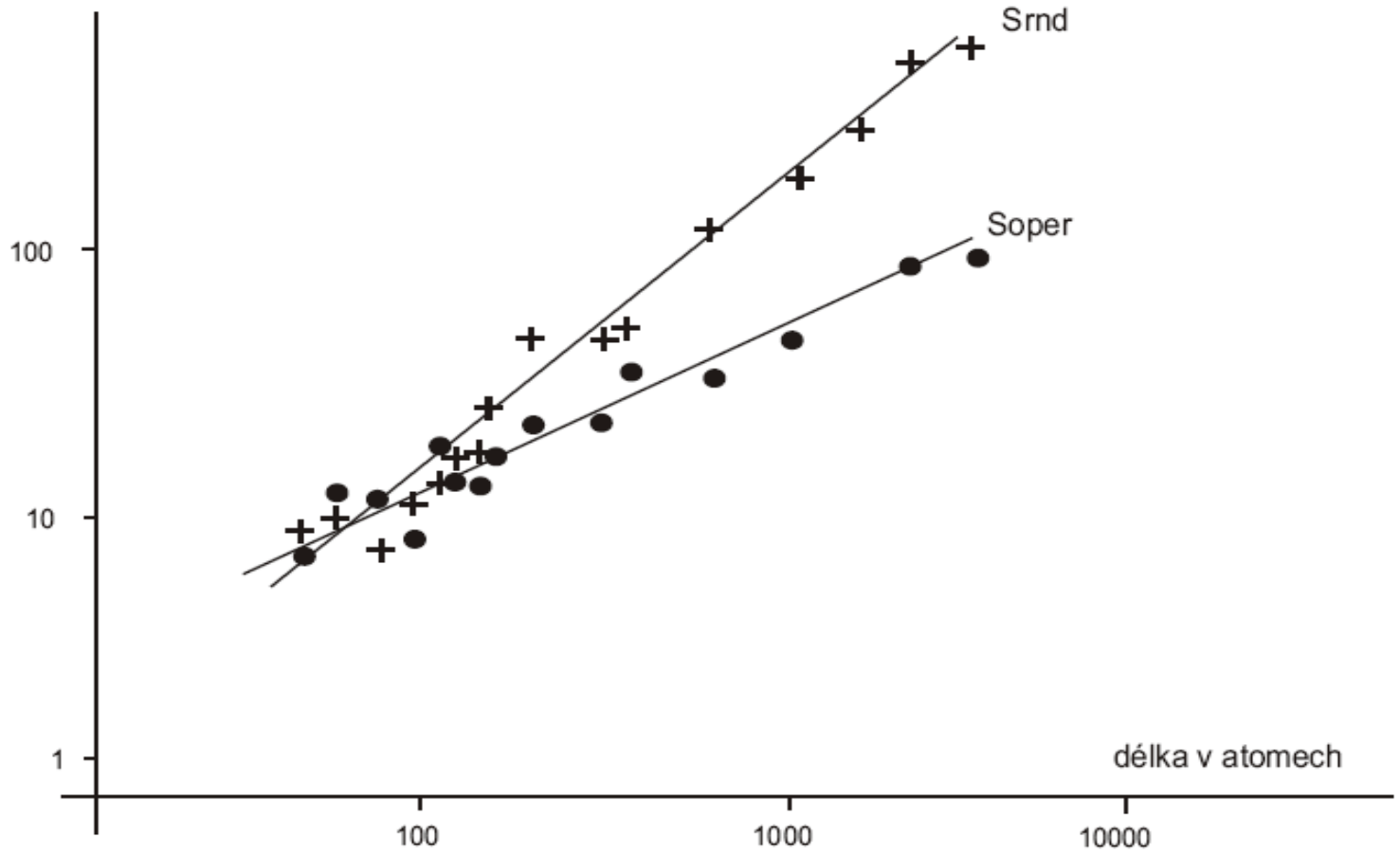
DobaOpr – průměrná doba opravy selhání.

Interní metriky

Zmeny(f,t) – počet změněných míst souboru *f* včase *t* (týdnu/dni). Tato metrika se snadno zjišťuje a její trendy mohou v průběhu prací poskytnout cenné informace.

MTBF(t) – (Mean Time Between Failures): střední doba mezi poruchami (v určitém období, např. týdnu, *t*).

Vztah mezi rozsahy slovníků operandů a operací



Výpočet charakteristik programu

	Del	*Noper*	*Nrnd*	*Soper*	*Srnd*
`begin`	1	1		1	
`var x,y:real;`	7	5	2	5	2
`x:= y*2.0+sin(x)+2.0`	12	7	5	5	1
`end.`	1	1		1	
Celkem	21	14	7	12	3

Halsteadův vztah pro malé programy

Odvozeno z analýzy počtu rozhodnutí při psaní

$$Prac \cong C\ Del\ Nrnd\ (Soper/Srnd)\ log(Soper+Srnd)$$

V dobře napsaných programech jsou *Srnd* a *Nrnd* úměrné délce. Pro velké *Soper a Srnd* lze logaritmus nahradit konstantou. Pak ale bude

$$Prac \cong C'\ Del\ Soper$$

Halsteadův vztah pro malé programy

Poněvadž je pozorováno, že

$$Soper \cong C\text{“ } Del^{b}$$

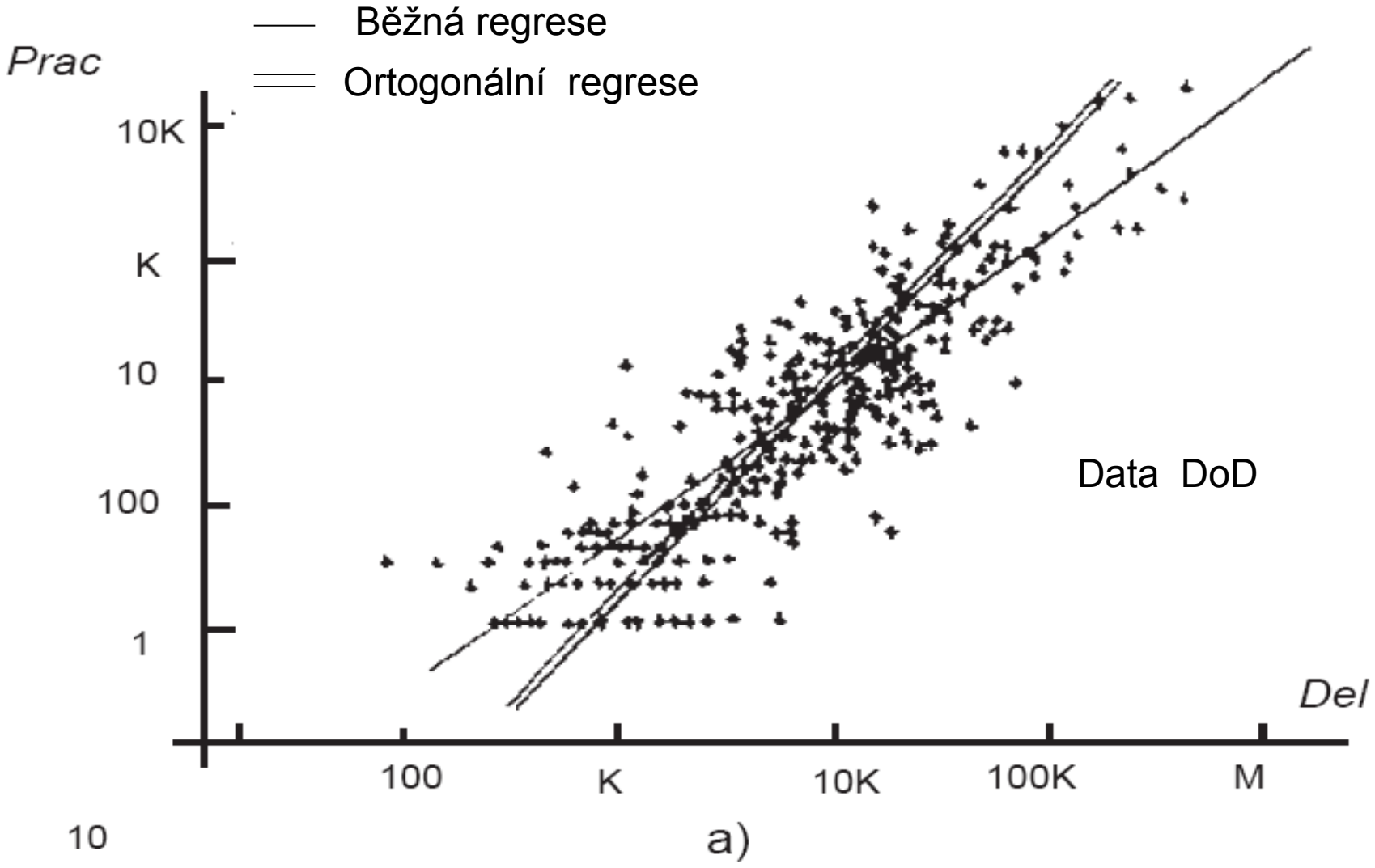
dostaneme

$$Prac = c\ Del^{1+b}\ \ b \cong\ 0{,}25$$

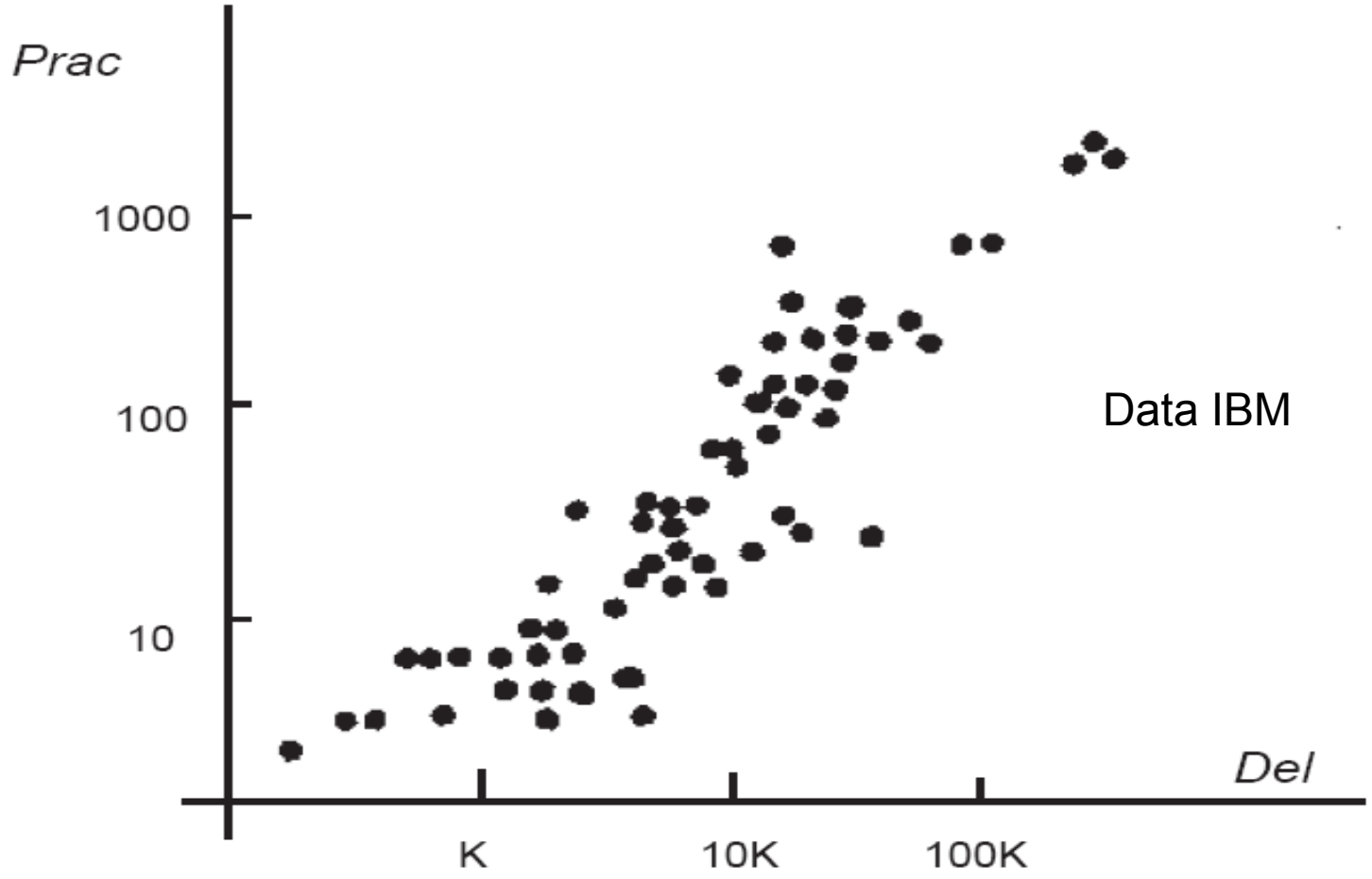
Pro velké programy je hodnota b příliš vysoká. Je to důsledek toho že vztah modifikujtakové technologie, jako dekompozice a znovupoužití částí.

