SWE 2

Jaroslav Ráček

Organizační záležitosti

- Přednášky jsou nepovinné.
- Závěrečná zkouška proběhne písemnou formou.

Software?

„*Software* je tvořen celkovým souhrnem počítačových programů, procedur, pravidel a průvodní dokumentace a dat, která náleží k provozu počítačového systému.“

Softwarový produkt je výrobek určený k předání uživateli.

Charakteristiky software

- Je vyvíjen a řešen inženýrskými pracemi, není vyráběn v klasickém slova smyslu.
- Fyzicky se neopotřebuje.
- Obvykle je vyroben na míru, málo produktů je sestaveno z předem existujících komponent.

Charakteristiky software

Generické produkty:

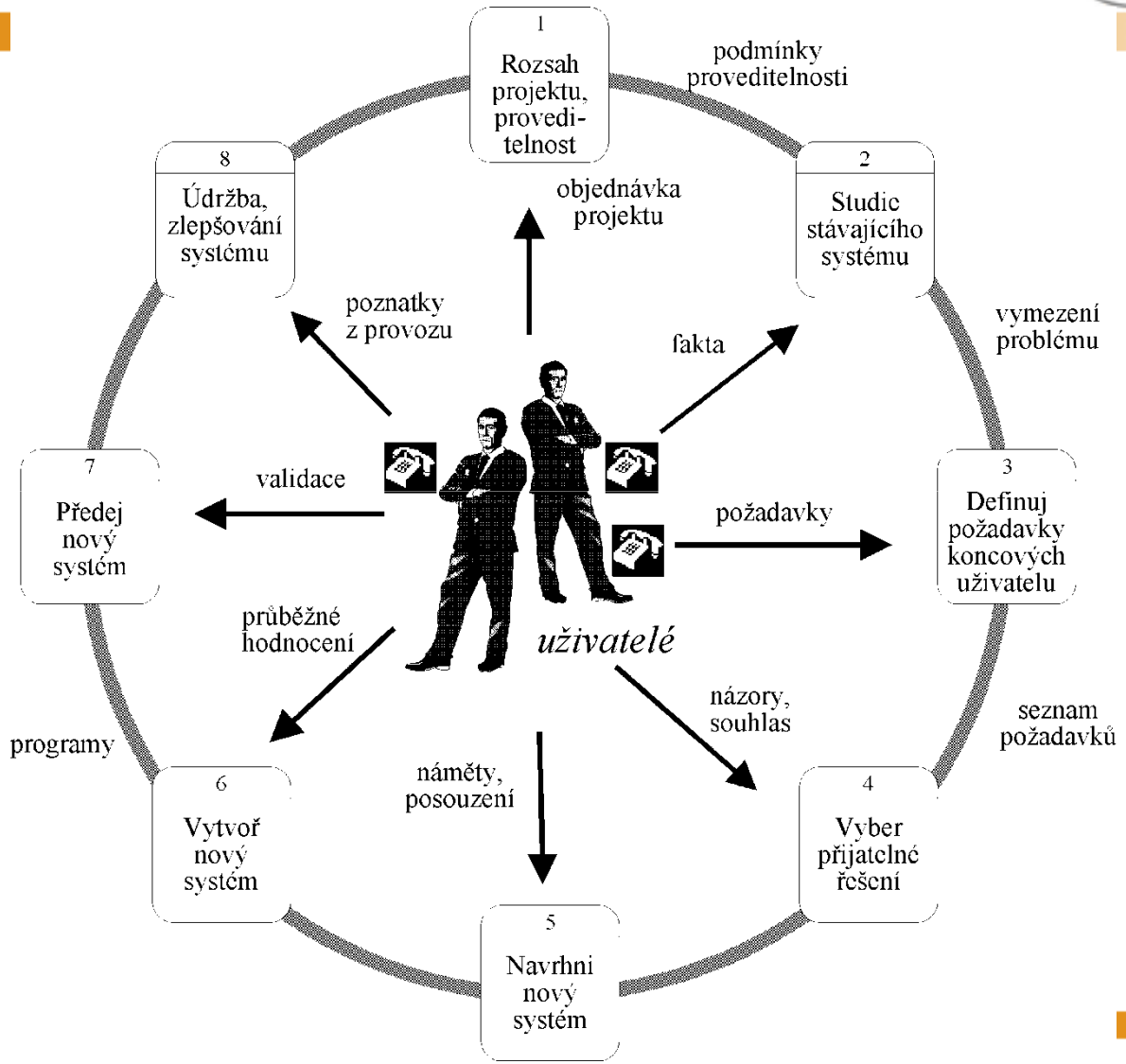
- Samostatné systémy vytvářené výrobní organizací a prodávané na volném trhu libovolnému ze zákazníků, který si může výrobek koupit.

