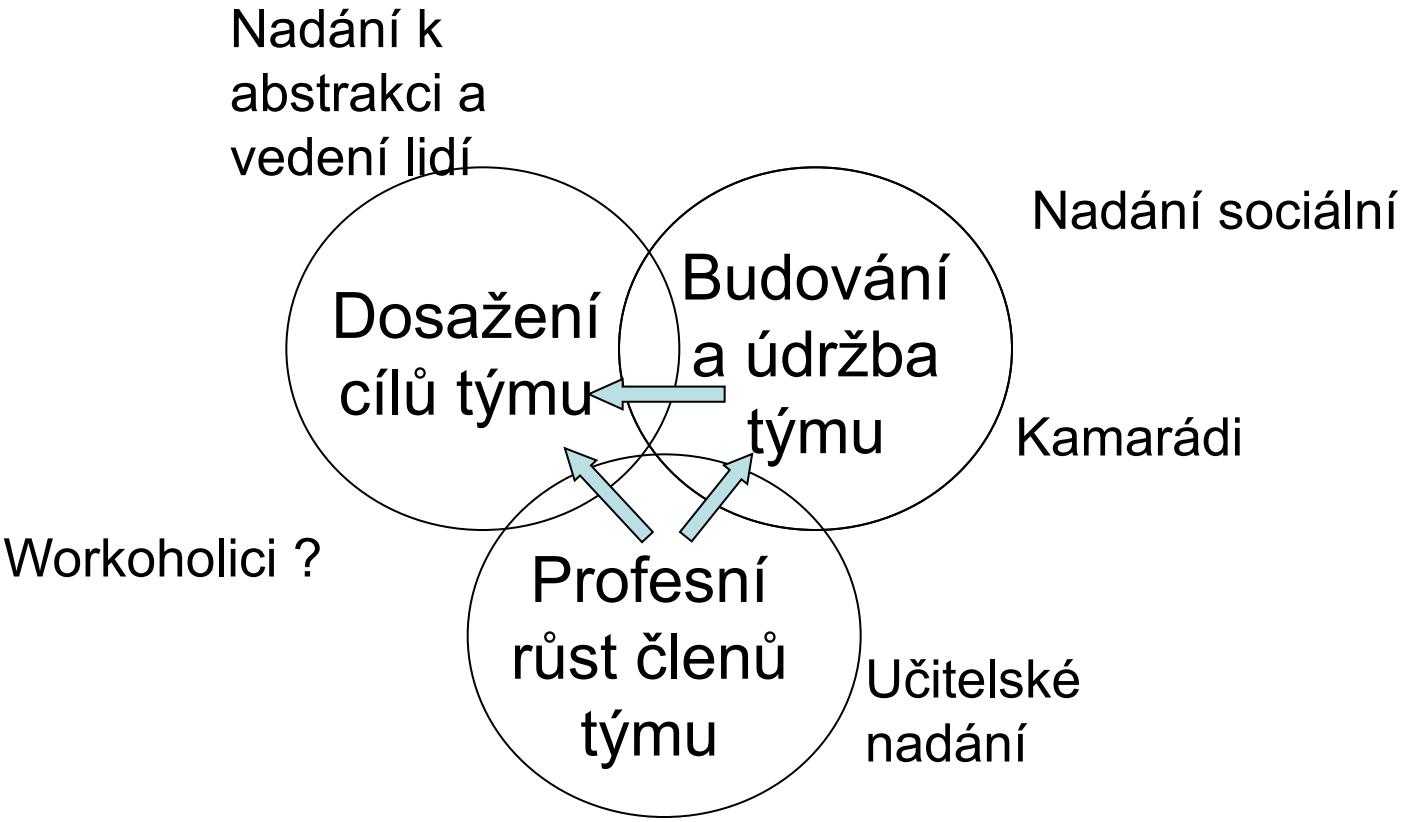
Kolik investovat do lidí a prostředků

„Neproduktivní investice“

Nejsou vidět

Tři aspekty činnosti týmu



Také rozvoj metod a nástrojů
tedy rozvoj prostředí

Kolik investovat do nástrojů

- Kolik dávat do „neproduktivních“ činností, takových, jejichž výstup není částí projektu
  - HW
  - Podpůrný SW
  - Nástroje
    - Kupované
    - Vlastní
  - Vzdělávání lidí

Himaláj a Stolová hora

- Kolik investovat do lidí a prostředků (a vlastně i do specifikací). Mám prostředky na n člověkoměsíců programování, prostředky na m člověkoměsíců investuji do podmínek práce, tím zvýším produktivitu $f(m)$-krát.

$f(m)$ by měla růst, musí být $f(0)=1$, tj. žádná investice, žádná změna

Výkon tedy bude

$$Q_n(m) = (n-m)f(m)$$

Pak Q nabývá maxima v bodě, kde

$$0 = -f(m) + (n-m)f'(m)$$

Zvolme $f(m)= 1+cm/n$.