Pokud se dekomponuje do malých komponent a systémy se liší jen počtem komponent a transport zpráv se nemusí pracně implementovat, bude b velmi malé.

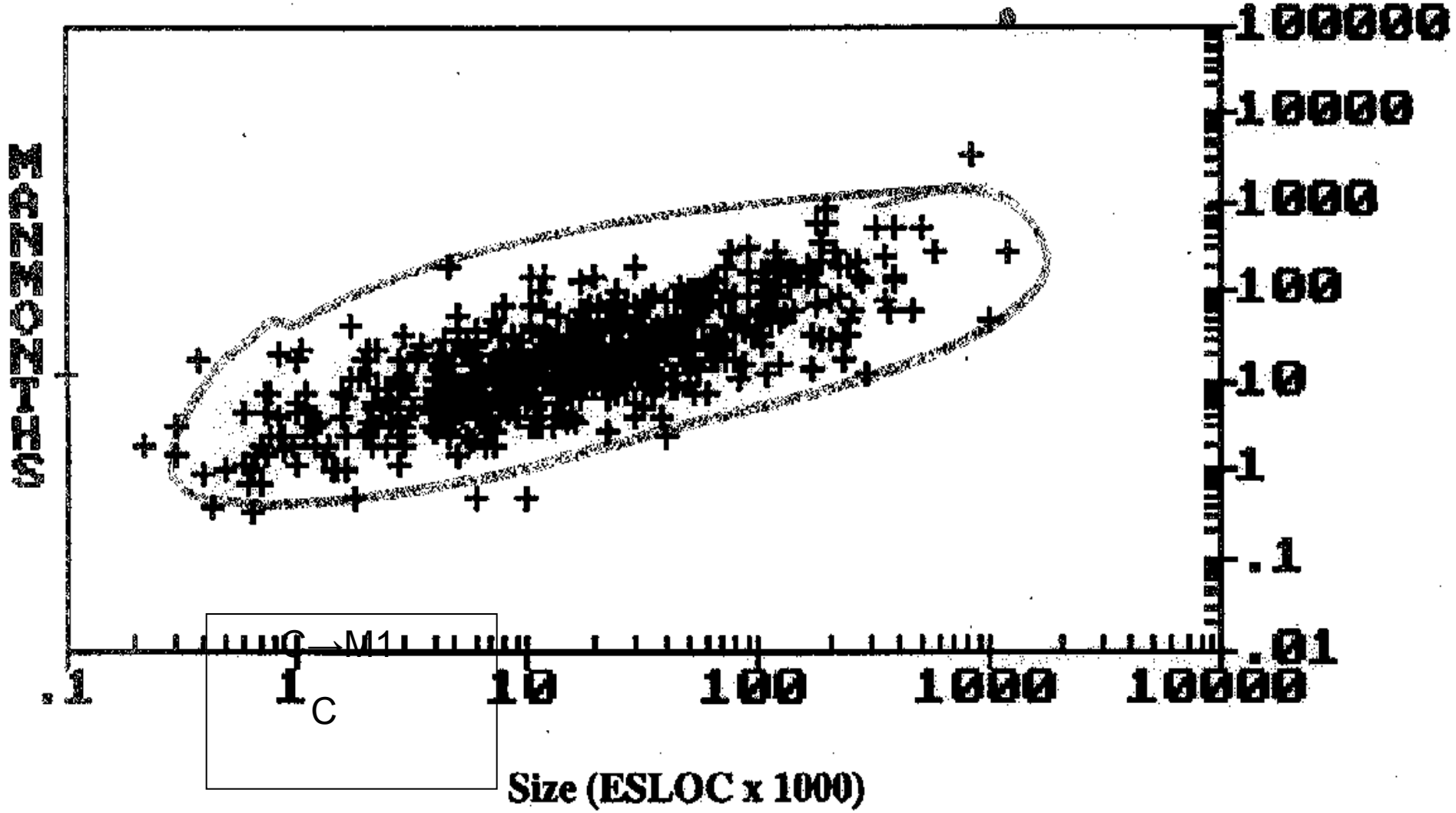
Závislost pracnosti na délce programu



Závislost pracnosti na délce programu

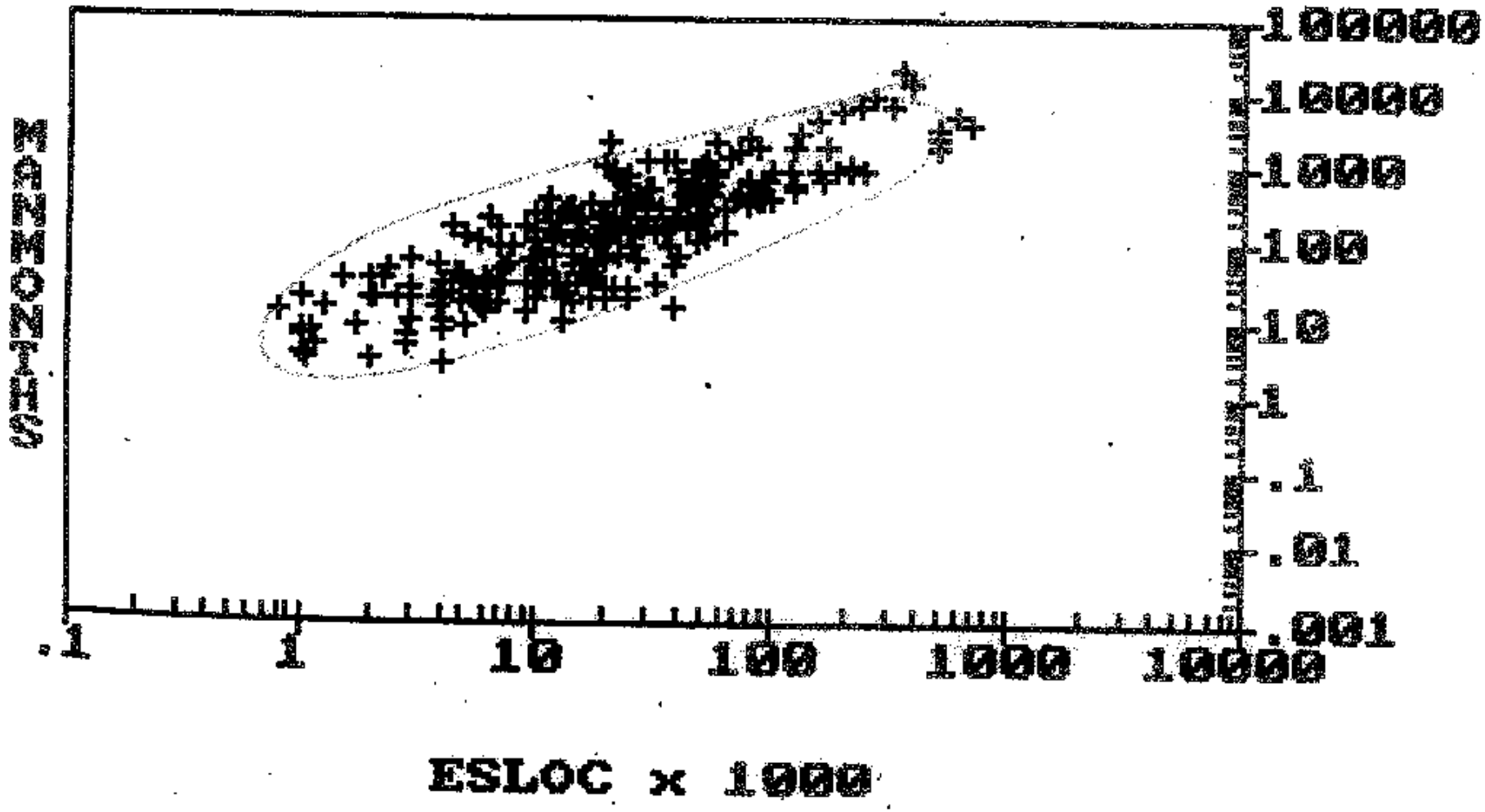


Business Systems

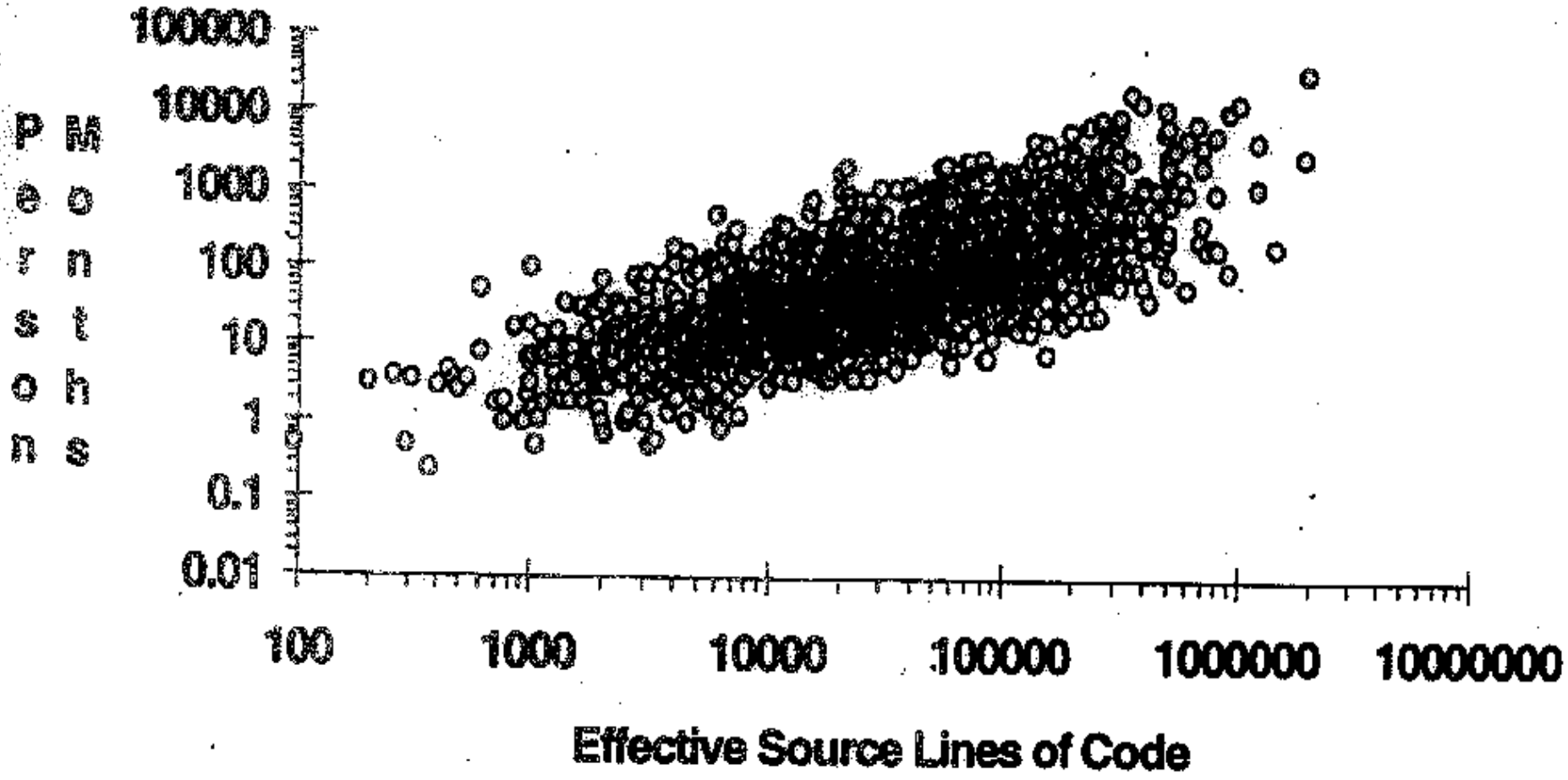


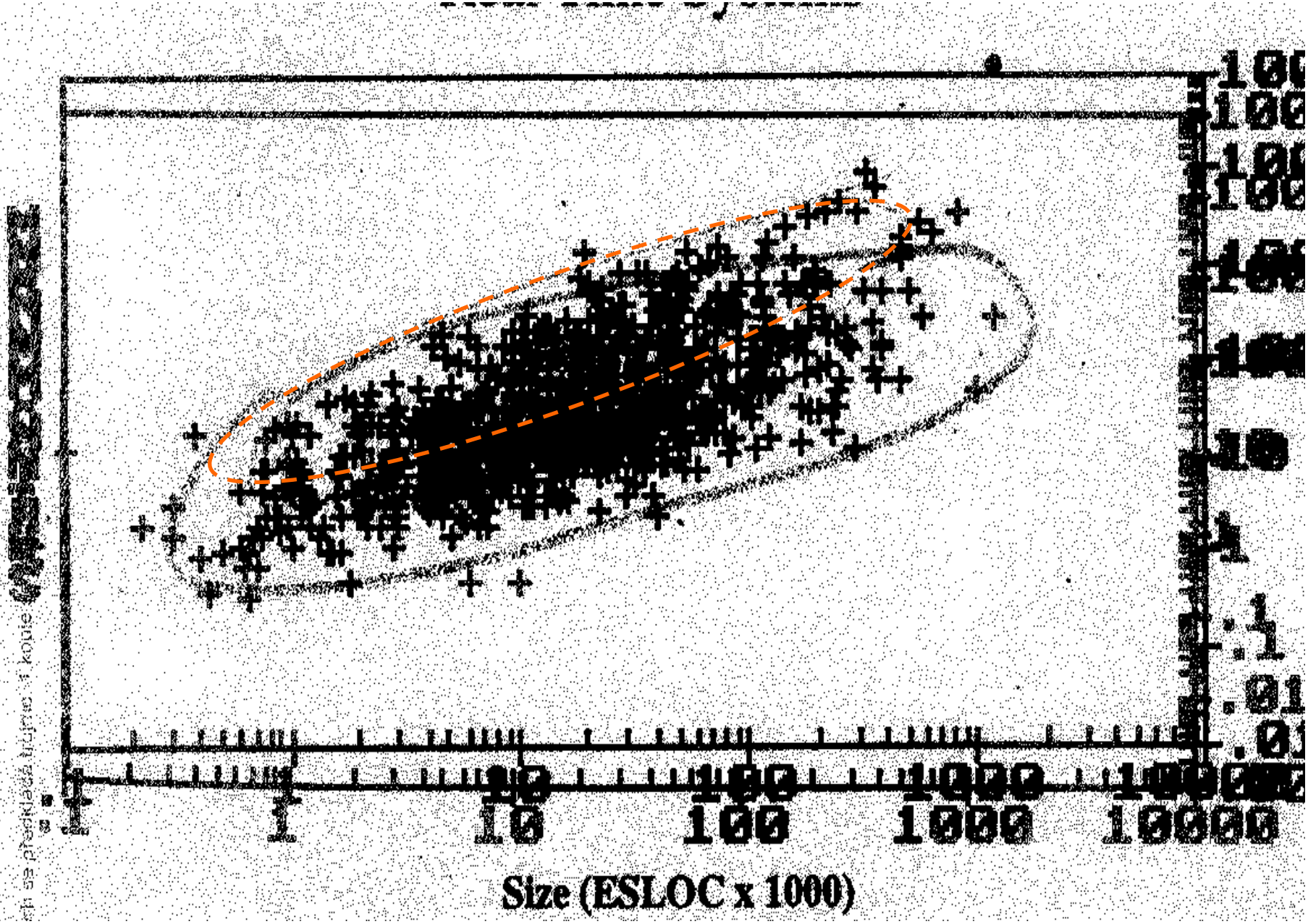
Effort vs. Size

Real Time Systems



Effort Behavior




1
10
100
1000
10000
Size (ESLOC x 1000)

Pozor na statistiku

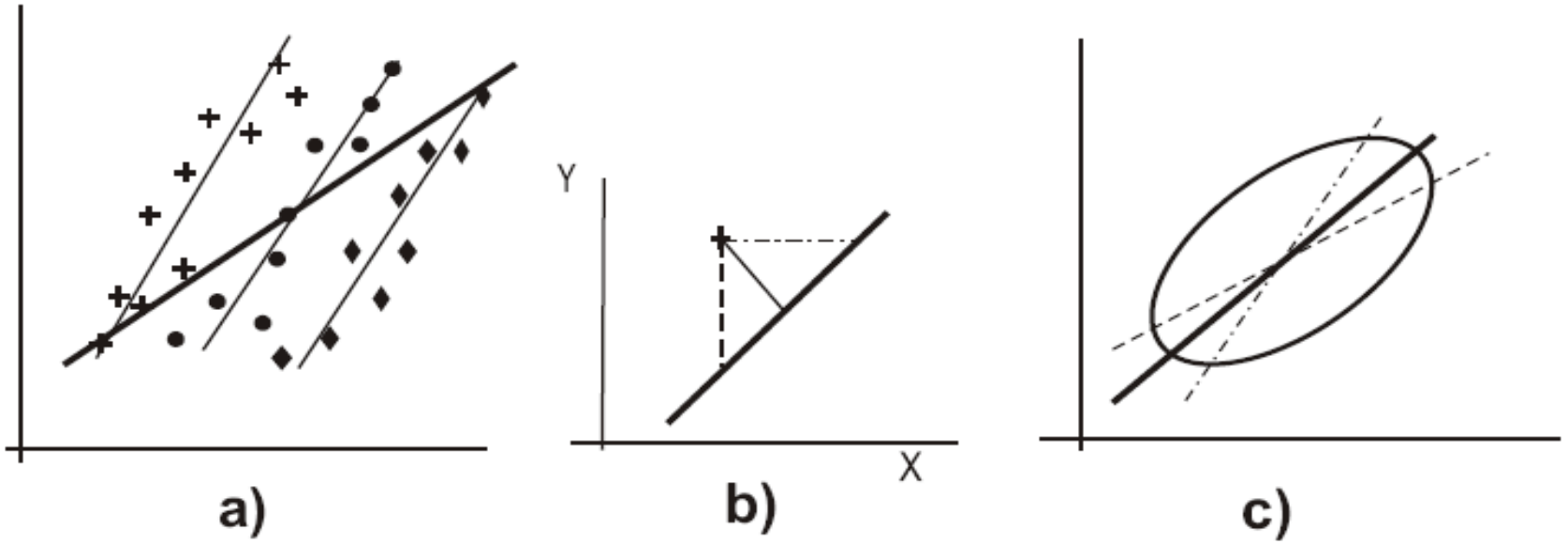
- Analýzou dat klasickou regresí dostaneme

$$Prac = c_{10} Del^{d}$$

Kde $d < 1,$ což není zřejmě reálné, neboť jedna instrukce ve velkém systému dá podle zkušeností více práce (např. díky nákladům na chod týmu, než jedna instrukce v malém systému.

Je nutné použít ortogonální regresi.

Problémy s regresí



Vysvětlení rozporných výsledků analýzy regrese.

a) Regresní přímky pro jednotlivé týmy (tence) mají větší sklon než regresní přímka pro data všech týmů (silná čára).

b) ——— odchylka uvažovaná při ortogonální regresi,
------ odchylka uvažovaná při standardní regresi,
–·–·– odchylka uvažovaná, je-li Y nezávisle proměnná.

c) Regresní přímky pro data pokrývající elipsu. Čím je elipsa širší, tím je menší sklon regresní přímky pro standardní regresi a X jako nezávislou proměnnou.

Pro obchodní systémy ani to nevysvětluje nízký koeficient regrese

- Možnost: Velké systémy více používají různé nástroje, jako jsou generátory kódu, CASE systémy, atd. Generovaný kód je „řinčí“

Prvá zákonitost

Pro délku a pracnost platí vztah

