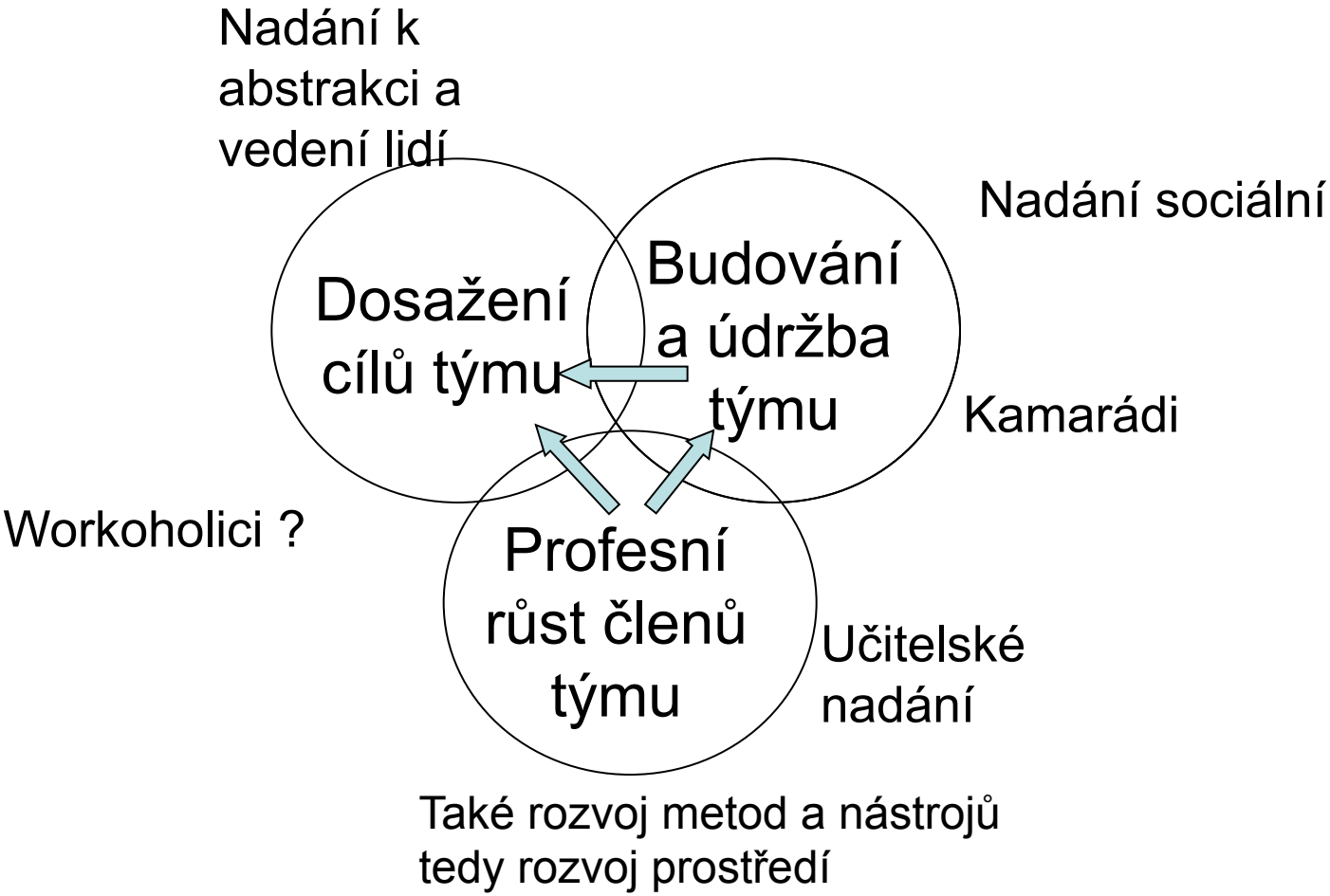
# Kolik investovat do lidí a prostředků

„Neproduktivní investice“

Nejsou vidět

# Tři aspekty činnosti týmu



# 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