Smluvní, zakázkové, zákaznické produkty:

- Systémy objednané určitým zákazníkem. Software vyvíjí na zakázku smluvně vázaný dodavatel.
- Specifikace generických produktů vytváří (určuje) obchodní oddělení, jedná se o vnitřní záležitost SW producenta.
- Specifikace zakázkového produktu tvoří důležitou součást kontraktu mezi zákazníkem a dodavatelem. Změny musí být odsouhlaseny a pečlivě oceněny.

Uživatel

Interakce řešitelů a uživatelů:



Další charakteristiky

- Problémy s údržbou
- Vysoká cena programů
- Programátorská produktivita

  extrémní individ.odchylky (1:28)
  => odhad ceny, doby ... ?
- Invariance programovacího jazyka

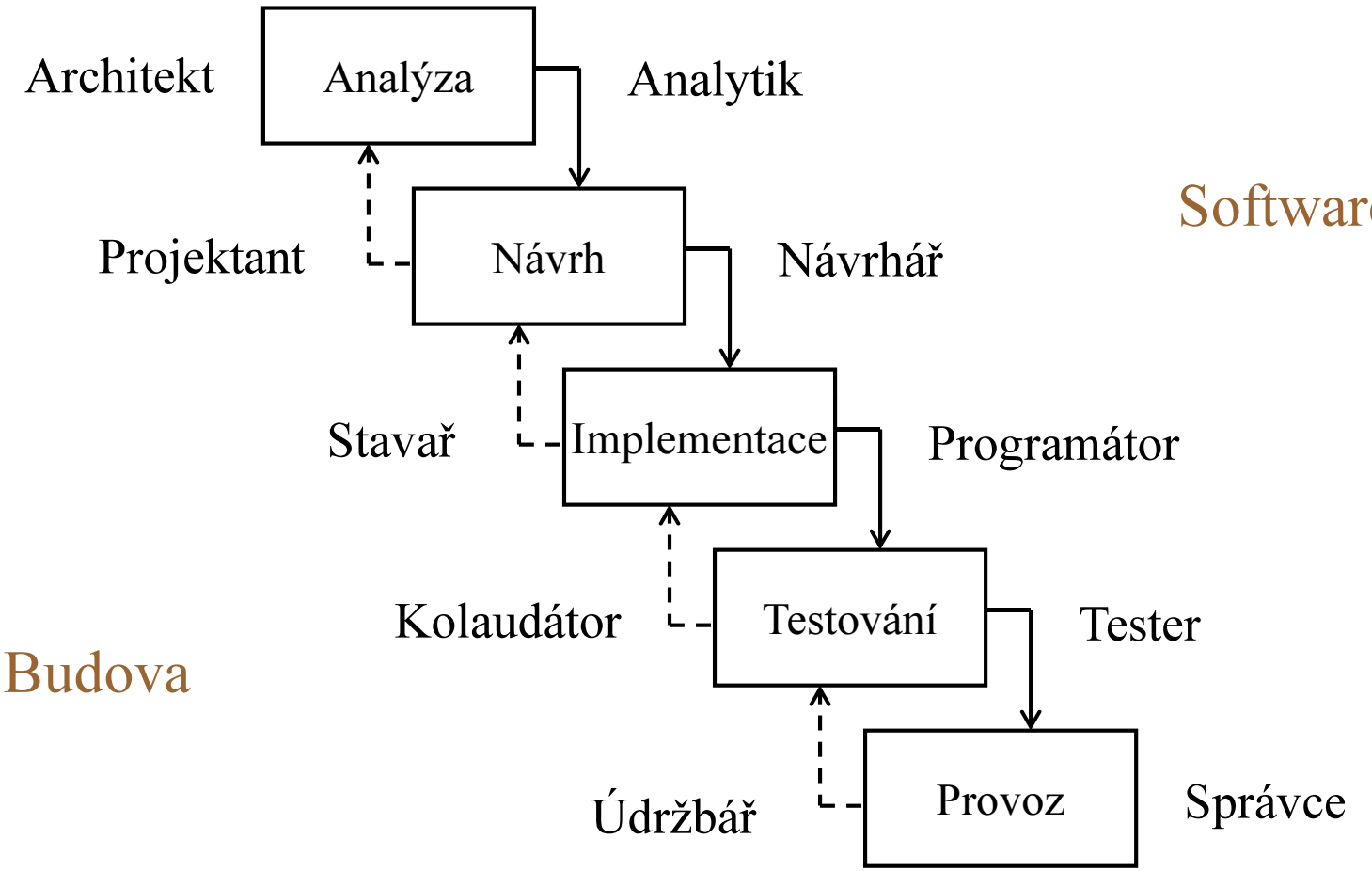
  Složitost aplikace má větší vliv na produktivitu,
  než volba programovacího jazyka
- Ceny odstranění chyb

Je to podobné ve všech inženýrských oborech

Dobře navržený software je jako dobře navržená budova.