$$log(Prac) = (1+a)log(Del) + c'$$

takže

$$Prac = c\ Del^{1+a}$$

Ortogonální regrese dává

$$a \cong 1/8$$

Prvá zákonitost

Pro délku a pracnost platí vztah

$$log(Prac) = (1+a)log(Del) + c'$$

takže

$$Prac = c\ Del^{1+a}$$

Ortogonální regrese dává

$$a \cong 1/8$$

Avšak c i a závisí na typu projektu

Použití a důsledky

- Vztah $Prac = c\ Del^{1+a}$ *se využívá v odhadu COCOMO*
- Pokud je pracnost budování middleware zanedbatelná klesne pracnost monolitního systému po dekompozici na n zhruba stejných dílů z $Prac_1 = c\ Del^{1+a}$ na

$$Prac_n = c\ n(Del/n)^{1+a} = n^{-a}\ Prac_1$$

Empirické výsledky

	Zdroj	Nezávisle proměnná X	Závisle proměnná Y	koeficient regrese	koeficient korelace
1.	Norden,1978	log(*Del*)	log(*Prac*)	1.125	0.85
2.	Walston, Felix, 1977	log(*Del*)	log(*Prac*)	1.125	0.80
3.	Putnamn, 1978	$\log(P/D^2)$	log(*Prod*)	-0.66	-0.98
4.	Walston, Felix, 1977	log(*Del*)	log(*Doba*)	0.44	0.62
5.	Walston, Felix, 1977	log(*Prac*)	log(*Doba*)	0.40	0.77
6.	Walston, Felix, 1977	log(*Prac*)	log(*Team*)	0.62	0.82
7.	Christensen, Fitsos, 1981	log(*Srnd*)	log(*Soper*)	0.33	0.78
8.	Fitsos, 1980	log(*Srnd*)	log(*Soper*)	0.48	0.69
9.	Wolberg, 1981	log(*Srnd*)	log(*Soper*)	0.45	0.88
10.	Wolberg, 1981	log(*Srnd*)	log(*Soper*)	0.29	0.86
11.	Halstead, 1977	*Del*	*Nrnd*	1.00	0.99
12.	Halstead, 1977	log(*Del*)	log(*Srnd*)	0.97	0.92
14.	Halstead, 1977	log(*Del*)	log(*Srnd*)	0.75	0.70
15.	Halstead, 1977	log(*Del*)	log(*Soper*)	0.48	0.50
16.	Halstead, 1977	log(*Del*)	log(*Soper*)	0.42	0.30

Tab. 15.4: Tabulka výsledků ortogonální regresní analýzy pro různé soubory dat. Data zřádků 9 až 1 se týkají malých podprogramů.