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

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

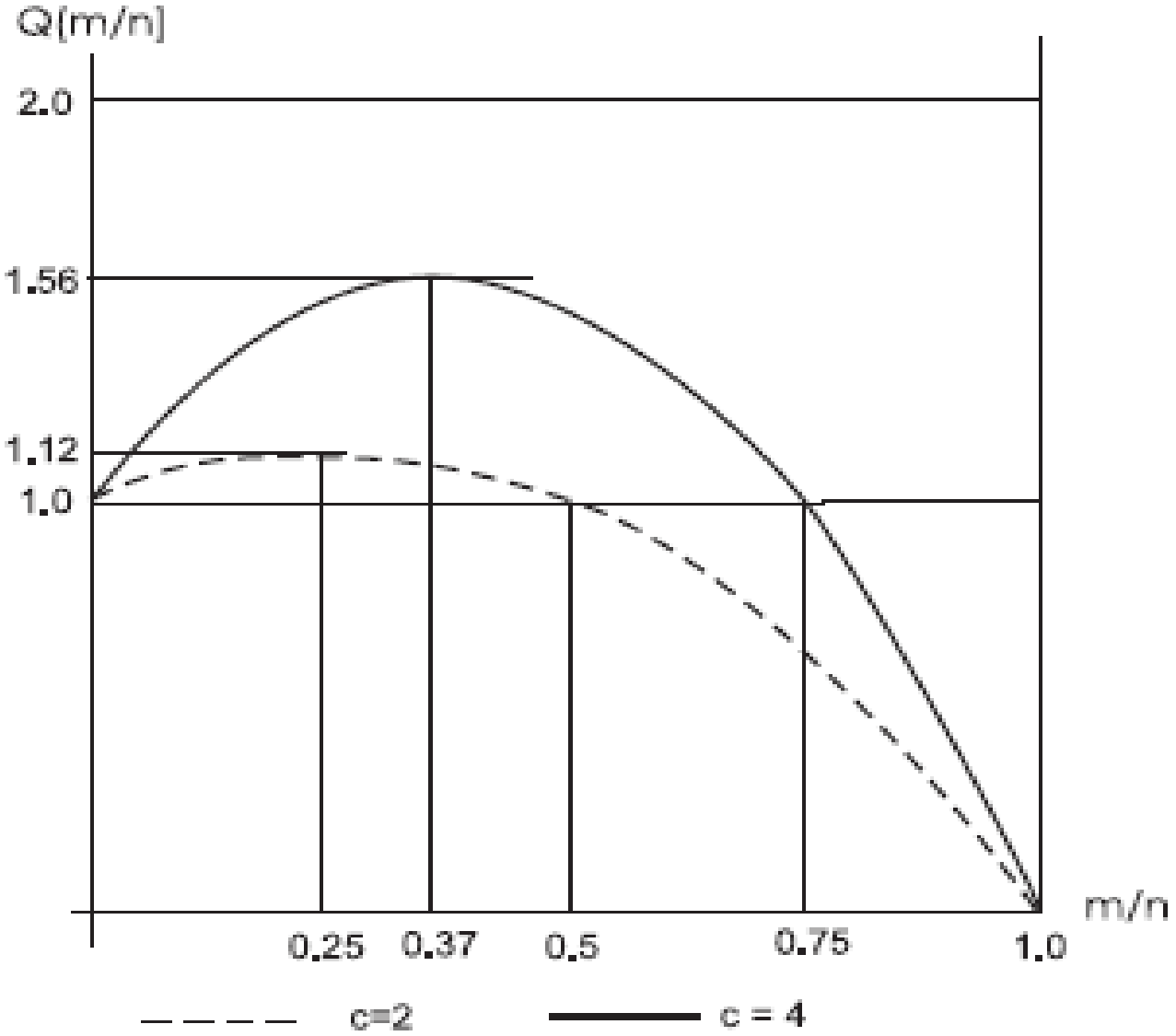
Převedeme-li $m/n$ na levou stranu, dostaneme