Je to dosti konzervativní odhad, f bývá superlineární.

Přínos bývá i v jiných projektech, investice do specifikací mají podobné efekty, podobají se efektům nových nástrojů

Himaláj a Stolová hora

Po dosazení do vzorce pro derivaci Q dostaneme podmínku pro maximum

$$0 = -(1+cm/n) + (n-m)c/n$$

$$0 = -1 - cm/n + c - cm/n$$

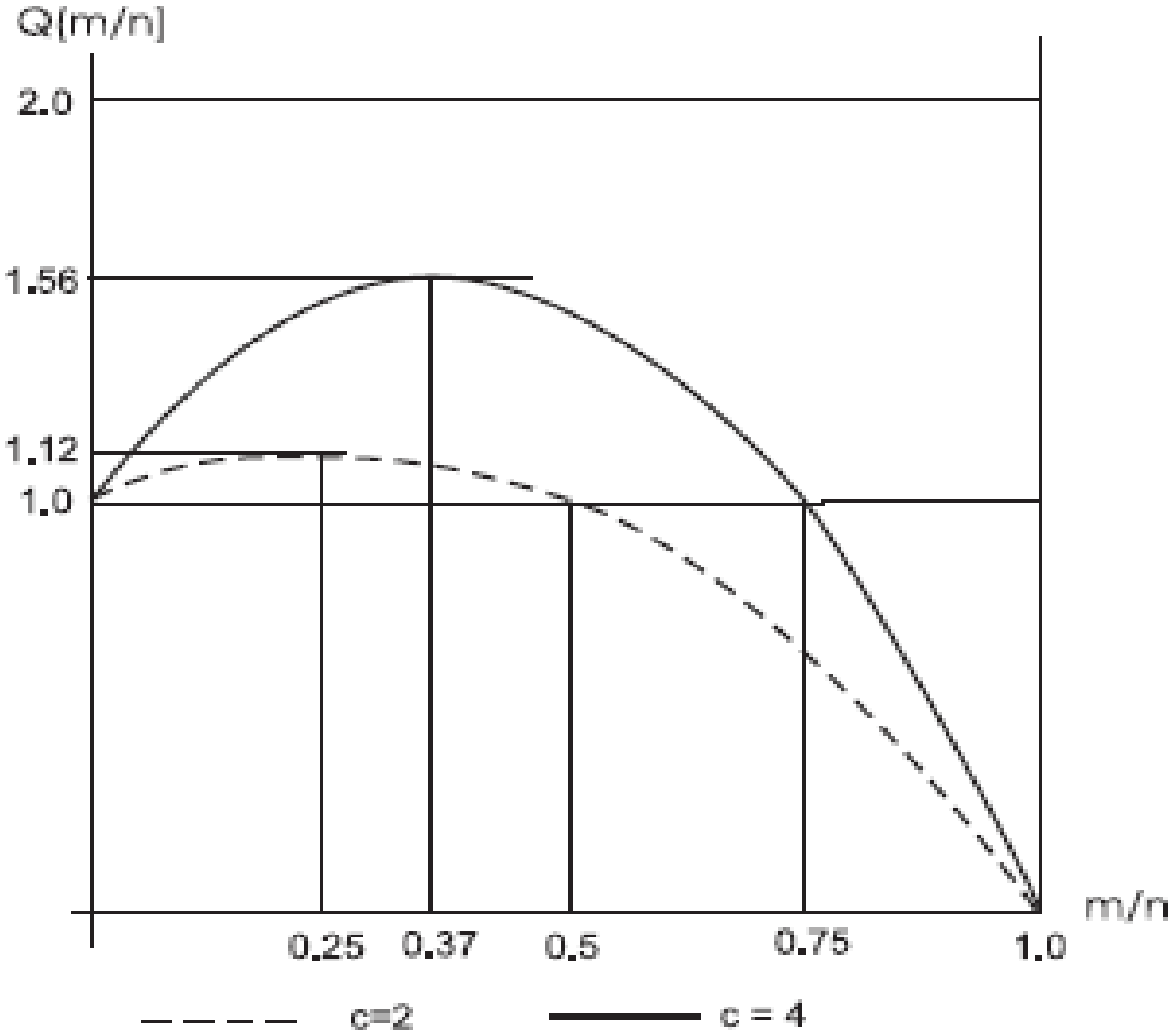
Převedeme-li m/n na levou stranu, dostaneme