Je to inženýrská práce.

Model životního cyklu „vodopád“ - budova nebo software?



Softwarové inženýrství

- Softwarové inženýrství je samostatný inženýrský obor, který řeší systematický přístup k vývoji, provozování, údržbě a nahrazování software.
- Jako obor s certifikátem v USA uznáno v roce 1997.
- Chybí ucelená teorie, základní terminologie není jednotná.
- Praktici používají kolekce technik, které se zdají být funkční.

Dobře řešený software

- Udržovatelnost: Měl by být umožněn vývoj SW podle měnících se potřeb zákazníka.
- Spolehlivost: Mezi atributy SW, na který je spolehnutí, patří spolehlivost, ochrana a bezpečnost. SW by neměl při výpadku systému způsobit fyzické, ani ekonomické škody.
- Efektivita: SW by neměl plýtvat prostředky systému (paměť, čas procesoru a pod.).
- Použitelnost: SW by měl mít přiměřené uživatelské rozhraní a odpovídající dokumentaci.

Kritické faktory SW produktivity

- Složitost

  Lze ji charakterizovat nepřímo, některými viditelnými atributy programu (architektura, počet proměnných).

- Velikost

  Programování v malém vs. programování ve velkém.

- Komunikace

  Jednotlivec, malý tým, velký tým.

- Čas, plán prací
- Neviditelnost SW

Programování *v malém*

- Ověřené techniky.
- Shora-dolů, strukturované kódování, postupné zjemňování, zdokonalování (step-wise refinement).
- Inspekce logiky a kódu.
- Nástroje - překladače, odvšivovače.

Programování *ve velkém*

- Plánovací mechanismy - dělení práce, harmonogram, zdroje.
- Dokumentovaná specifikace.
- Strukturovaný tým.
- Formalizované soubory testů, testovací příklady.
- Formalizované inspekce.

Proces vývoje SW

Vývoj software je proces, při němž jsou:

uživatelovy potřeby
↓ *transformovány na*
požadavky na software
↓ *transformovány na*
návrh
↓ *implementován jako*
kód
↓ *testován, dokumentován a certifikován pro*
operační použití

Základní aktivity při vývoji SW

- Specifikace
  Je třeba definovat funkcionalitu SW a operační omezení.
- Vývoj
  Je třeba vytvořit SW, který splňuje požadavky kladené ve specifikaci.
- Validace
  SW musí být validován („kolaudován“), aby bylo potvrzeno, že řeší právě to, co požaduje uživatel.
- Evoluce
  SW musí být dále rozvíjen, aby vyhověl měnícím se požadavkům zákazníka.

Viditelné znaky výroby SW

- Artefakty
  - Výpisy programů
  - Dokumentace
  - Data
  - Zdrojové soubory
- Procesy
  - Pracovní postupy
  - Uznávaná pravidla (rules-of-thumb)
  - Interakce mezi členy týmu