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

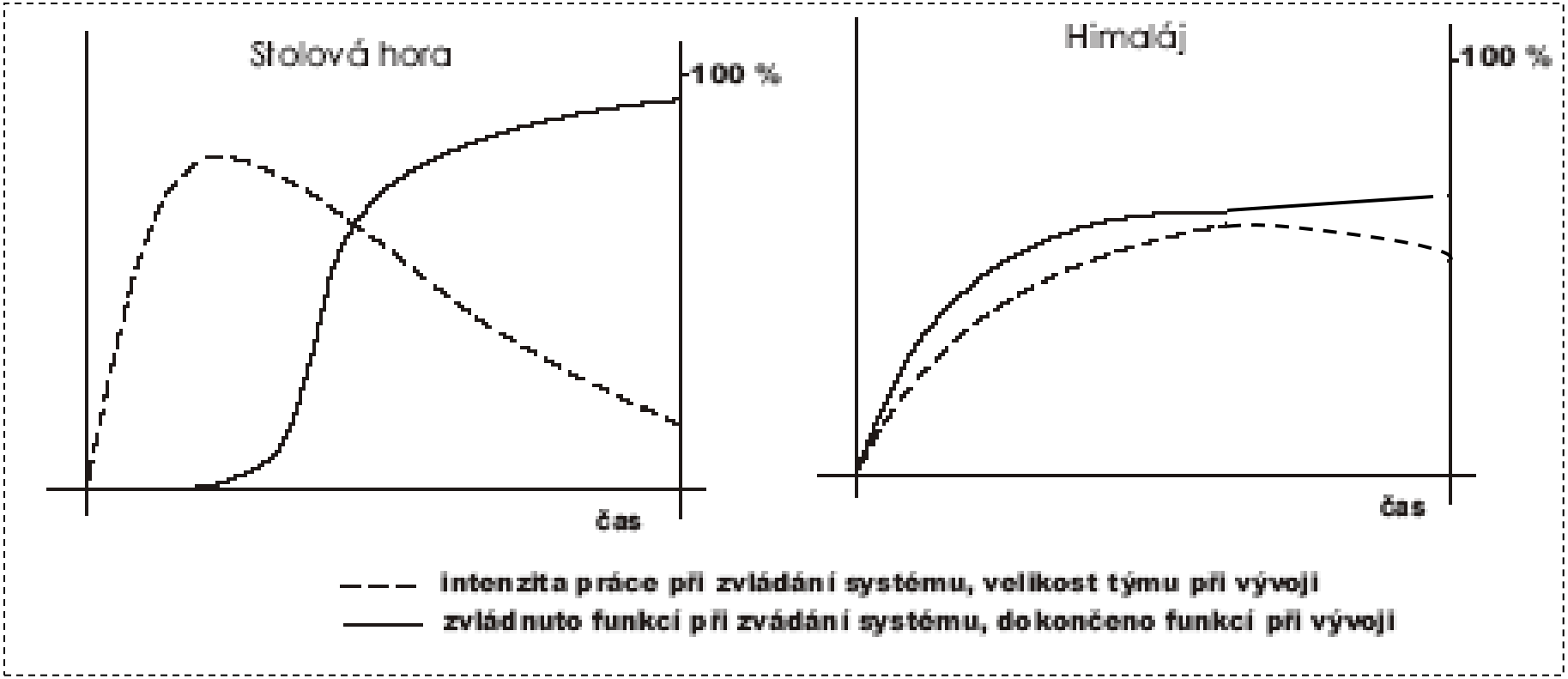
**Čili**

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

# Himaláj a Stolová hora

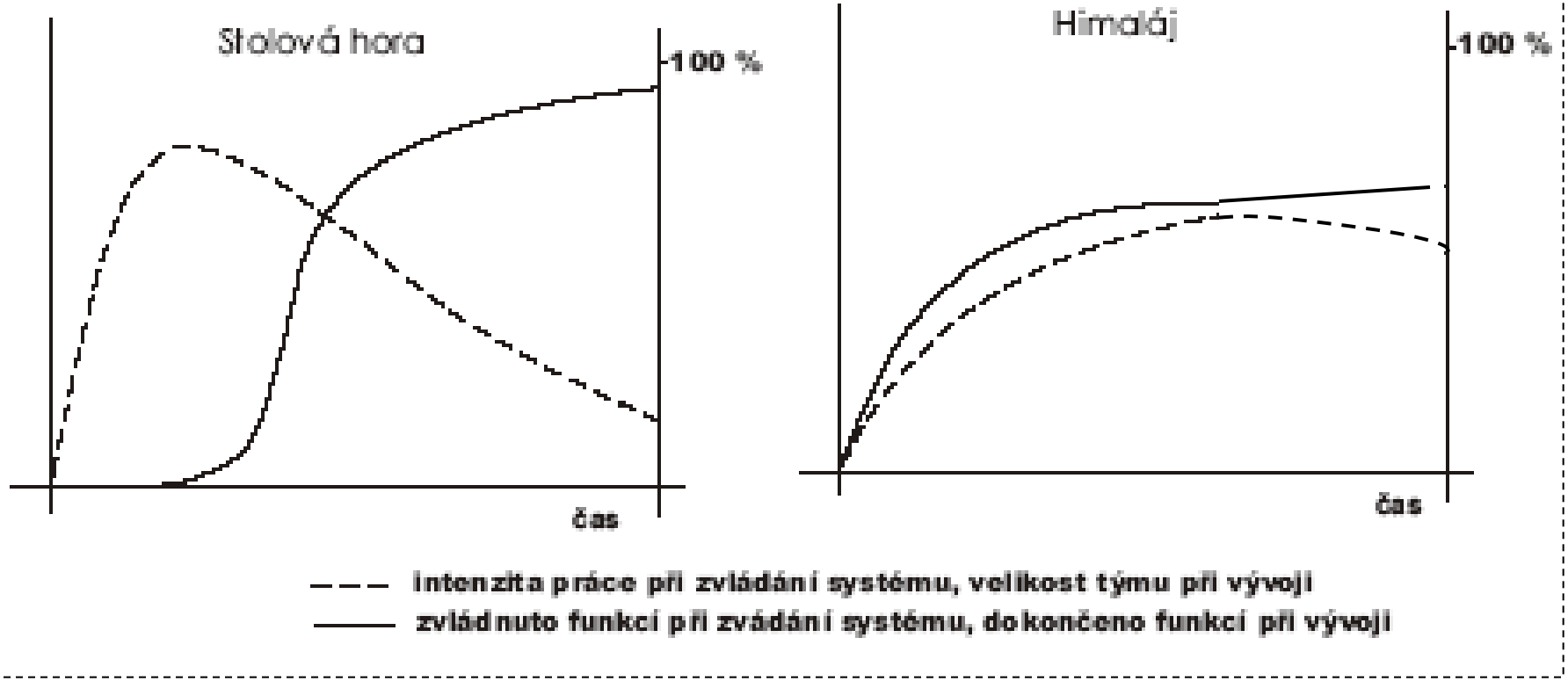


# Himaláj a Stolová hora




*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

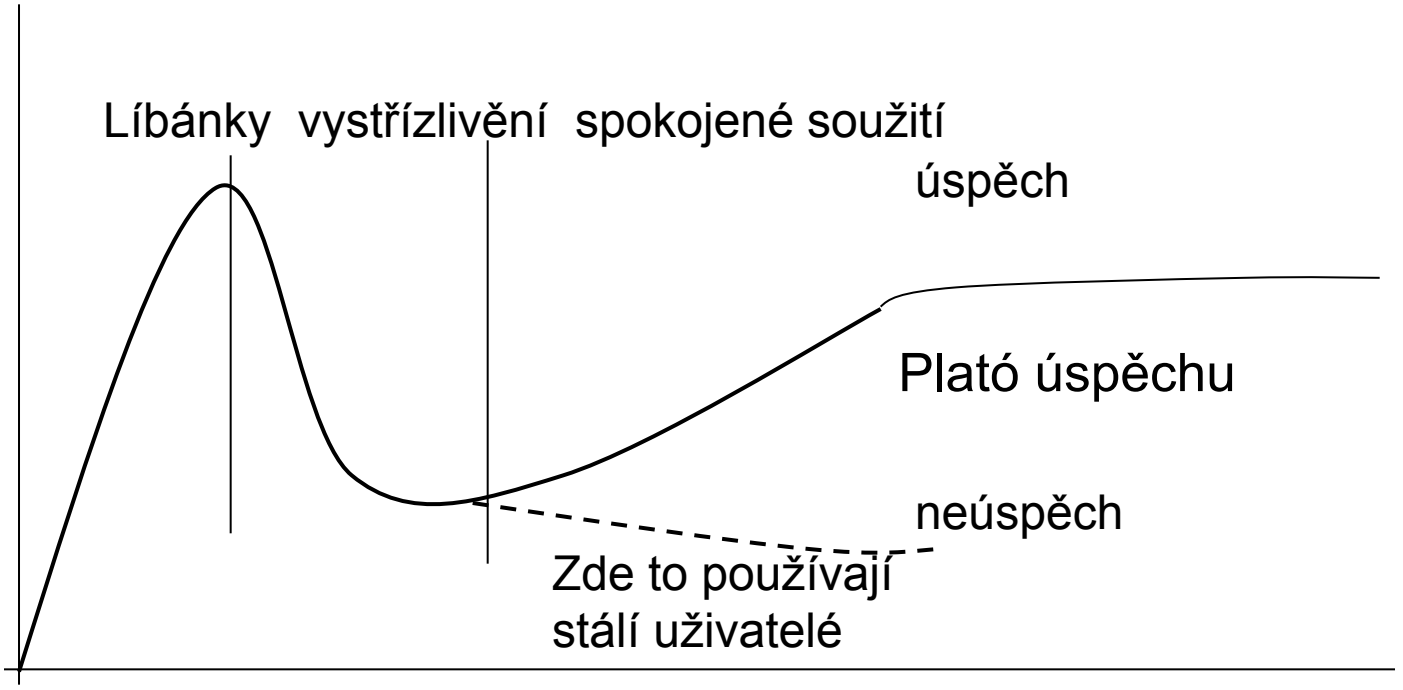
- Funkci $f(m)$ jsme zvolili poněkud spekulativně. K obdobným výsledkům ale dospějeme i pro jiné volby tvaru funkce $f$.
- Pro větší $n$ si mohu dovolit vývoj složitějších a tedy účinnějších nástrojů, tam lze při správné strategii docílit vynikající výsledky
- Neuvažujeme, že hlavní přínos může být v 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