$$2cm/n = c-1$$

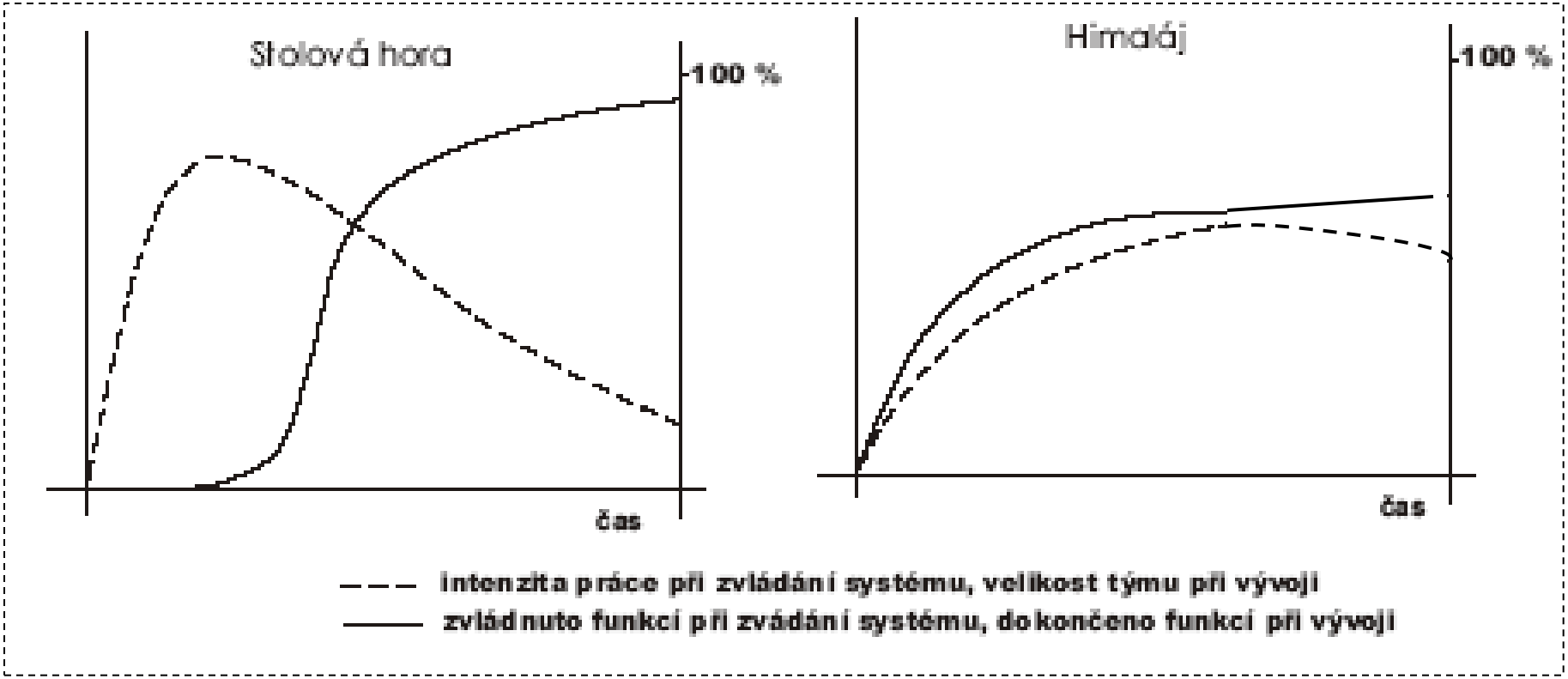
Čili

$$m/n = 1/2 - 1/(2c)$$

Himaláj a Stolová hora

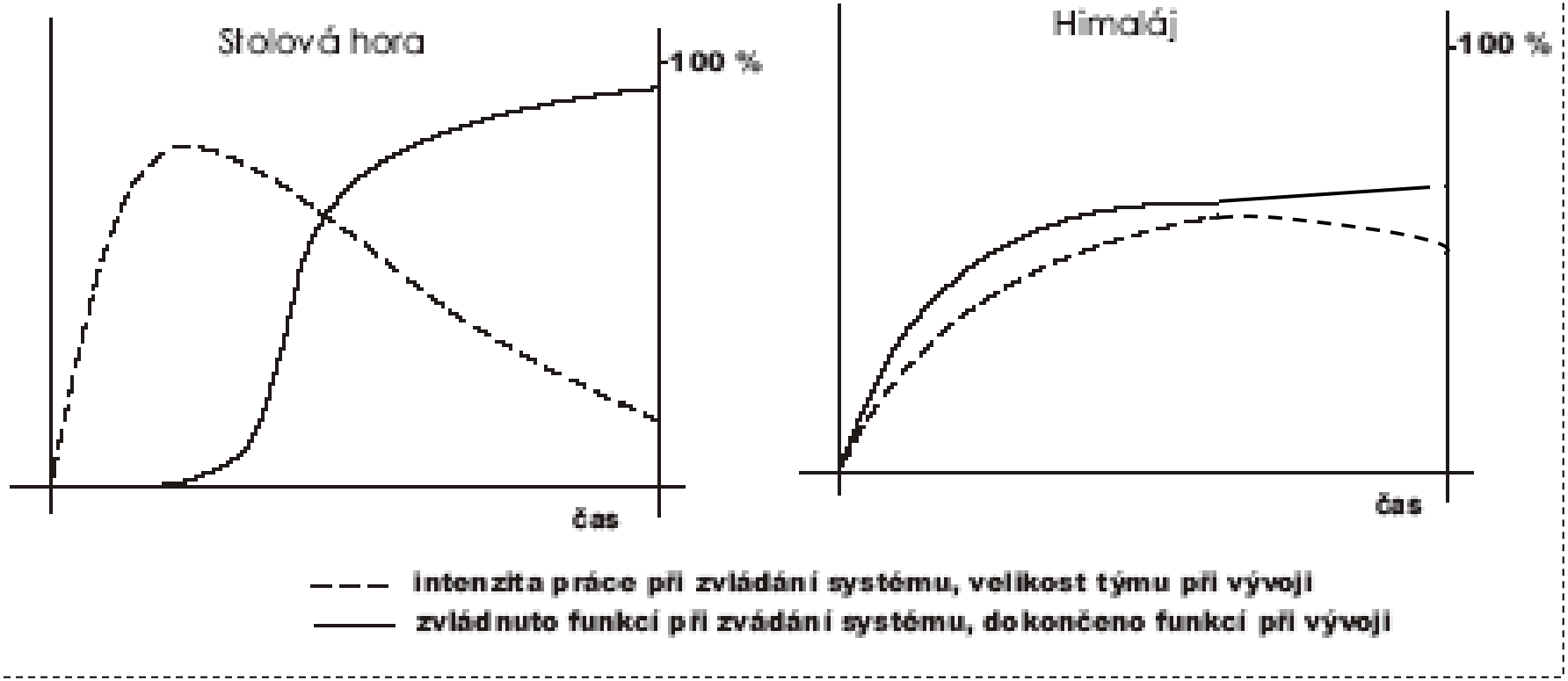


Himaláj a Stolová hora



Stolová hora – dlouho nic, pak prudký výstup na vrchol, tam je pohodlí
Himaláj – začnu brzy stoupat, vrchol stále v nedohlednu

Himaláj a Stolová hora



Riziko Himaláje: Dlouho nejsou hmatatelné výsledky, větší riziko, že jsme vyhodili peníze a promarnili čas ⇒ raději zkusit na menším projektu nebo službě v SOA

Himaláj a Stolová hora

Tabulka bodů maxima

c	1/2-1/(2c)	$\max(Q(m)/n)$	zvýšení %
2	0.25	9/8	12.5
3	0.33	4/3	33.3
4	0.37	25/16	56.2
6	0.42	49/27	104.2
8	0.44	81/32	153.2

Tab. 12.1: Efekty zvýšení výkonu při použití nástrojů.

Himaláj a Stolová hora

1. Přínos nového nástroje se ovšem většinou neomezuje pouze na daný projekt. Přínosy v dalších projektech mohou být značné. Mnoho se ušetří na údržbě. Příkladem správnosti této úvahy je jazyk C při vývoji prvé verse Unixu.
   – Je tedy třeba dodržovat pravidlo 1/3 na vedlejší výdaje prakticky vždy, kdy je v dosahu nástroj přinášející dostatečné efekty.
2. Doba zvládnutí kupovaného nebo doba vývoje nového nástroje je dána vlastnostmi nástroje samotného. To znamená, že m/n nebude přesně vyhovovat podmínce maxima, takže skutečný efekt bude pak o něco menší, než je uvedeno v následující tabulce.

Himaláj a Stolová hora

Máme-li více nástrojů, pak můžeme postupovat tak, že postupně přidáváme nástroje s náklady m_1, m_2, m_3 atd. Prvý nástroj dá zvýšení produktivity c_1, prvé dva c_2, atd. Implementujeme nebo vyvíjíme tolik a takové nástroje, aby