Putnamova rovnice

Podle třetí řádky tabulky

$$\log(Prod) \mathrel{\widehat{=}} c_{25} - 0.67 \cdot \log(Prac/Doba^2)$$

čili

$$Prod \mathrel{\widehat{=}} c_{26} Prac^{-2/3} Doba^{4/3}$$

Z definice Del = Prac Doba, takže

$$Del \mathrel{\widehat{=}} c_{26} Prac^{1/3} Doba^{4/3}$$

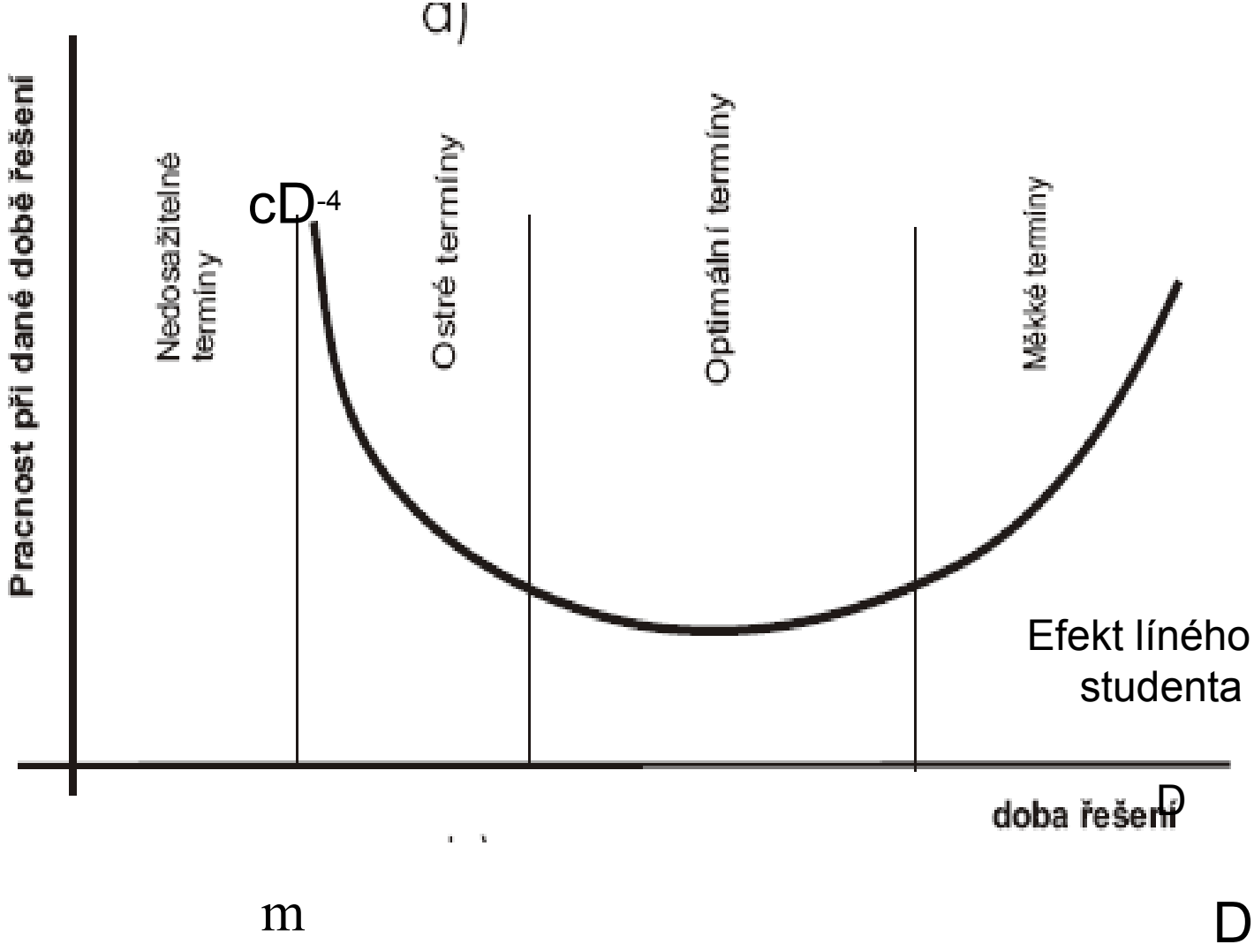
To je Putnamova rovnice

Vliv napjatých termínů

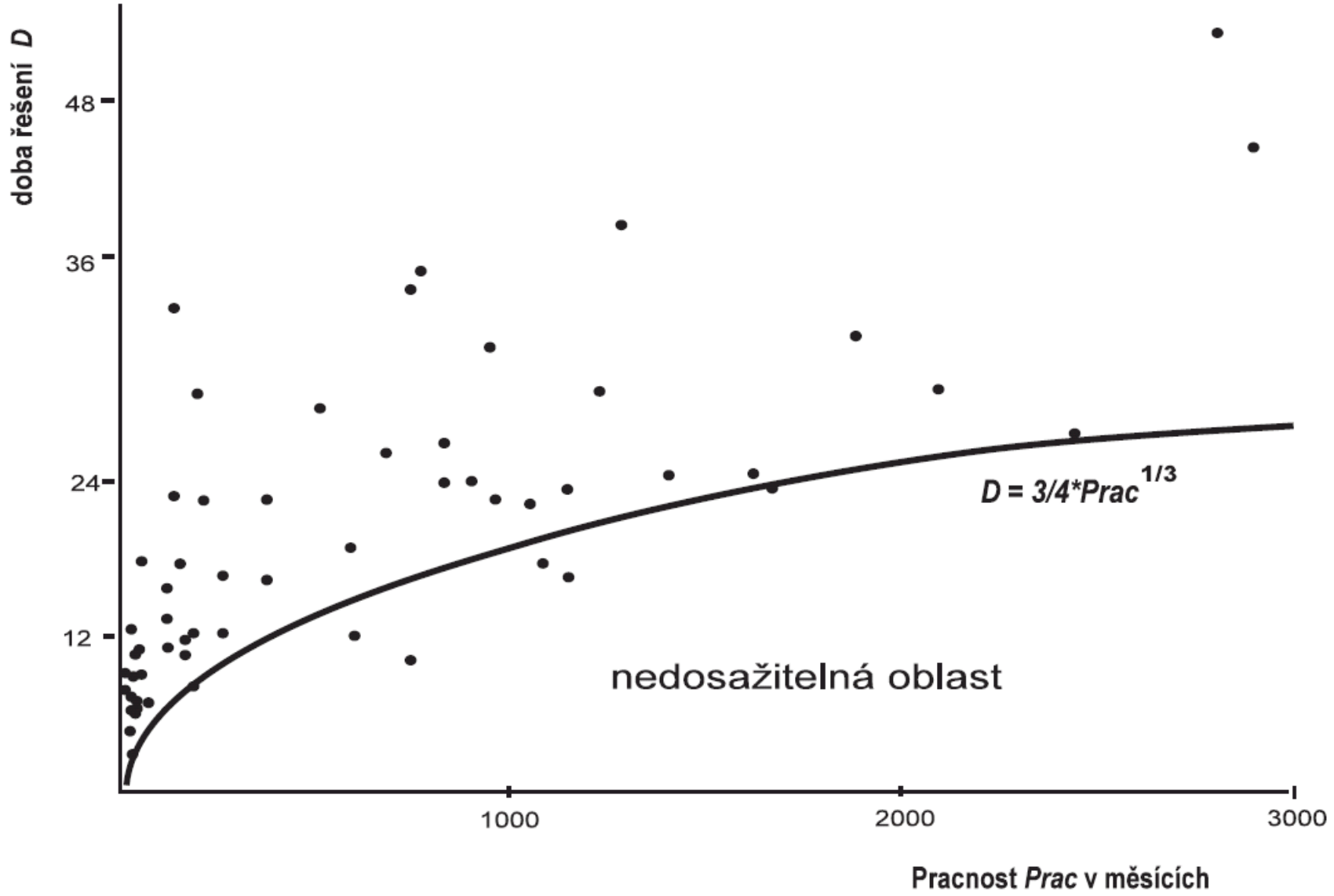
- Dobu řešení nelze v praxi libovolně zkracovat,
- Pro každý projekt existuje tedy mez m, pod níž se nelze prakticky dostat. Interval $<0,m>$ se nazývá nedosažitelná oblast pro daný projekt. m je funkcí *Prac*
- Pro více projektů je nedosažitelná oblast oblast roviny *(Prac, Doba),* kde

$$Doba < \tfrac{3}{4}\, Prac^{1/3}$$

Pracnost u nedosažitelné oblasti



Některé starší výsledky pro SW projekty



Chování pracnosti u nedosažitelné oblasti

Uvažujme dvě realizace A,B stejného projektu s atributy

$$Del_A, Doba_A, Prac_A$$

$$Del_B, Doba_B, Prac_B.$$

Vydělením Putnamových rovnic pro obě realizace dostaneme

$$Del_A/Del_B = (Prac_A/Prac_B)^{1/3} \cdot (Doba_A/Doba_B)^{4/3}$$

Chování pracnosti u nedosažitelné oblasti

Nechť $Doba_A > Doba_B$. Poněvadž programy psané ve spěchu bývají delší je možné předpokládat $Del_A \leq Del_B$ tj. $Del_A / Del_B \leq 1$