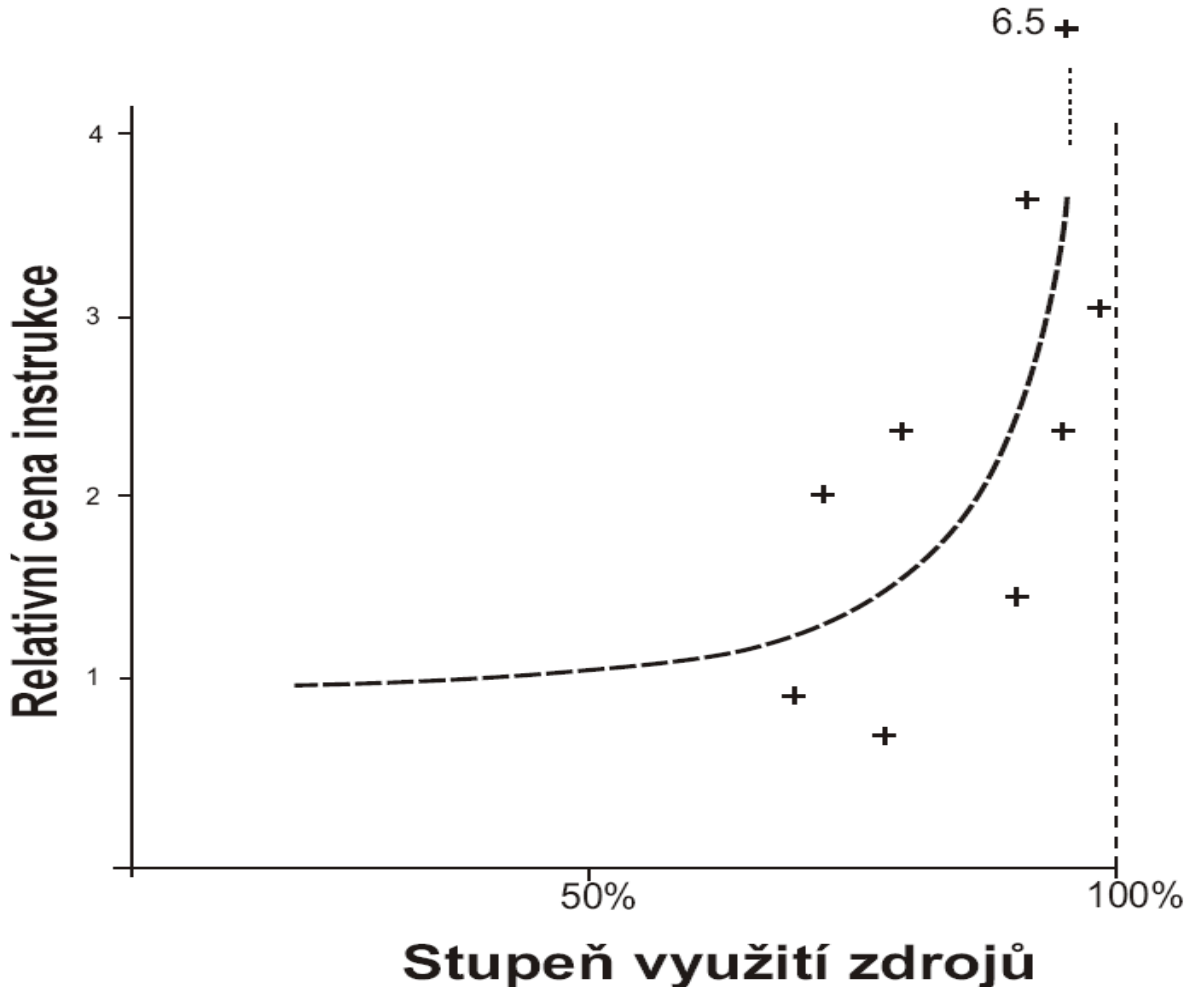
- Funkce množství positivních ohlasů



# 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 až na 70%, klesne-li pod 50%

# Podobné jevy

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

# Vývoj uživatelského rozhraní

Návrh a vývoj uživatelského rozhraní je specifickou částí vývoje systému s vlastními problémy a metodami a standardy

# Faktory akceptovatelnosti softwaru

1. Sociální a společenská akceptovatelnost.
   1. Potřebné
   2. Dá se s danými lidmi rozumně používat
2. Praktická akceptovatelnost.
   - 2.1 Užitečnost (Usefulness).
   - 2.1.1 Funkčnost.
     - 2.1.2 Použitelnost (Usability).
   - 2.3 Cena.
   - 2.4 Kompatibilita, přenositelnost.
   - 2.5 Modifikovatelnost.
   - 2.6 Spolehlivost.
   - 2.7 Dostupnost pro všechny uživatele.

# Faktory použitelnosti softwaru

- snadnost naučení,
- efektivnost při používání,
- dobře se pamatuje, jak systém používat,
- málo chyb uživatele způsobených špatným ovládáním rozhraní,
- subjektivní příjemnost práce se systémem pro uživatele,
- dobré ergonomické vlastnosti

# Efekty uživatelského rozhraní

- Ovlivňuje spokojenost zákazníka se systémem
- Je důležité i z čistě obchodního hlediska. Je „obalem“ softwaru
- Podstatně ovlivňuje složitost ovládání IS (až 10% efektu nepočítaje důsledky stresu).
- GUI je ergonomicky náročné
- Čtvrtina požadavků uživatelů se obvykle týká rozhraní

# Použitelnost

Jak snadno se se systémem pracuje.