$$\Sigma_1^n m_i \leq 1/2 - 1/(2c_n)$$

a c_n bylo co největší

Himaláj a Stolová hora

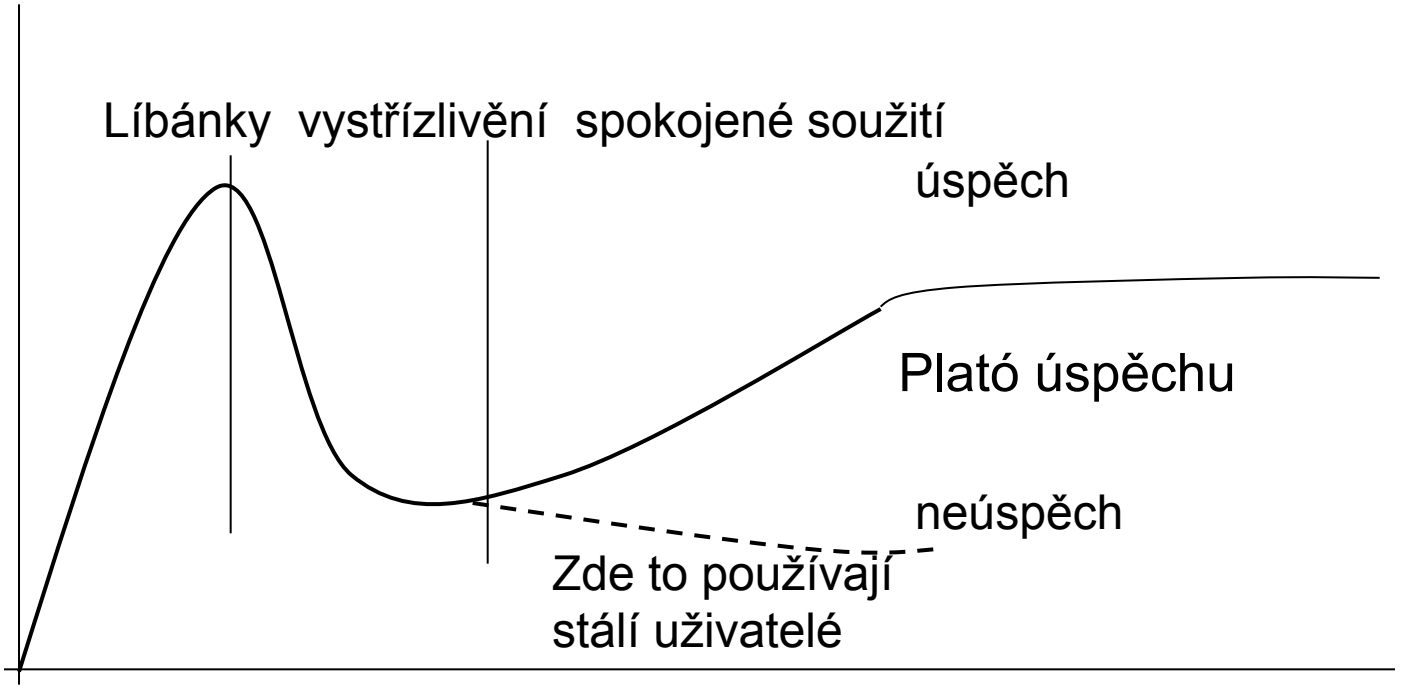
- Funkci $f(m)$ jsme zvolili poněkud spekulativně. K obdobným výsledkům ale dospějeme i pro jiné volby tvaru funkce f.
- Pro větší n si mohu dovolit vývoj složitějších a tedy účinnějších nástrojů, tam lze při správné strategii docílit vynikající výsledky
- Neuvažujeme, že hlavní přínos může být v úspoře nákladů na údržbu (přehlednost, snadnost oprav)
- Nástroje mohou zlepšovat logiku a kromě toho i umožňovat znovupoužitelnost a efektivitu
- Je třeba rozvíjet prostředí a nástroje (vývoj, nákup), musím ale na to mít schopné lidi, *stejný efekt ale může mít investice do znalostí lidí*

Modernost metodiky

- Nový nástroj se zprvu používá tam, kde se staré přístupy neosvědčily, proto jsou výsledky zprvu skvělé. Další důvod neúspěchu je, že ho používají inovátoři a ne běžní uživatelé (podobné efekty existují i ve školství)
- Pak se narazí na meze a inovátory to navíc již nebaví a přijde rozčarování, někdy neoprávněné
- Nakonec upadne nástroj buď do zapomnění, nebo se osvědčí a je rutinně používán – plató úspěchu. Někdy to trvá řadu let (u objektové orientace to bylo více než deset let)
- Je třeba být u novinek zdravě ale ne přehnaně skeptický