Po úpravách dostaneme z poslední rovnice

$$1 \geq Del_A / Del_B = (Prac_A / Prac_B)^{1/3} \ (Doba_A / Doba_B)^{4/3}$$

Z toho po úpravách, považujeme-li hodnoty s indexem A za konstantní dostaneme obdobu Stefan-Botzmannova zákona

$$Prac_B \geq c_{30} \ Doba_B^{-4}$$

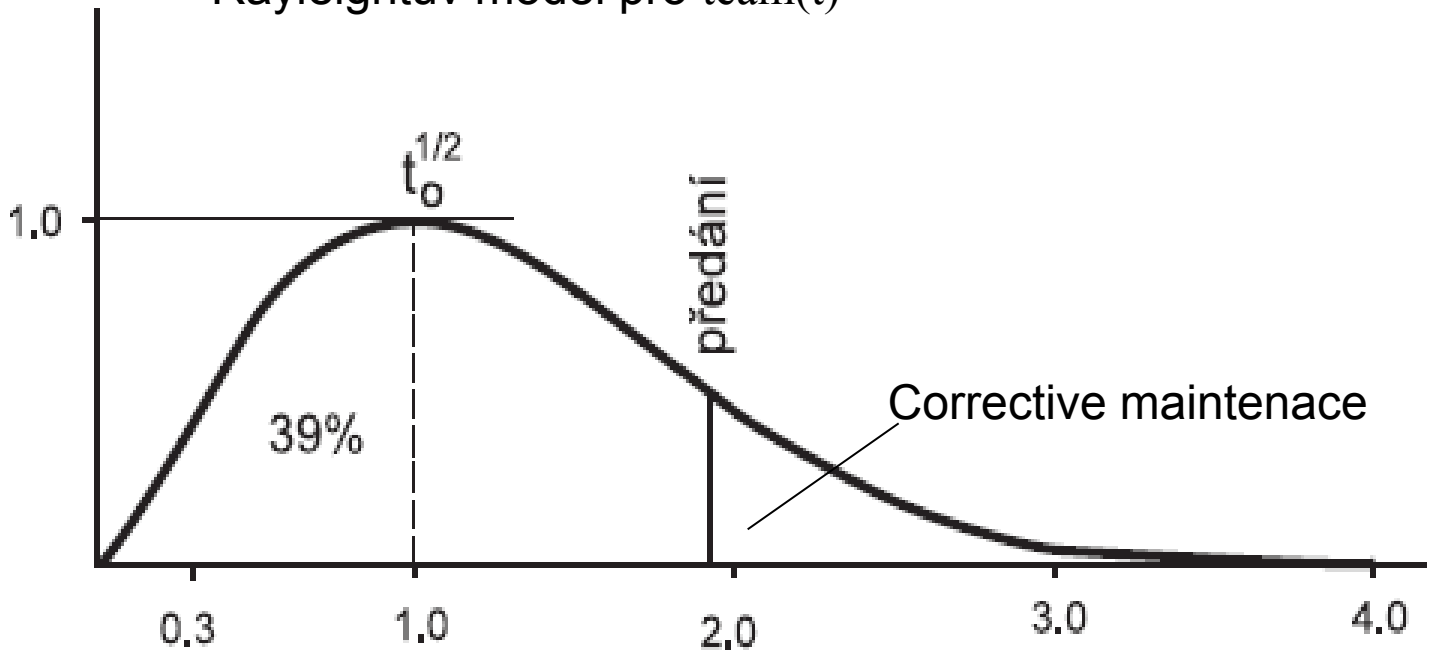
Použití ve function points

Zkrácení termínu o šestinu prodlouží pracnost dvakrát

$$Prac_A \doteq Prac_B \cdot (6/5)^4 \doteq 2 \cdot Prac_B .$$

Ale zkrácení o polovinu zvýší pracnost šestnáctkrát. Metodika Function points nemá opravu na zkracování termínů tak drastickou, zkrácení na polovinu ale nepovažuje za možné.

Rayleightův model pro team(t)



Obr. 15.12: Rayleightova křivka. Část plochy pod křivkou po předání jsou práce, které budou provedeny v rámci údržby (corrective maintenance). To, co se do okamžiku předání nestihne udělat, přechází do údržby, kde se projevuje jako neodstraněné chyby.

$$team(t) \mathrel{\widehat{=}} K \cdot t / t_0 \cdot \exp(-\frac{t^2}{2t_0})$$

Kritika Rayleightova modelu

- Příliš rychle klesá v nekonečnu, potom by ale byl SW bez chyb možný a to se nepozoruje
- Množství práce do maxima je příliš veliké, neodpovídá datům o corrective maintenance (40% pracnosti vývoje)
- Nevysvětluje, proč platí pro SW Stefan-Botzmannův zákon
- Má málo parametrů a změna parametrů nemění příliš tvar
- Proto si půjčíme Planckův zákon

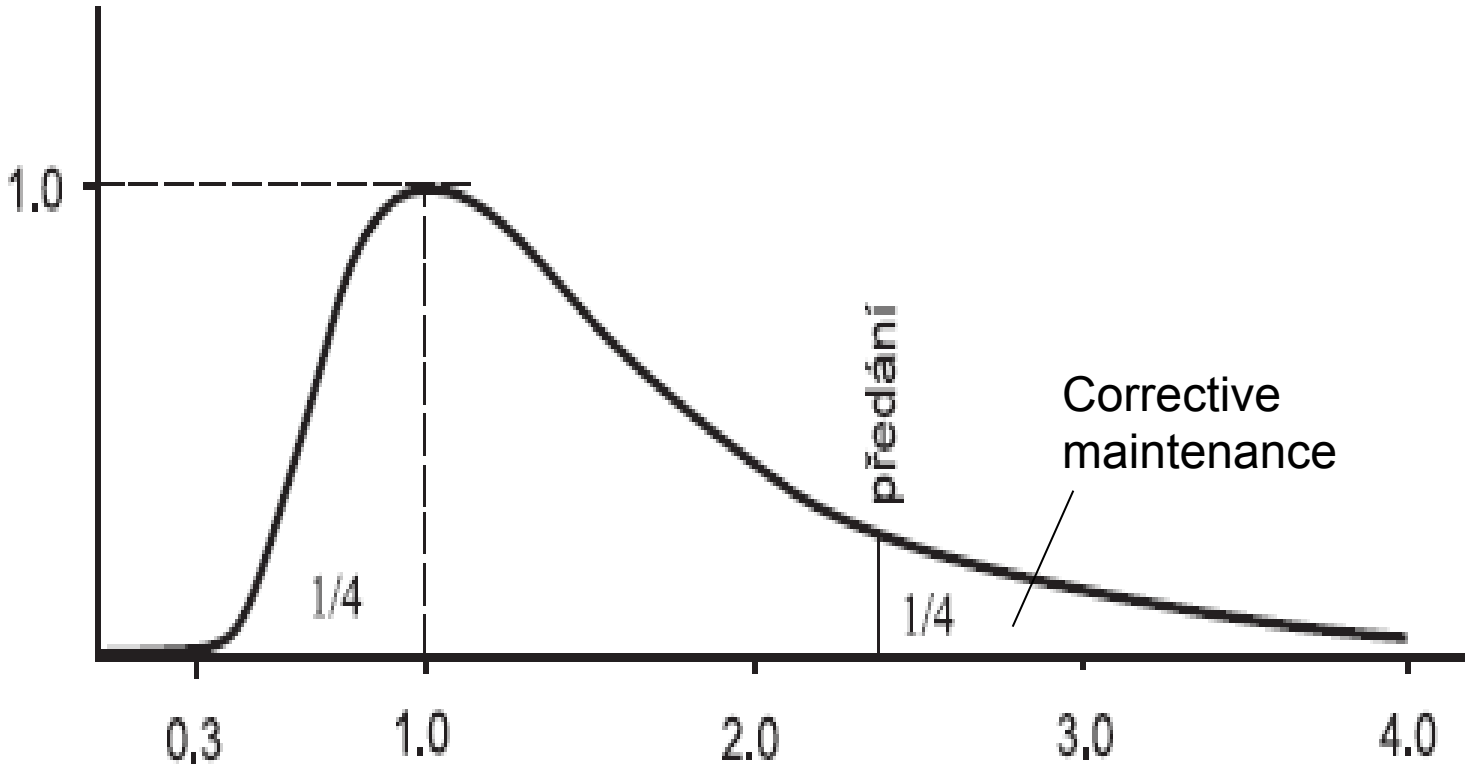
Planckův model

- Standardní tvar zákona záření absolutně černého tělesa

$$\mathrm{Team}(t) = \frac{Ct^{-5}}{\exp(D/t)-1}$$

D určuje tvar křivky a polohu maxima

Planckův model



Obr. 15.13: Normalizovaný tvar Planckovy křivky $Planck(t) = 142.32 \cdot t^{-5}/(\exp(4.9651/t) - 1)$.

Kritika Planckova modelu

- Začíná růst až od 1/3 (Existují názory, že je to OK, pokud do $team(t)$ nezahrnujeme práce na specifikacích).
- Má také málo parametrů

A. Zavedeme lineární transformaci nezávisle proměnné
B. Změníme exponent 5 v polynomiální části

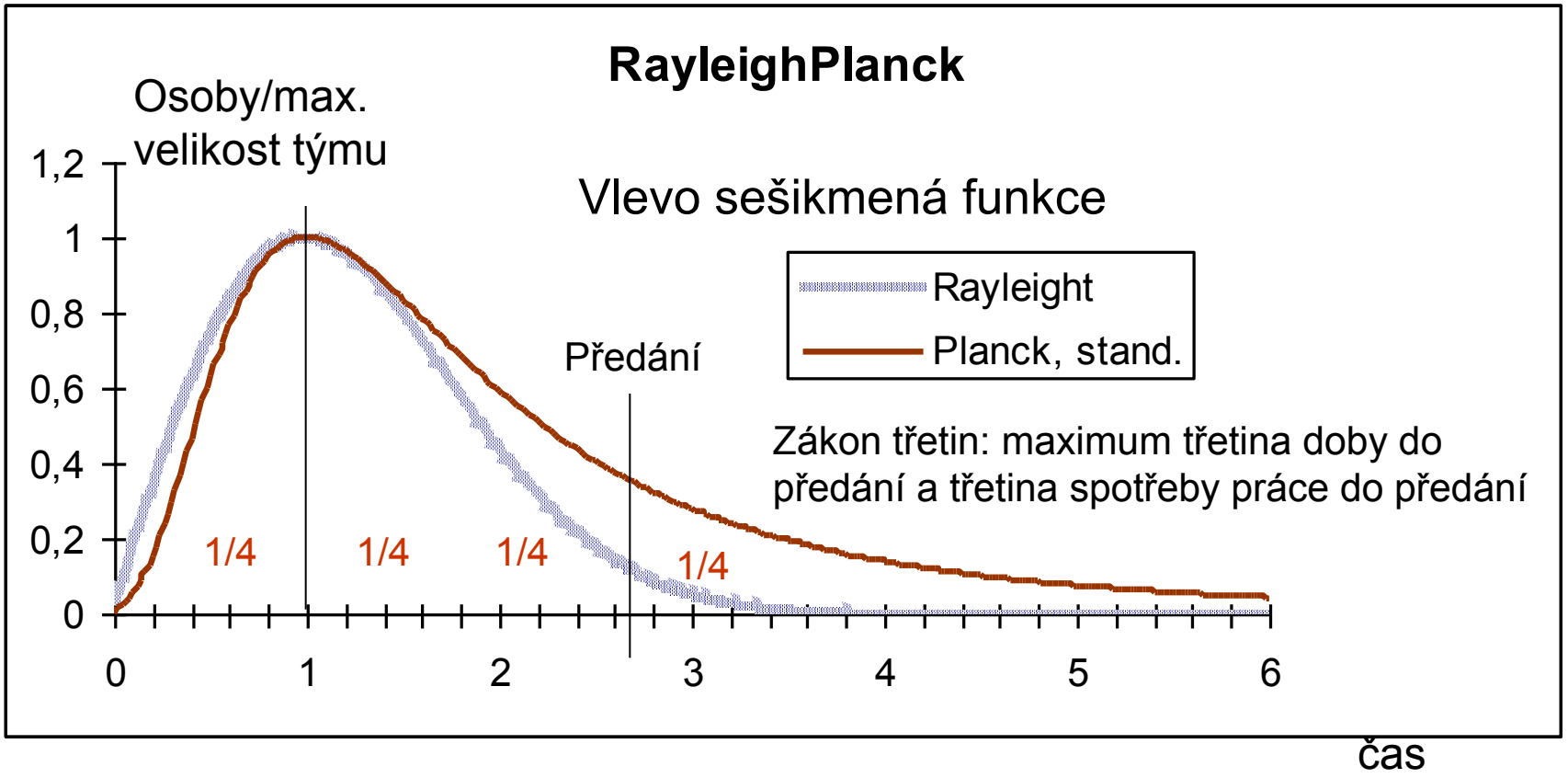
Modifikace Planckova modelu

A. Transformace nezávislé proměnné

$$\text{team(T)} = \frac{c \cdot d^5 (T + k \cdot d)^{-5}}{\exp\left(\frac{D \cdot d}{T + k \cdot d}\right) - 1}$$