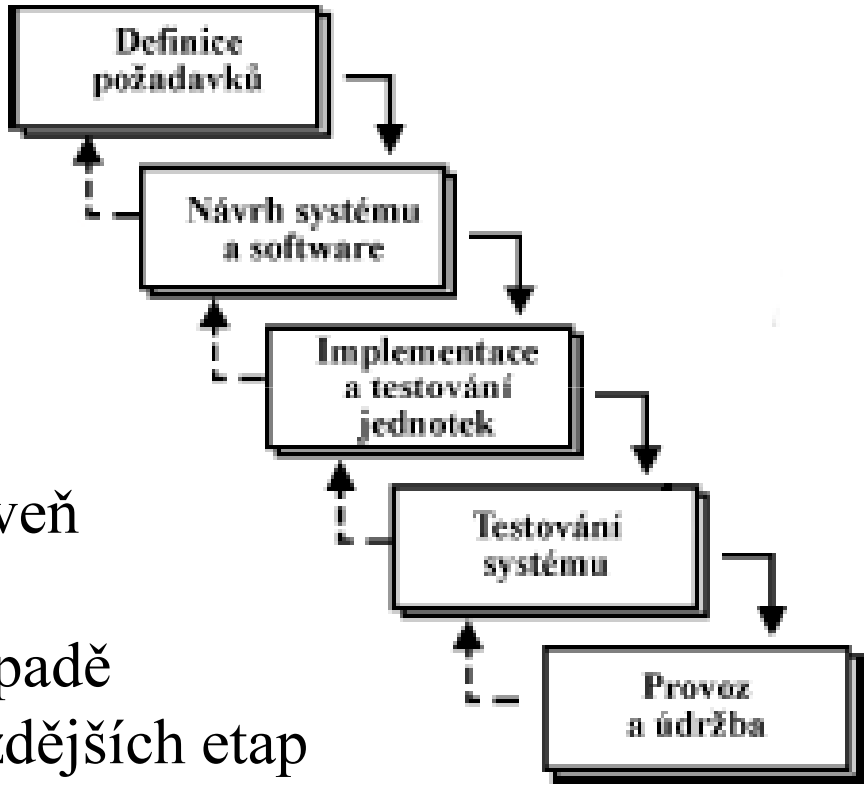
Charakteristiky výrobního procesu SW

- Srozumitelnost
  Je proces explicitně definován a je snadné porozumět definici procesu?
- Viditelnost
  Vyúsťují procesní aktivity ve zřetelné výsledky tak, že postup procesu je viditelný zvenčí?
- Spolehlivost
  Je proces navržen tak, že se lze chybám procesu vyhnout, nebo je zachytit dříve, než zaviní chyby ve výrobku?
- Přijatelnost
  Je definovaný proces přijatelný a použitelný inženýry zodpovídajícími za produkci?

Charakteristiky výrobního procesu SW

- Robustnost
  Může proces pokračovat i v po výskytu neočekávaných problémů?
- Udržovatelnost
  Může se proces vyvíjet tak, aby odrážel měnící se organizační požadavky nebo identifikovaná zlepšení procesu?
- Rychlost
  Jak rychle lze realizovat výrobní proces, který z dané specifikace vytvoří hotový systém předaný zákazníkovi?
- Podporovatelnost
  Do jaké míry lze aktivity procesu podpořit nástroji CASE?

Model životního cyklu „vodopád“



- Integrace systému zároveň s jeho testy
- Problémy s cenou v případě nezdaru v některé z pozdějších etap

Model životního cyklu „vodopád“

Problémy:

- Reálné projekty nedodržují jednotlivé kroky v předepsaném pořadí.
- Uživatel nedokáže v počátečních etapách formulovat požadavky na systém zřetelně a přesně.
- Zákazník musí být trpělivý.
- Pozdní odhalení nedostatků může vážně ohrozit celý projekt.

Manažeři tento model preferují.

Viditelnost životního cyklu „vodopád“

Aktivita	Výstupní dokumenty
Analýza požadavků	Studie proveditelnosti Obrysové požadavky
Definice požadavků	Dokument požadavků
Specifikace systému	Funkční specifikace Plán testování Návrh uživatelského manuálu
Návrh architektury	Specifikace architektury Plán testování systému
Návrh rozhraní	Specifikace rozhraní Plán integračních testů