Modernost nástroje

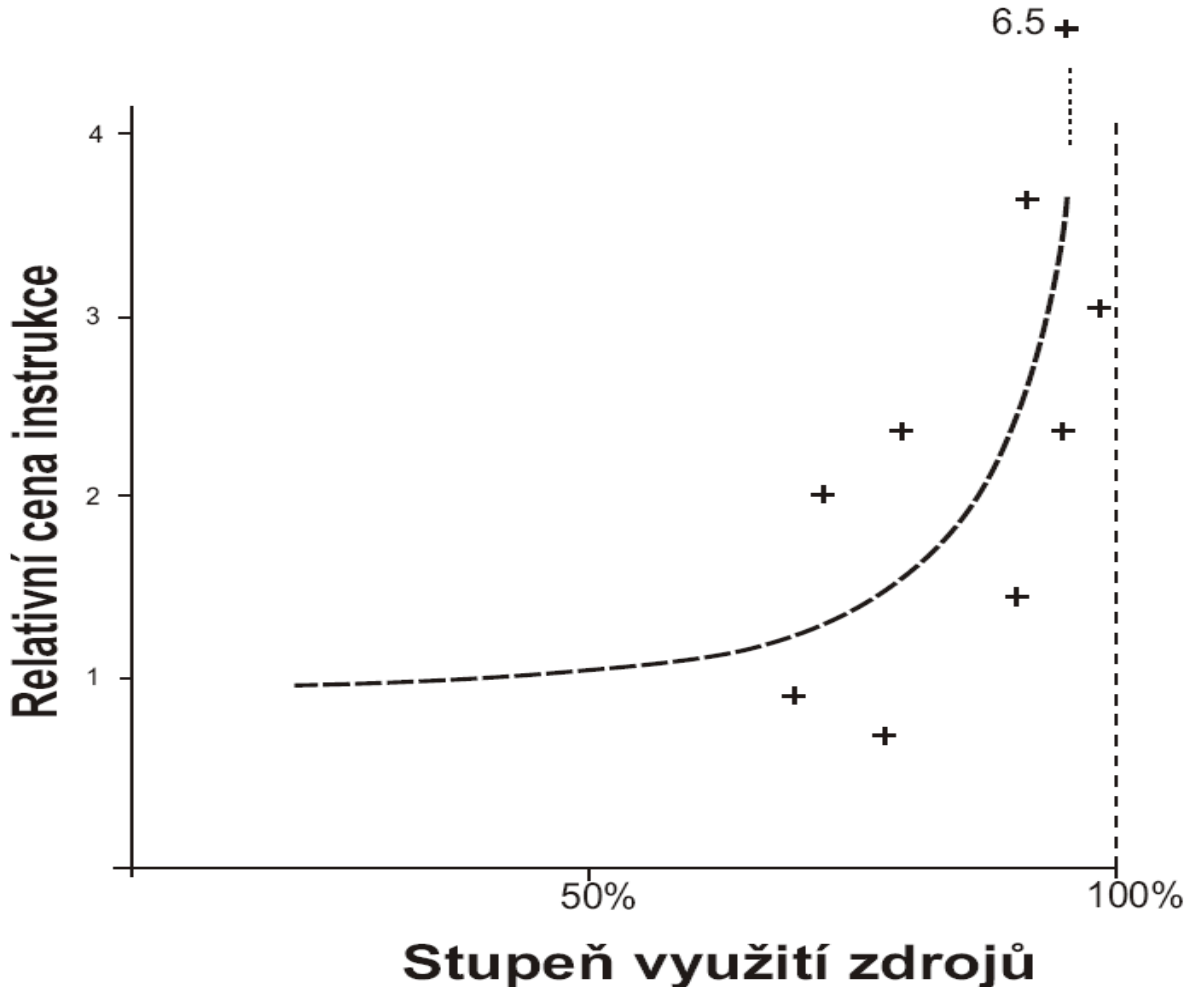
- Funkce množství positivních ohlasů



Jak rozvíjet kapacity

Je nutné počítat s tím, že požadavky porostou a úpravy ve svém důsledku rovněž zvyšují nároky (záplatovaný program je delší, než by byl, kdyby se celý napsal znovu

Vliv vytíženosti HW, data z několika projektů



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%
  - V některých podnicích zvětšují reservu alespoň na 70%, klesne-li pod 50%
    - Příklad. Letecký dispečing

Mnohdy se sleduje úspora na nesprávném místě

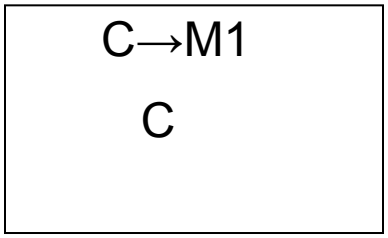
- Je to zkušenost z mnoha praktických projektů
- Neanalyzuje se dostatečně nutnost neustálých změn během údržby
- Zzbytečně šetří na koncových zařízeních i menších podnicích

Podobné jevy

- Kapacita paměti
- Možnosti volby překrmeného kupovaného SW
- Zapomíná se, že v mnoha případech je investice do kapacit relativně malá a že ji dále snižují moderní technologie SW, např. cloud
- Zahrnout do úvah Moorův zákon