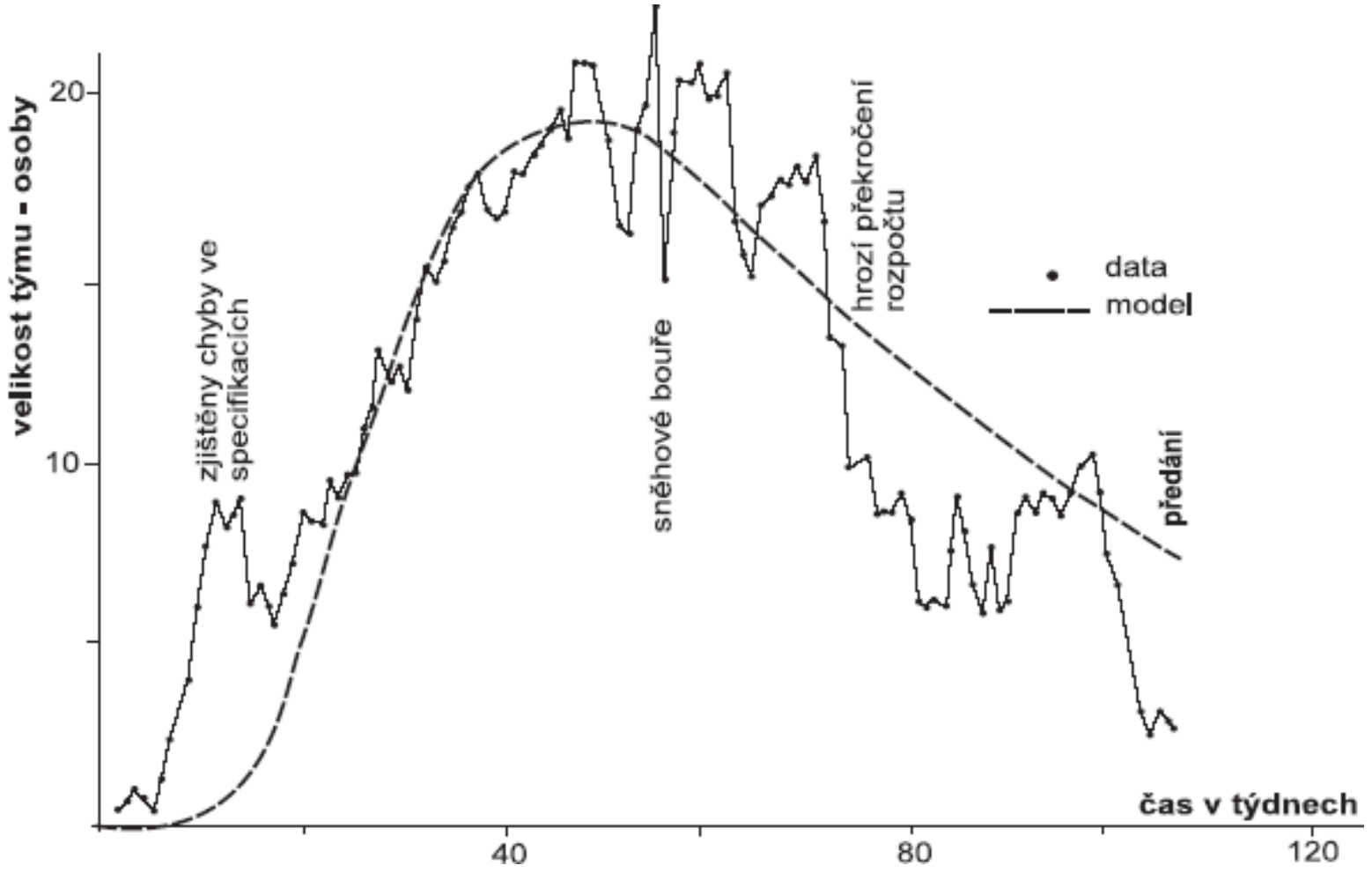
Dává dobré výsledky

Najímaný tým, vrchol a odhad doby řešení

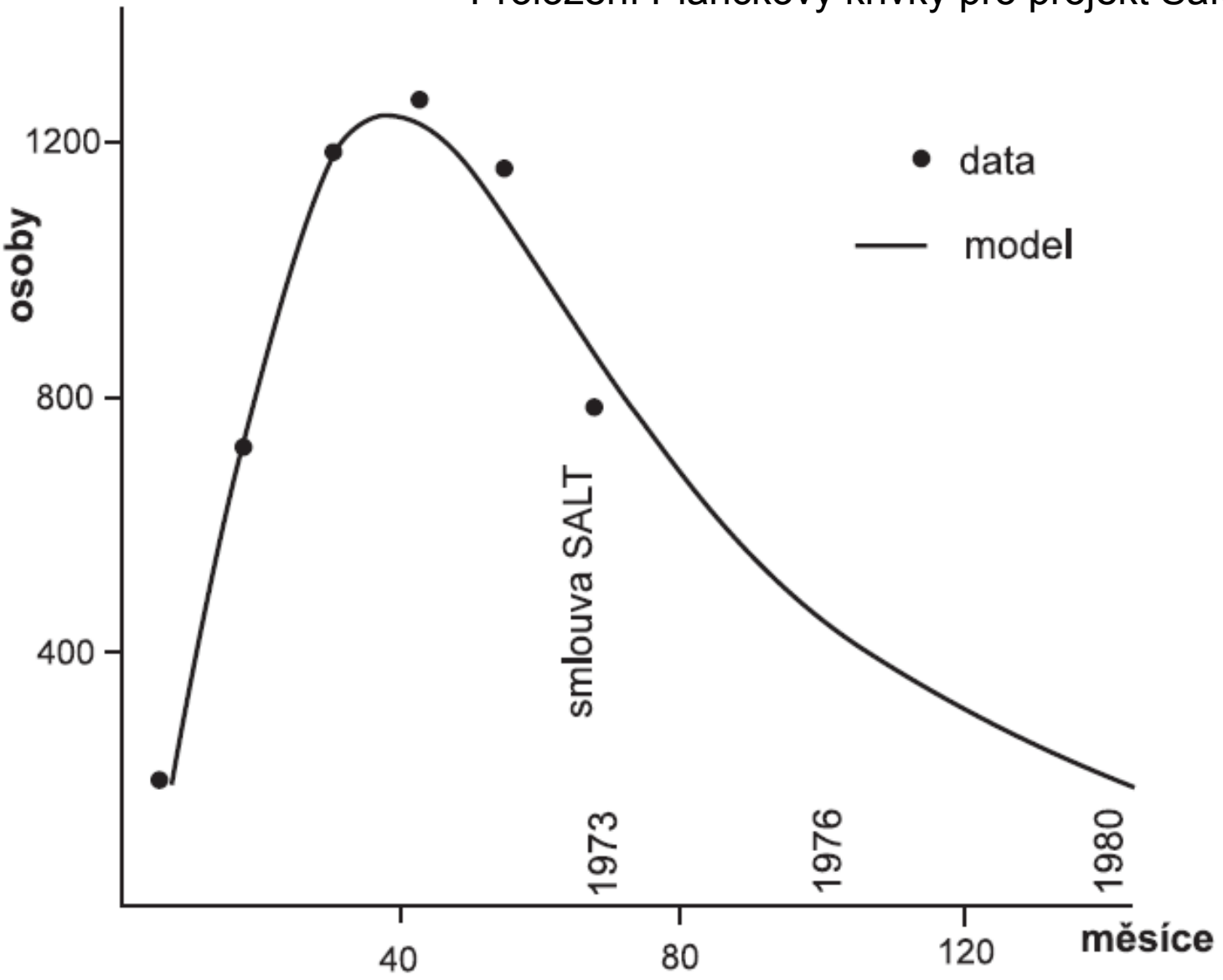


Transformace proměnných tak, aby max bylo v bodě 1 a mělo hodnotu 1 a v nule byla hodnota funkce prakticky nula. U pevného týmu odpovídá intensitě práce

Proložení Planckovy křivky pro SW řízení ponorek



Proložení Planckovy křivky pro projekt Safeguard



Obr. 15.14: Planckův model velikosti týmů pro projekt Safeguard.

Bylo jasné, že projekt nelze v rozumné době dokončit i proto, že nebylo možno včas vytvořit SW. A také by to stálo majlant

Zobecnění Planckova modelu

- Planckův model lze zobecnit

$$\text{team}\ (T) = \frac{C(T+kd)^{-q}d^{q}}{\exp(D/(T+kd))-1}$$

Paramery jsou C, D, k, d q. Stefan-Botzmannův zákon pak dostane tvar

$$\text{Prac}_B \geq c_{30}\ \text{Doba}_B^{-q+1}$$

Zobecnění Planckova modelu

- Zobecněný Planckův model dostaneme výše uvedeným postupem, má-li Putnamům zákon tvar
  - $Del \cong cPrac^{1/p} Doba^{q/p}$