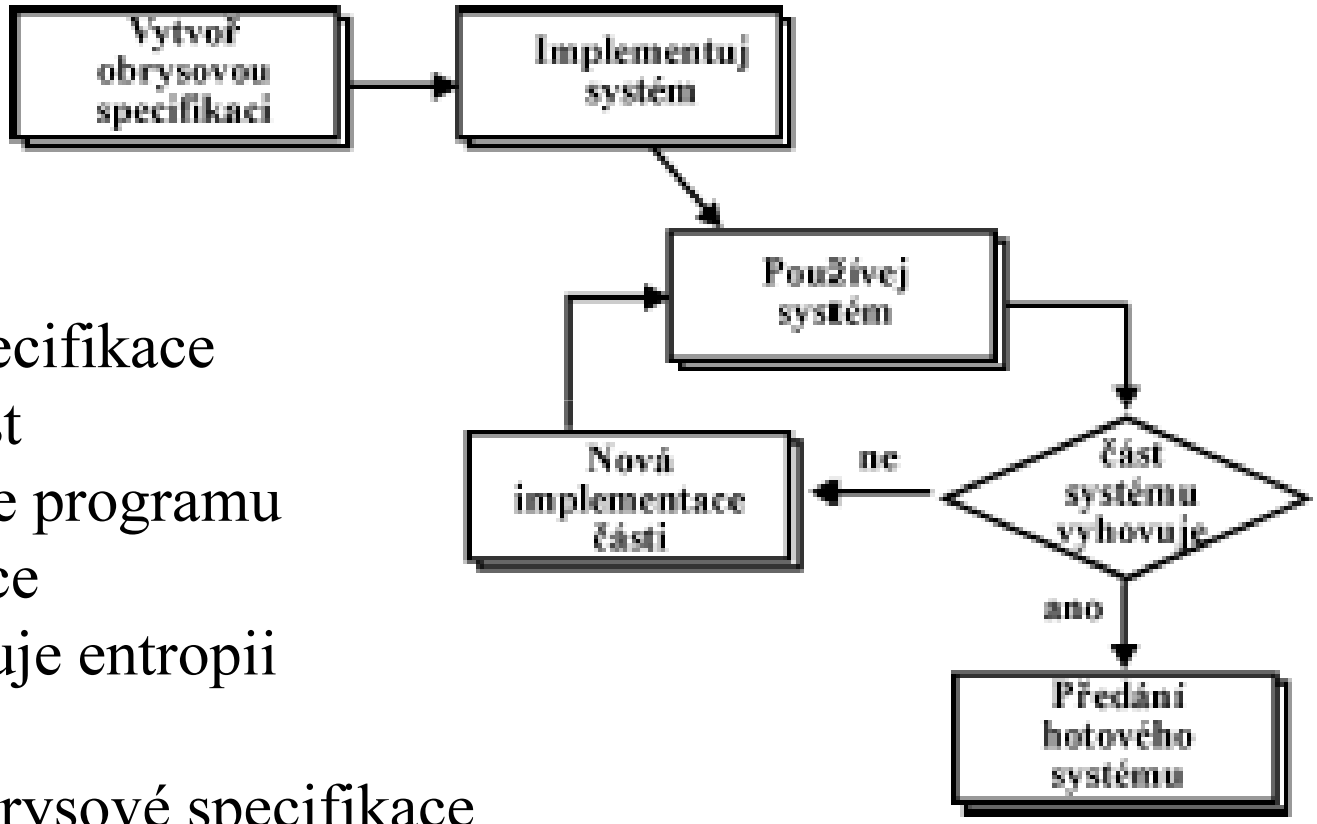
Viditelnost životního cyklu „vodopád“

Aktivita	Výstupní dokumenty
Podrobný návrh	Specifikace návrhu Plán testování jednotek
Kódování	Kód programu
Testování jednotek	Protokol o testování jednotek
Testování modulů	Protokol o testování modulů
Integrační testování	Protokol integračních testů Konečný uživatelský manuál
Testování systému	Protokol testování systému
Přejímací testování	Konečný systém a dokumentace

Vývoj podle obrysové specifikace

- Jsou vyžadováni vysoce talentovaní pracovníci.
  Průměrný tým nelze použít pro tento typ vývoje. Úspěšné produkty byly takto vytvořeny malými týmy vysoce talentovaných.
- Systémy jsou obvykle špatně strukturované.
  Neustálé změny poškozují strukturu systému. Evoluce je obtížná a nákladná.
- Proces není viditelný.
  Manažer potřebuje pravidelné výstupy, aby mohl řídit proces. Při rychlém vývoji není efektivní produkovat dokumenty, které přesně zachycují každou verzi.