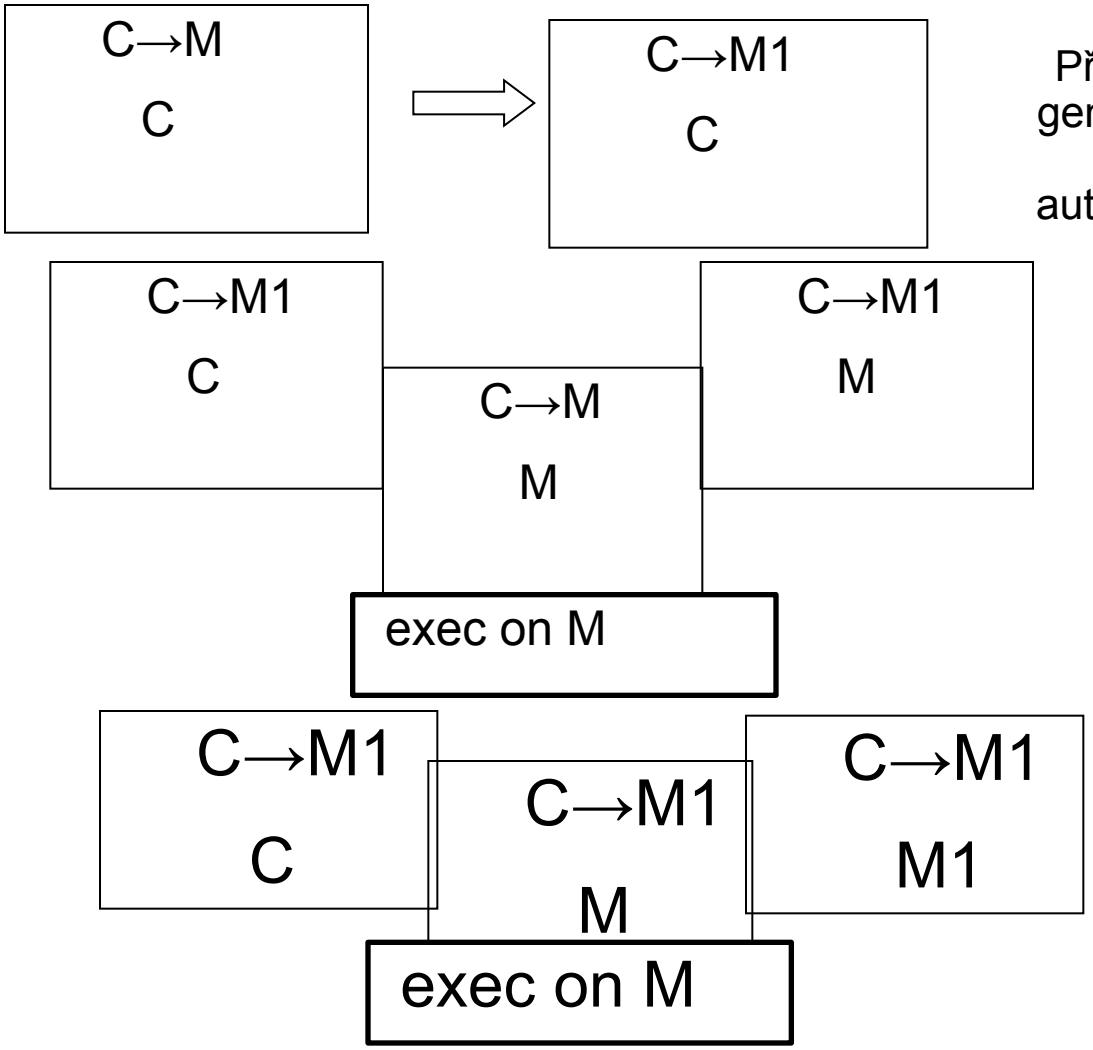
Dobrý nástroj je vždy výhodou

Kompilátor



Kompilátor
z C zapsaný v C

Přenos kompilátoru

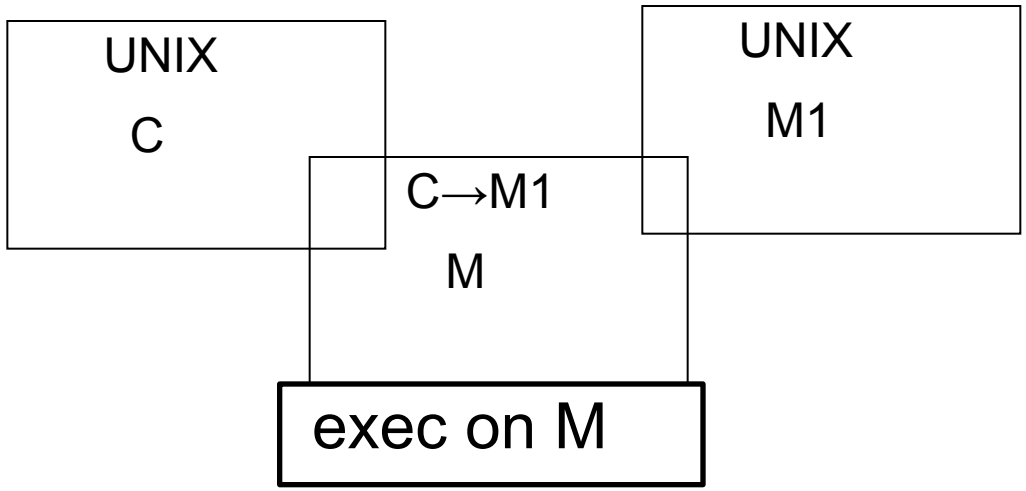


Přepíše se generátor kódu jen zčásti automatizovaně

Kompilátor v C přeložen C kompilátorem běžícím na M, získán Křížový kompilátor běžící na M překládající pro M1