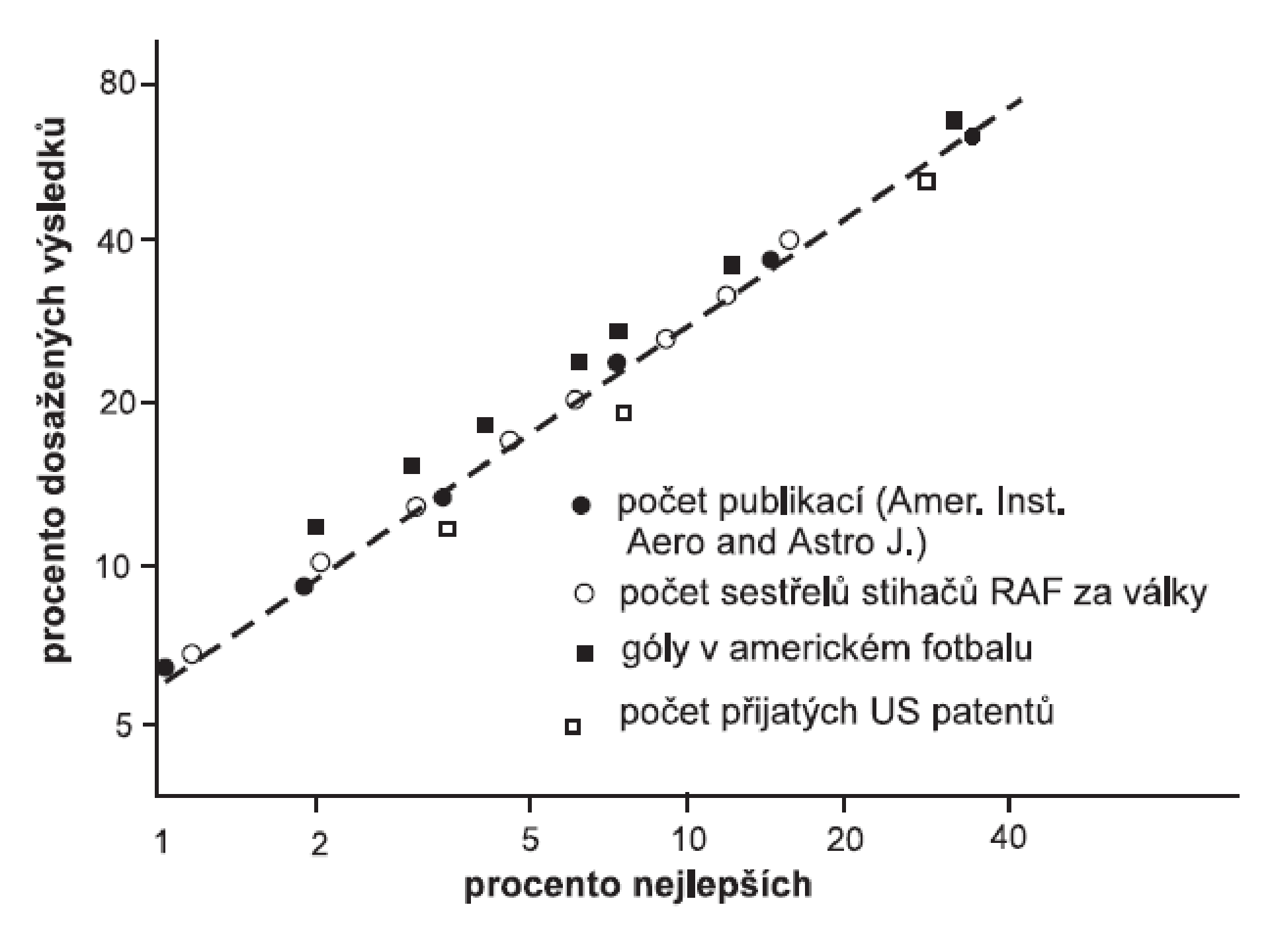
pro vhodné kladné p

Použití Rayleigtova a Planckova modelu

- Uvažujeme-li fakt, že část práce se po předání e přesune do údržby, platí následující pravidla
- **Rayleigt - zákon polovin**. V době dosažení maxima velikosti týmu jsme za polovinou doby řešeni a spotřebovali jsme přes polovinu práce
- **Planck – zákon třetin**, jsme asi v třetině doby řešení a spotřebovali jsme třetinu práce (a tedy budeme muset dvojnásobek dosud spotřebované práce ještě vynaložit)

Rozložení výkonnosti

Budiž $a(p)$ procento výsledků, které dosáhlo p procent nejlepších (např. $a(1)$ je procento branek, které nastřílelo jedno procento nejlepších. Uvidíme, že $a(1)=7$) Vyneseme-li graf $a(p)$ v dvojitě logaritmickém měřítku dostaneme následující obrázek.


procento dosažených výsledků
80
40
20
10
5
1
2
5
10
20
40
procento nejlepších
počet publikací (Amer. Inst. Aero and Astro J.)
počet sestřelů stihačů RAF za války
góly v americkém fotbalu
počet přijatých US patentů

Výsledky nejlepších

- Z grafu vyplývá, že

$$a(p)=cp^{1/2}$$

kde c nepříliš silně závisí na typu činnosti

Procento nejlepších udělá 7,5% výsledků, 20% nejlepších udělá polovinu práce, 40% nejhorších neudělá skoro nic

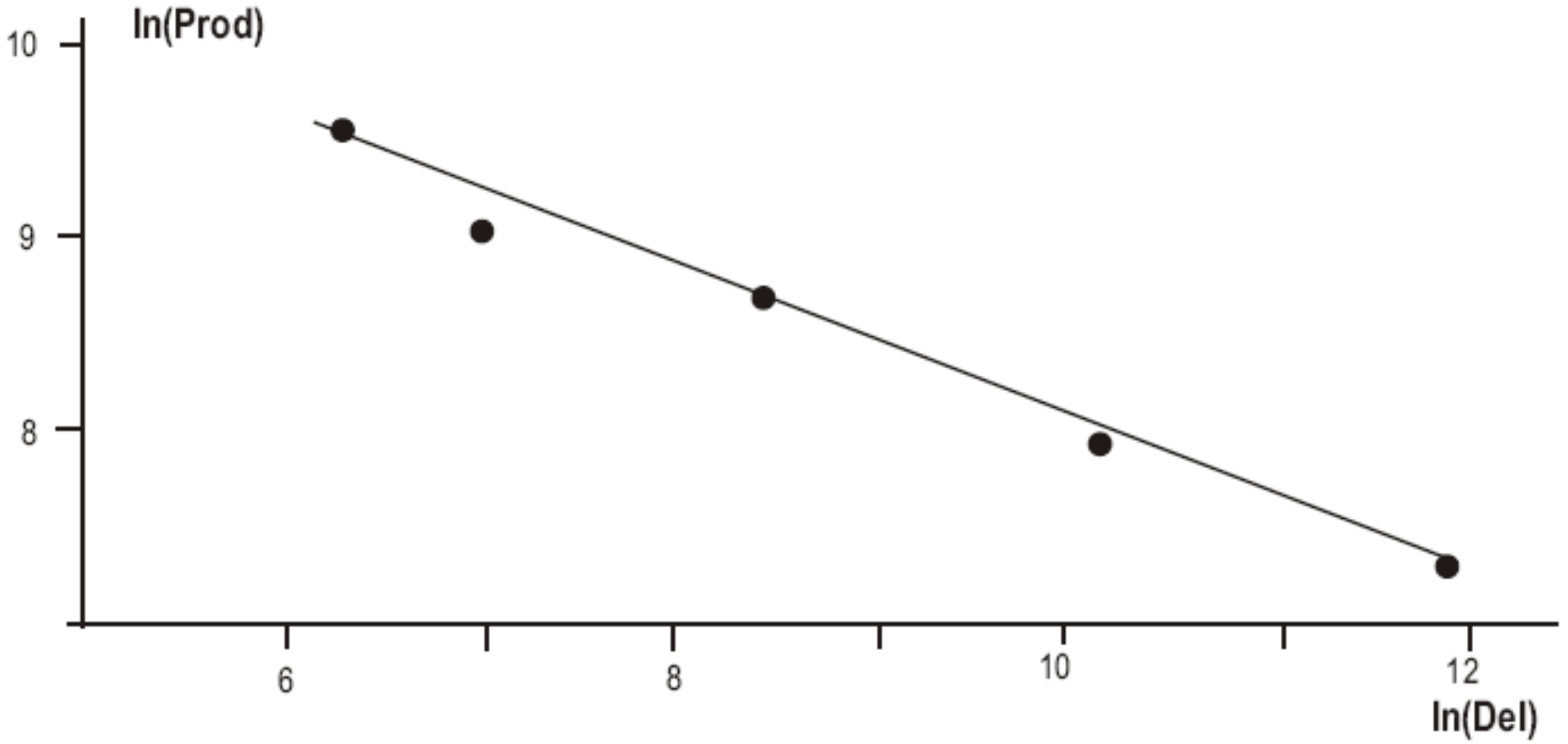
Seřadíme-li pracovníky podle výkonnosti, a budeme-li vynášet kolik udělal každý jednotlivec, bude mít příslušná funkce tvar

$$b(p) = 1/2cp^{-1/2}$$

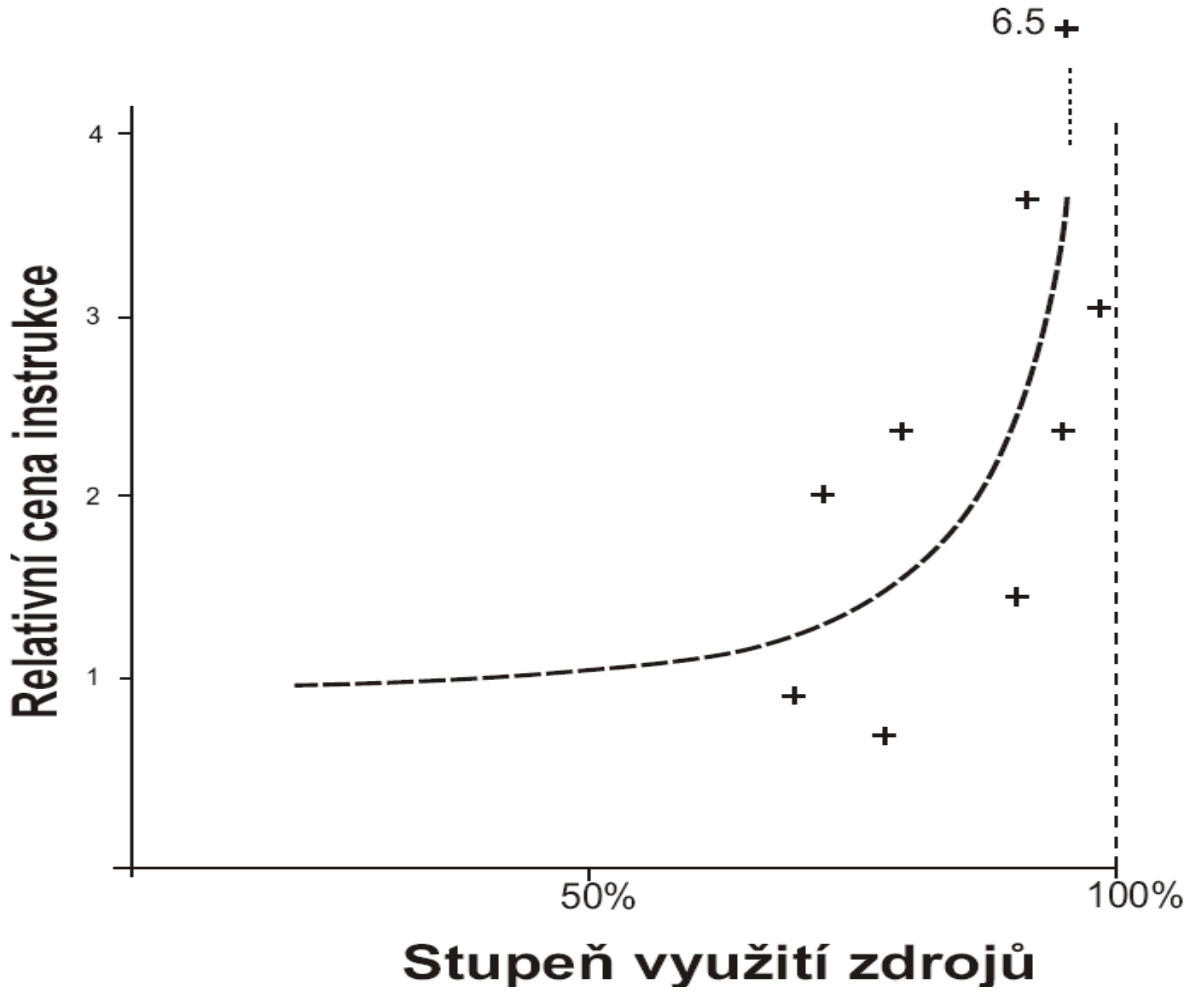
Použití

- Vědomí důležitosti talentu a kvality lidí
  - Nejlepší udělají nejen nejvíce výsledků na hlavu, ale také udělají i nejlepší výsledky
- Pokud jsou ve větší skupině problémy na straně kvalitních lidí, je nutné zkoumat, čím to je a zda to vadí (např. zda se nejedná o rutinní práce, kde se mohou kvalitní lidé cítit nevytíženi)

Závislost produktivity na délce programu pro malé programy, směrnice = -1/4.