Jde o vlastnost uživatelského rozhraní, v SOA podstatně ovlivňuje SW inženýrské vlastnosti systému

Viz Nielsenovy knihy na téma Usability

# Přesvědčit o významu rozhraní

- Přesvědčit management  a  někdy i koncového uživatele, že je to problém
- Získat prostředky (někteří doporučují i více než deset procent nákladů na vývoj)
- Zařídit, aby se vývoje a především testování účastnili budoucí (koncoví) uživatelé
- Vytvořit nástroje pro sledování rozhraní a jeho zlepšování během provozu
- Ve vývojovém týmu jsou žádoucí specialisté na UI
- Funkce systému mají umožňovat kvalitní rozhraní (exity), srozumitelné zprávy

# Začátečníci a pokročilí

- *Noví uživatelé (dále začátečníci)* dávají přednost takovým nástrojům, které nevyžadují rozsáhlé zaučování předem  a jsou výhodné pro zvládání systému metodou pokusů a omylů. Používají často nápovědu
- *Dlouhodobí uživatelé* budou mít tendenci používat klávesové zkratky a různá urychlení, které většinou znamenají i větší nároky na paměť.
  - Jak usnadnit přeškolení začátečníka pokročilého uživatele

# Začátečníci a pokročilí

- Problém je v tom, že se používáním systému ze začátečníka stává pokročilý
- Není to  ale úplně hladký proces, je na to třeba myslet při návrhu UI, který by měl upozorňovat na možnosti zrychlení
- Neobejde se to ale ani bez výcviku (seznamování těch, pro které to má větší význam, s možnostmi zrychlení)

# Nápověda

- Kombinovat s tutoriálem, interaktivním manuálem a demo
- Interaktivní kontextová nápověda by měla být chápána jako pomoc v nouzi. GUI by mělo, pokud možno, být intuitivní, konsistentní a samovysvětlující

# Začátečníci a pokročilí

- Zkušení uživatelé často preferují znakové rozhraní nejen proto, že s ním lze pracovat rychleji, ale také proto, že nemusí tolik pozorovat obrazovku a nemusí tolik používat myš. Klávesové rozhraní je ergonomicky výhodnější. Lze lépe kontrolovat, co se opravdu provádělo
- Je žádoucí obě techniky (ovládání myší a ovládání znaky) kombinovat pro dosažení optimálního výsledku.

# Další výhody znakového rozhraní

- Snazší žurnál (log) pro kontrolu práce
  - Rychlejší vytváření „byznys inteligence“
- Snazší vytváření procedur
- Snazší vytváření výkonných příkazů (srv SQL a grafické rozhraní v Accesu)
- Hlavní bariéra – pro mnohé moc složité, nebezpečí příliš strmé křivky učení

# Životní cyklus uživatelského rozhraní

- Analýza požadavků, specifikace požadavků.
- Návrh UI.
- Realizace prototypů a jejich testování s uživateli.
- Integrace nejlepších nápadů
  - iterativní vylepšování
- Integrace UI se zbytkem systému a testování UI společně s uživateli.
- Sledování vlastností UI za provozu a další vylepšování vlastností UI.

# Životní cyklus uživatelského rozhraní

1. Při vývoji UI je ještě obtížnější než při návrhu funkcí odhadnout optimální vlastnosti UI, protože ty závisí na psychologii, dovednostech a znalostech budoucích uživatelů.
2. Testování UI je možné pouze tak, že systém testují uživatelé. Testování se provádí sledováním práce uživatelů.
3. Vlastnosti uživatelů se během používání systému vlivem zkušeností mění. UI musí vždy počítat se začátečníky i s pokročilými.
4. Důležitou roli hrají problémy ergonomie

# Životní cyklus uživatelského rozhraní

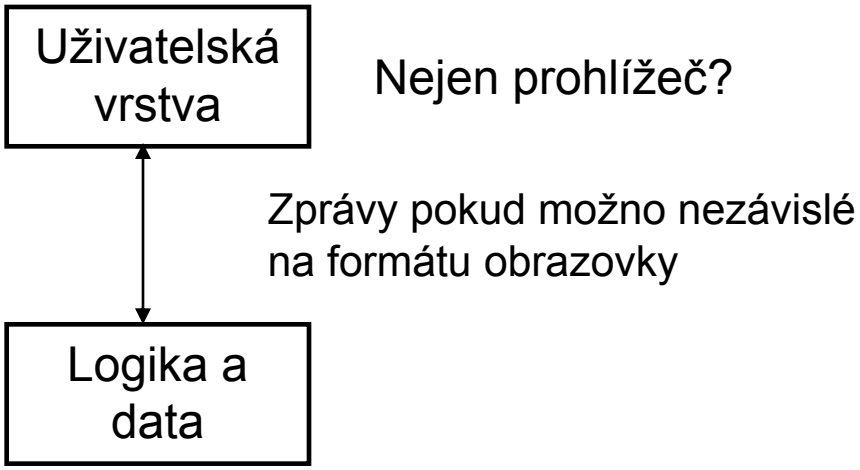
Při vývoji UI se porovnávají různá řešení a vybírají se nejlepší nápady

Obvykle se používá iterativní vývoj (námitky uživatelů se akceptují, hledá se vylepšování metrik a nová řešení se znovu testují)

# Činnosti při specifikaci požadavků

- Poznání uživatele
- Analýza podobných či konkurenčních řešení
- Kontrolovatelné cíle
  - Termíny
  - Hodnoty některých metrik
- Odhad finančních efektů
  - 15% výkonu při práci u počítače – 3-5% platů