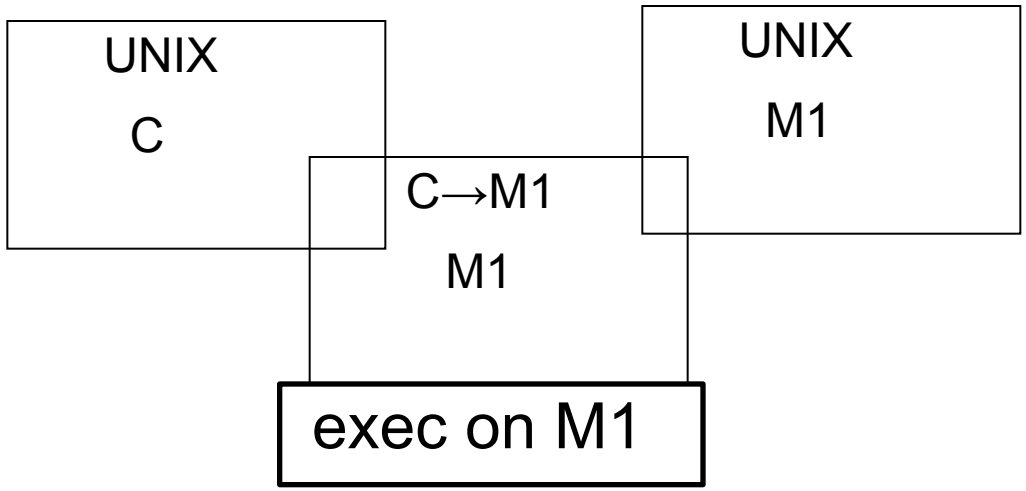
Cílový kompilátor

Přenos UNIXU

- 

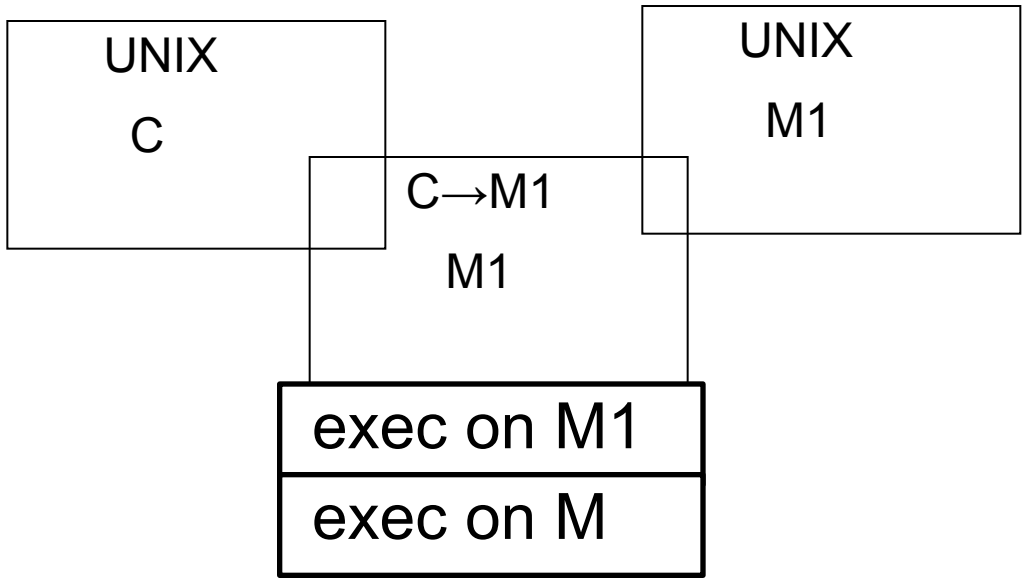
Je ale nutné přepsat drivery, cca 10% nákladů

Přenos UNIXU

- 

Je ale nutné přepsat drivery, cca 10% nákladů

Přenos UNIXU, virtuálně



Je ale nutné přepsat drivery, cca 10% nákladů