Model „inkrementálního“ životního cyklu



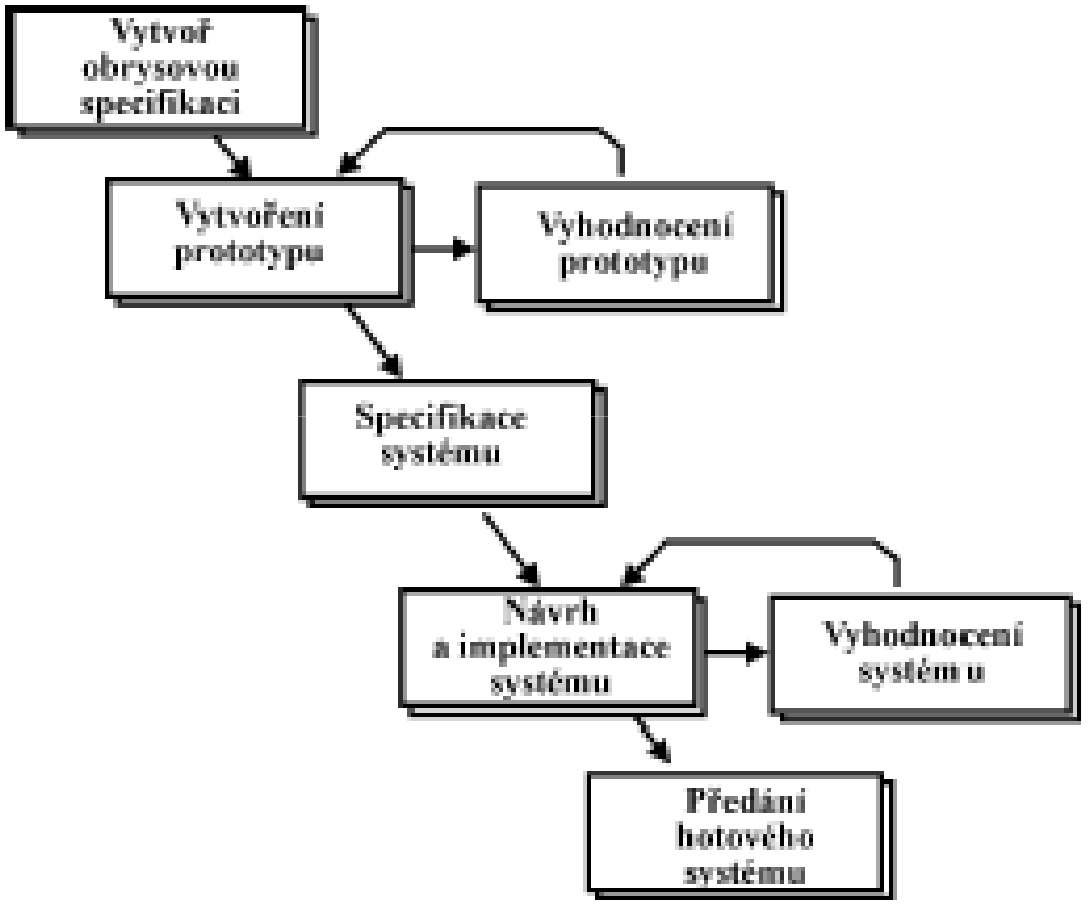
Problémy:

- Obrysová specifikace vs. skutečnost
- Dokumentace programu vs. specifikace
- Údržba zvyšuje entropii

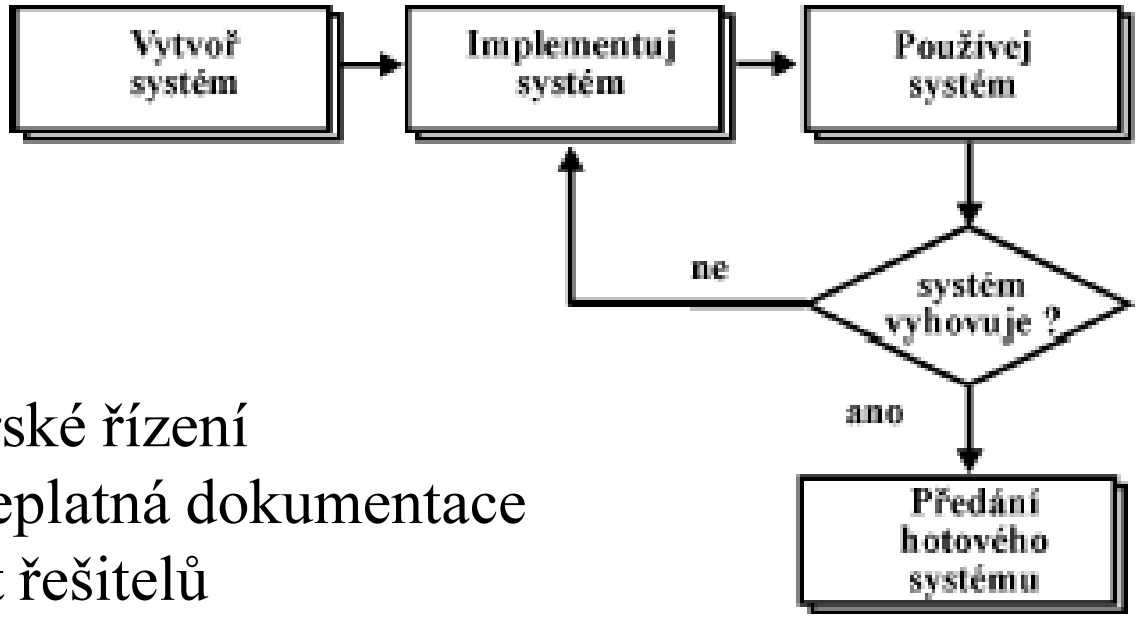
Vývoj podle obrysové specifikace

Model životního cyklu „prototypování“

- Tvorba prototypů probíhá za účelem získávání poznatků.
- Po specifikaci je prototyp zapomenut.