# Uživatelská vrstva




Formát obrazovky závisí na zkušenostech uživatele, módách a také na tom, jak vystihneme psychologii práce. Musíme  jej iterativně zlepšovat

# Analýza a specifikace požadavků

- Poznání (koncového) uživatele, navázání spolupráce, problémy
  - Vedoucí koncového uživatele nevytvoří prostor
  - Vedoucí vývojářů se bojí, že vývojáři budou fungovat jako nápověda pro líné koncové uživatele
  - Co se hlavně sleduje
    - Kvalifikační předpoklady a zkušenosti (kolik je začátečníků a kdo ze spolupracovníků jim může pomoci)
    - Typy koncových uživatelů a jejich obeznámenost s IT
    - Popis úkolů a scénářů činnosti

# Návrh rozhraní

- Paralelní návrh možných řešení, (heuristická evaluace, převzetí dobrých nápadů z existujících sytémů)
- Spoluúčast koncových uživatelů, ti co budou používat. Na začátku lze využít cizí lidi (hlavně u variant pro začátečníky)
- Koordinace a kontrola konsistence návrhu pro různé funkce

# Heuristická evaluace

- Sledování, zda UI splňuje ověřené zásady (jsou v různých dotaznících, až 400 zásad). Základní požadavky na UI jsou:

# Heuristická evaluace

Analyzovat existující systémy (např. Microsoft). Je vhodná podobnost s těmito systémy, připouští-li to funkčnost (srv. grafické systémy). Lidé se UI snáze naučí

Jednoduchý a přirozený dialog v jazyce uživatele,

- Jen to, co je v dané etapě nepostradatelné, podoba s mezilidským dialogem

Minimalizovat nároky na paměť (bezstavovost)

# Heuristická evaluace

Konzistentnost (stejné texty, barvy obrázky)

Zpětná vazba (teploměry, zprávy o postupu práce)

Jasně vymezené exity (ukončení, návrat o několik kroků,  )

Zkratky (horké klávesy), ikony i menu pro časté akce

Kvalitní chybové zprávy (srozumitelný název a odkaz na podrobnosti)

Vylučovat situace vedoucí k chybám

Různé varianty nápovědy

# Heuristická evaluace – výtvarné aspekty

Na obrazovce nemá být mnoho logicky odlišných údajů (do 12), logicky stejné vstupy se počítají jednou, př. DODACÍ LIST.

Nejdůležitější vlevo nahoře, střed

Nehýřit barvami a tvary, zatěžuje zrak a unavuje to mozek

Různé displeje nemají stejné podání barev a to může významně zhoršit čitelnost (nedávat spektrálně blízké barvy do popředí a do pozadí)

Vliv výtvarného řešení (je to hezké)

# Heuristická evaluace – výtvarné aspekty

Teplé barvy vpředu, studené v pozadí

Při zdobnosti snáze uděláme chybný návrh barev

V provozu je důležité, jak je to pracné a jak mne to chrání před chybami

Zdobnost je vítána je-li svým způsobem nápovědou a snižuje-li námahu

# Zpětná vazba

- Reakce systému se pociťuje jako okamžitá,  je-li do desetiny sekundy. U psaní musí být ještě kratší
- Reakce okolo sekundy je tolerována, jde-li o složitější akci
- Reakce přes více sekund vyžaduje indikaci průběhu a nemá být příliš často

# Testování rozhraní

- Realizace prototypů
- Předvedení a zkoušení několika uživateli
  - Někdy se vyplatí prvý test dělat s „brigádníky“ To je zvláště vhodné pro ověřování variant pro začátečníky
  - Ne vždy lze, např. je-li nutná věcná znalost toho, co systém podporuje
- Log průběhu

# Zahájení testu

**Příprava** nástrojů a lidí, místo, nástroje

**Test provádí** obvykle člen skupiny uživatelů za přítomnosti vývojáře. Je třeba brát ohled na velké individuální rozdíly mezi jednotlivými uživateli a také mezi nezkušenými a zkušenými pracovníky. Ověřování UI je proto třeba provádět s větší skupinou pečlivě vybraných budoucích uživatelů. Bývá výhodné provést nejprve zaškolení v ovládání systémových prostředků. Optimální počet testujících je 3 až 6.

# Zahájení testu

Testéři pracují s tou částí UI, se kterou budou skutečně pracovat při provozu systému.

Při organizaci testů je žádoucí navodit atmosféru spolupráce:”Pracujete na tom, aby vám to, co budete používat, sloužilo dobře“.

Uživatelé nemají pracovat ve stresu.

Výsledky testů by měly být anonymní.

Upozornit, že se systém na základě požadavků testérů může změnit

# Zahájení testu

Testování lze do jisté míry provádět na částečně funkčních prototypech. Vždy je však nutné nakonec provádět testy i na oživeném systému. Důvodem je potřeba testovat doby odezvy.

Je obvyklé, že se na základě analýzy výsledků testů UI podstatně mění. Testovat je nutné i nové verze UI.