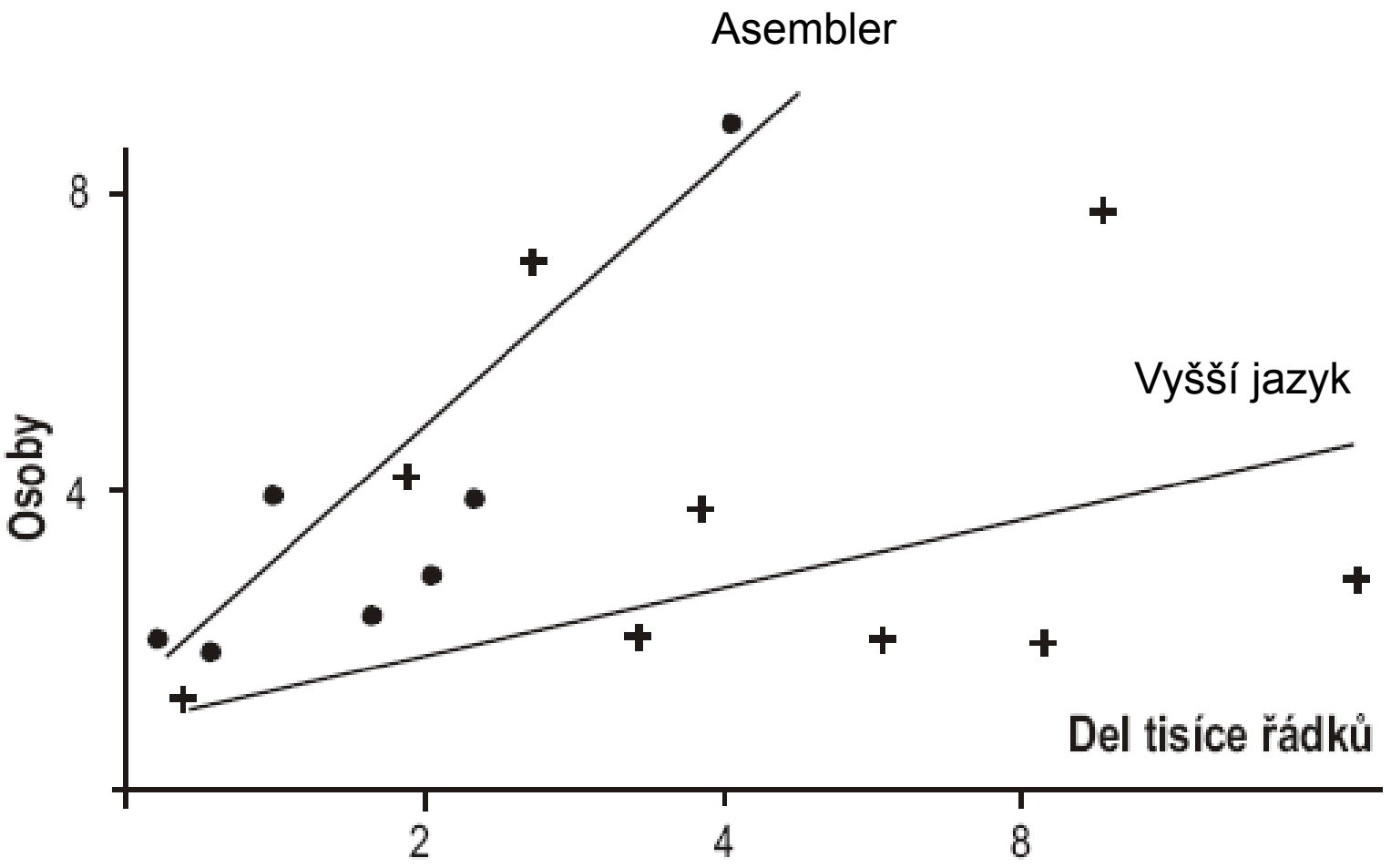
Vliv vytíženost HW



Využití

- U produktů, které nemají charakter masové spotřeby
  - Instalovat systém s 50-60% rezervou aby byl prostor na úpravy
  - Zvětšit reservu na alespoň 50%a, klesne- reserva pod 40%

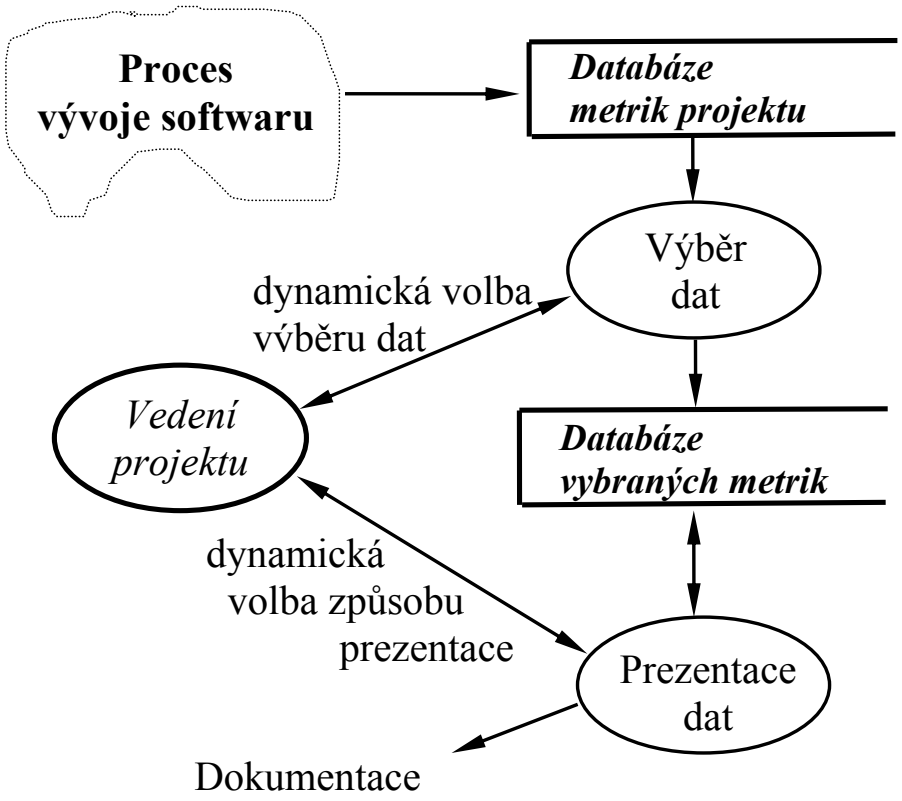
Náročnost údržby

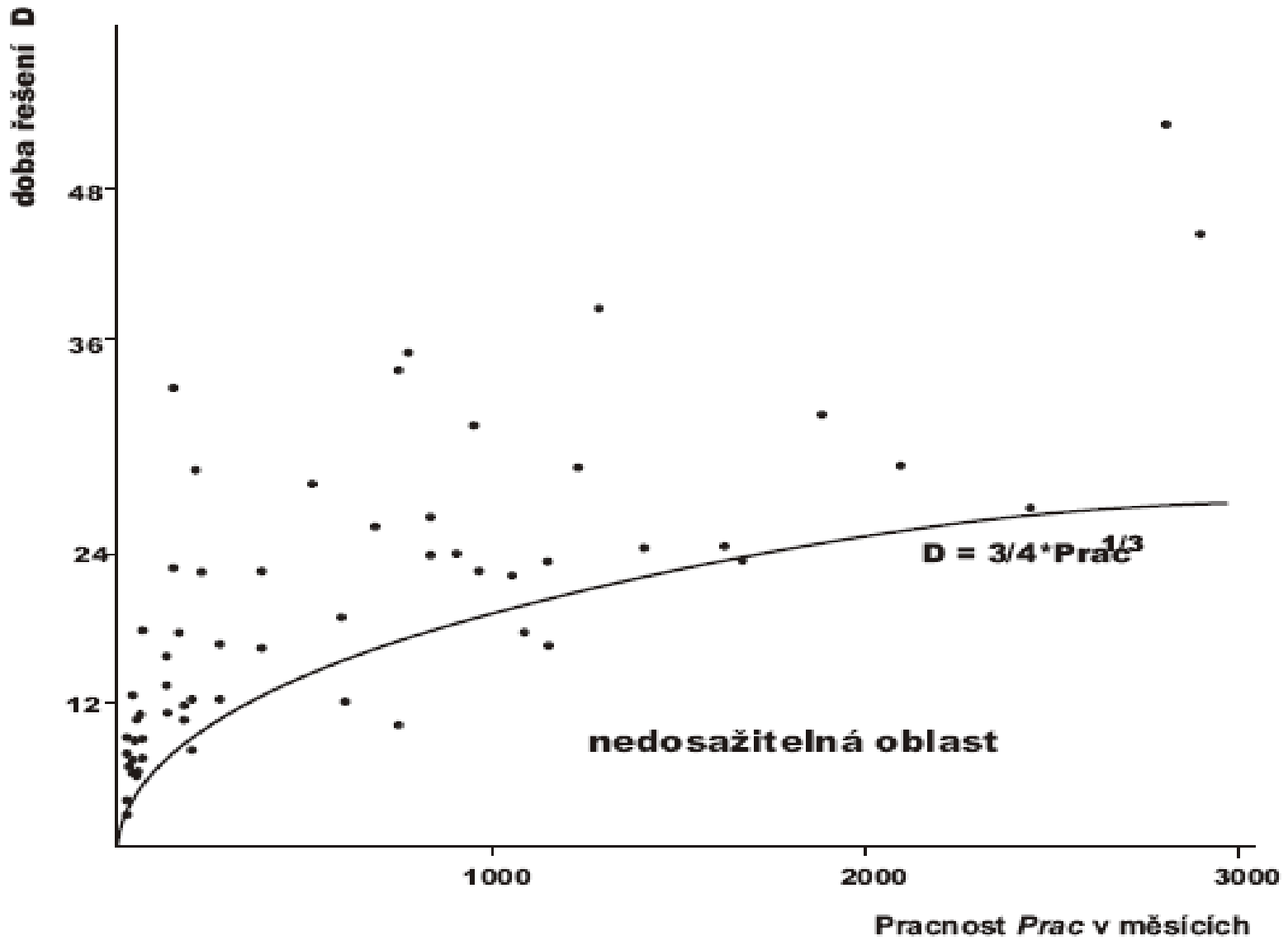


Využití

- Asembler je na údržbu (měřeno počtem osob udržujících systém na tisíc řádků) asi pětkrát pracnější než klasický programovací jazyk. U moderních systémů může být rozdíl ještě markantnější
- Poněvadž jeden řádek vyššího jazyka nahradí i několik řádek assembleru, je rozdíl produktivity alespoň 1:10
- Vyhýbat se asembleru, je-li to možné

Databáze metrik




doba řešení D
48
36
24
12
1000
2000
3000
Pracnost Prac v měsících
D = 3/4*Prac^1/3
nedosažitelná oblast