Model životního cyklu „výzkumník“



Problémy:

- Obtížné manažerské řízení
- Neexistující či neplatná dokumentace
- Nenahraditelnost řešitelů

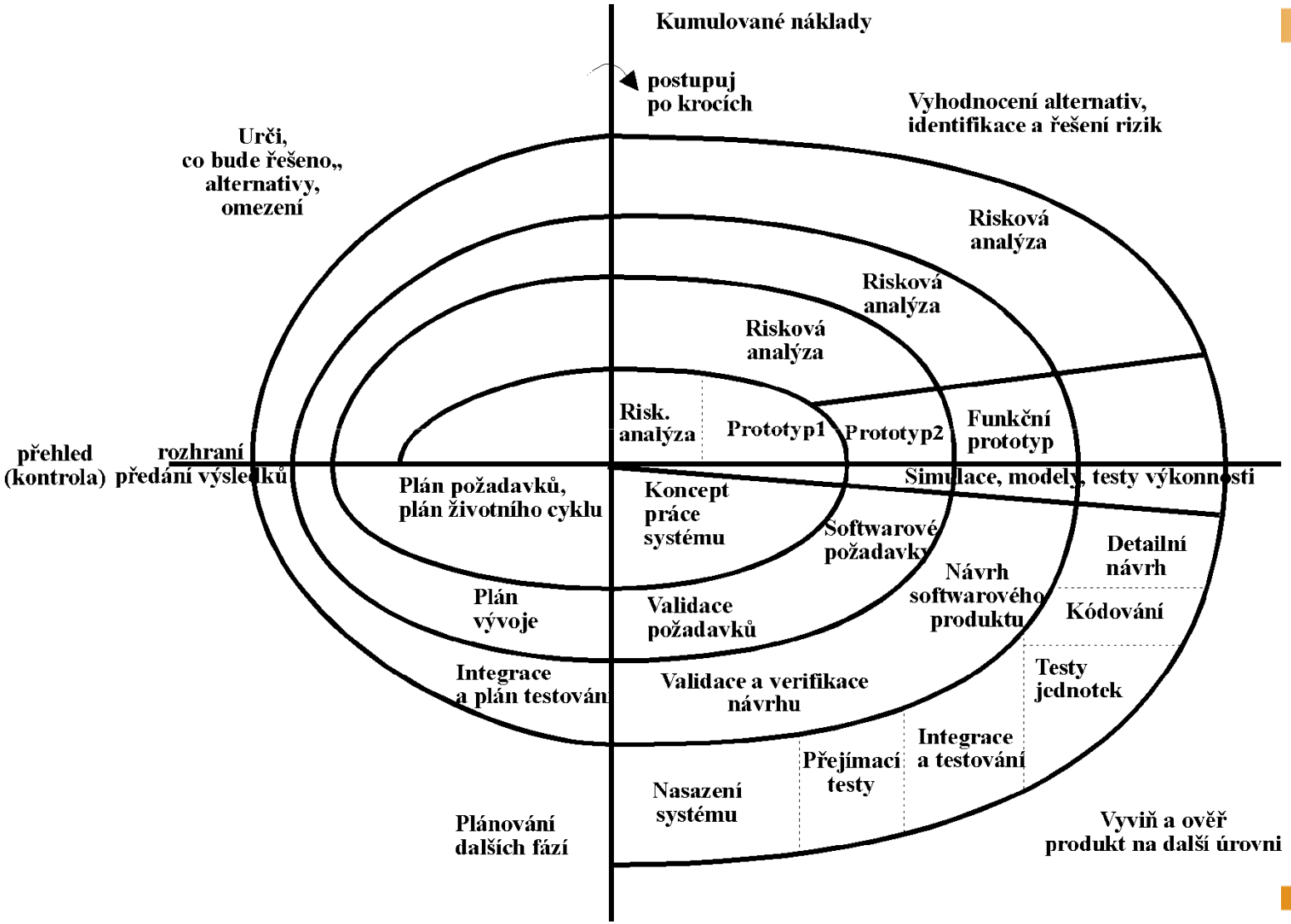
Je to experimentování, u kterého často netušíte, jak dopadne.

Model životního cyklu „výzkumník“

Pro daný typ problému je třeba zvolit vhodný životní cyklus.