# Body instruktáže při zahájení testu

účelem testů je testovat systém, nikoliv uživatele;

účelem testů je dosáhnout spokojenosti uživatelů;

je žádoucí, aby se všichni svobodně vyjadřovali;

poněvadž se jedná o testování, může výsledný systém fungovat poněkud jinak;

poprosit, aby o testu testující nehovořili, aby tím ostatní testéry neovlivňovali;

# Body instruktáže při zahájení testu

zdůraznit, že účast na testu je dobrovolná a lze jej kdykoliv ukončit a že výsledky testu jsou anonymní a důvěrné;

uvést, že jsou možné otázky, že je vítáno vyjádření pochybností, a že se má systém v budoucnu používat bez cizí pomoci;

požádat o ”myšlení nahlas“, tj. poprosit, aby testující nahlas komentoval, co dělá a proč;

vítány jsou pochvaly i kritické poznámky

poprosit o pokud možno rychlou práci bez chyb.

# Provedení testu

- Je důležité, aby testování nebylo přerušováno telefony, návštěvami atd.
- Je výhodné, když uživatel provede na počátku rychle a úspěšně nějakou část dialogu, je pak méně nervózní.
- Atmosféra při testování by měla být uvolněná.
- Nedávat najevo, že uživatel je pomalý nebo že dělá chyby.
- Vyloučit kibice, včetně (to především)šéfů testujícího.
- Zastavit testování, vznikne-li pocit, že testér toho má již dost.

# Vyhodnocení testu

- Požádat uživatele o celkový názor, především o to, jak je spokojen.
- Výsledky testů zaznamenat v dohodnuté formě.
- Provést analýzu žurnálu (ten je dodré sledovat i za (zkušebního) provozu
- Výsledky zavést do databáze projektu (pokud existuje).
- Sestavit vlastní hodnocení

# Vyhodnocení testu

- Hodnocení problémů
  - 0 – celkem OK
  - 1 – kosmetická - opravit, bude-li čas
  - 2 - méně závažná – opravit,pokud nebouu závažnější
  - 3 – závažná – opravit co nejdříve, ale není překážkou pro zahájení provozu
  - 4 - katastrofická – systéím nelze provozovat

# Metriky pro hodnocení UI

Doba provedení určitého úkolu.

Počet variant úkolů úspěšně provedených za určitou dobu.

Poměr úspěšně provedených úkolů k počtu chyb.

Doba potřebná pro nápravu chyby.

Počet příkazů použitých během testů.

Počet příkazů, které nebyly vůbec použity.

# Metriky pro hodnocení UI

Frekvence použití nápověd a manuálů.

Počty kritických a pochvalných komentářů uživatele.

Počet případů, kdy byl uživatel frustrován a kdy potěšen.

Rozsah mrtvých dob“, během nichž uživatel nekomunikuje.

Počet případů, kdy uživatel nevěděl jak dál.

# Sledování za provozu

- Log průběhu a jeho analýza
- Dotazníky, sledování reakcí
- Úpravy a jejich efekty

| Metoda | Etapa | Počet testérů | Hlavní výhody | Hlavní nevýhody |
|---|---|---|---|---|
| Heuristické evaluace | Návrh | 0 | Detekuje jednotlivé problémy použitelnosti, lze využít experty. | Není kontakt s uživateli. |
| Měření výkonů. | Testování UI, analýza podobných a konkurenčních produktů. | alespoň 10 | Přesná data. Snadná kritéria porovnání. | Obvykle se nedaří porovnat všechny aspekty. |
| Myšlení nahlas. | Návrh, informativní evaluace. | 3 až 5. | Levné. Zachycuje chyby v pochopení systému v poměrně reálné situaci. | Nepřirozené pro testujícího. Nevhodné pro zkušené. Ti rychleji jednají než mluví. |
| Pozorování. | Analýza a evaluace UI, tvorba scénářů. | alespoň 3 pro každý subsystém | Sleduje skutečné činnosti v reálu. Inspiruje návrh funkcí. | Subjektivní, obtížně kontrolovatelné, časově náročné, často se nenajde dost času na provedení. |
| Dotazniky | Analýza, sledování problémů za provozu. | Většina uživatelů | Anonymní. Lze opakovat, zachytí názory více uživatelů. | Nebezpečí, že důležití uživatelé neodpoví nebo dotazník přesně nepochopí. |
| Interview. | Analýza úkolů. | Několik | Flexibilní, lze jít do hloubky. | Časově náročné, náročné na kvality moderátora, subjektivní. |
| Žurnál (log) | Závěr testování, provoz. | Všichni uživatelé | Běží stále. Výhodné pro vyhodnocování efektivnosti a trendů. | Je nutno realizovat jednoduchý IS pro vyhodnocování. |
| Sledování názorů uživatelů. | Provoz. | Co nejvíce. | Lze sledovat trendy v názorech a potřebách uživatelů. | Obtížně se zajišťuje. „Proč se o to starat, když už to pracuje.“ |