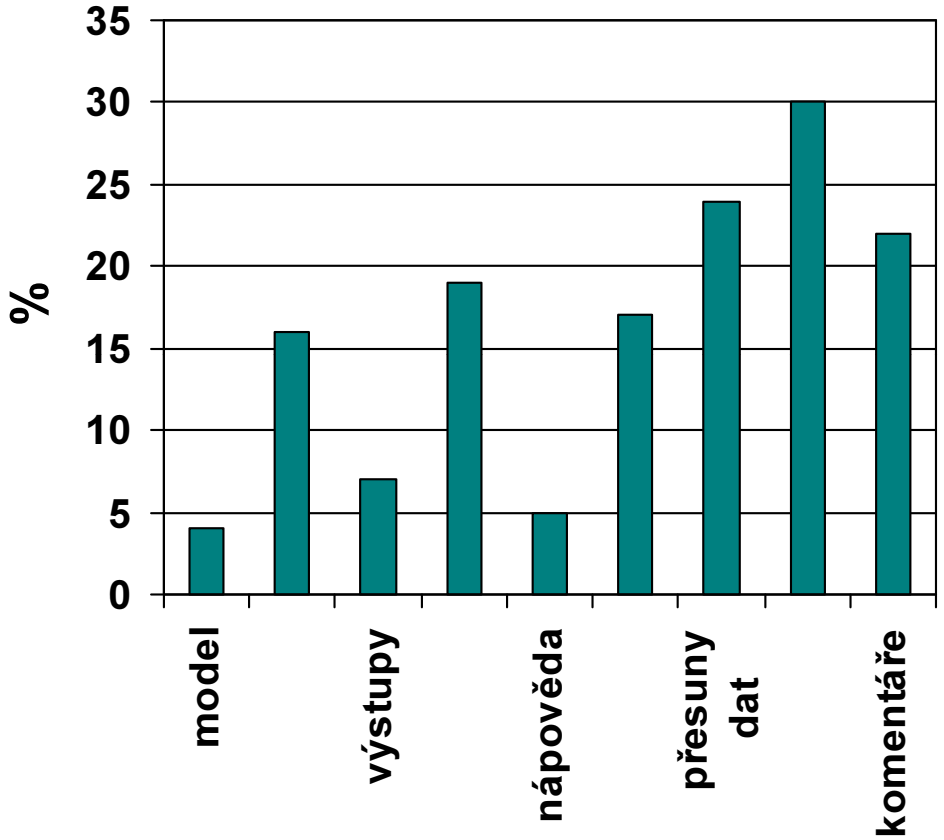
Je model „výzkumník“ vhodný pro spojování dvou mrakodrapů pomocí vysuté lávky?

Spirálový model životního cyklu (Boehm, 1988)



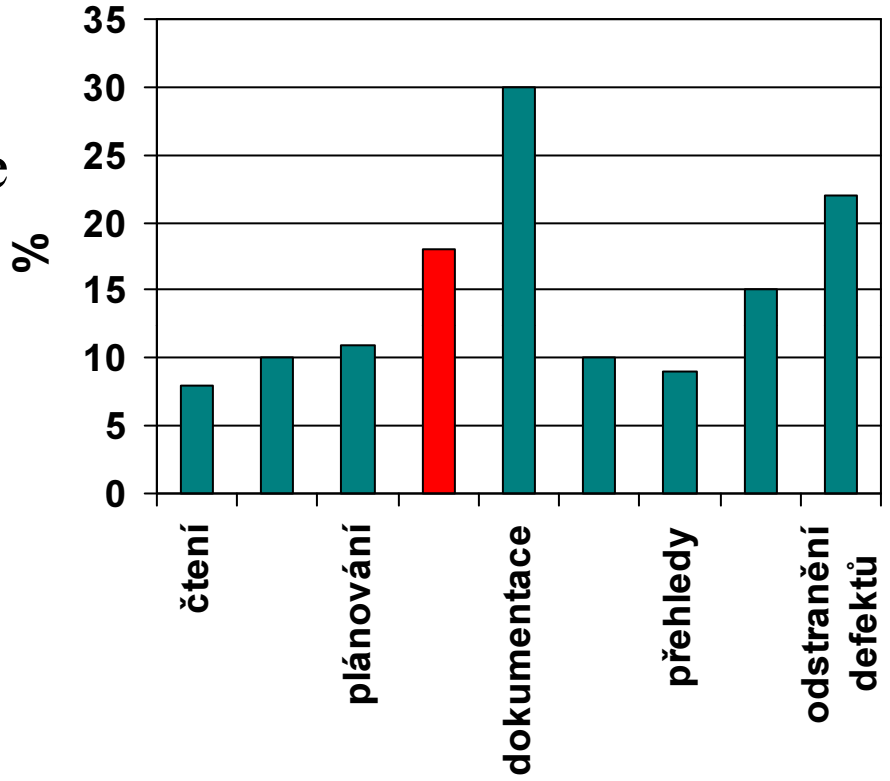
Složení aplikace

- Výpočty na modelu
- Uživatelské vstupy
- Uživatelské výstupy
- Řízení
- Nápověda
- Zpracování chyb
- Přemístění dat (uvnitř)
- Deklarace dat
- Komentáře



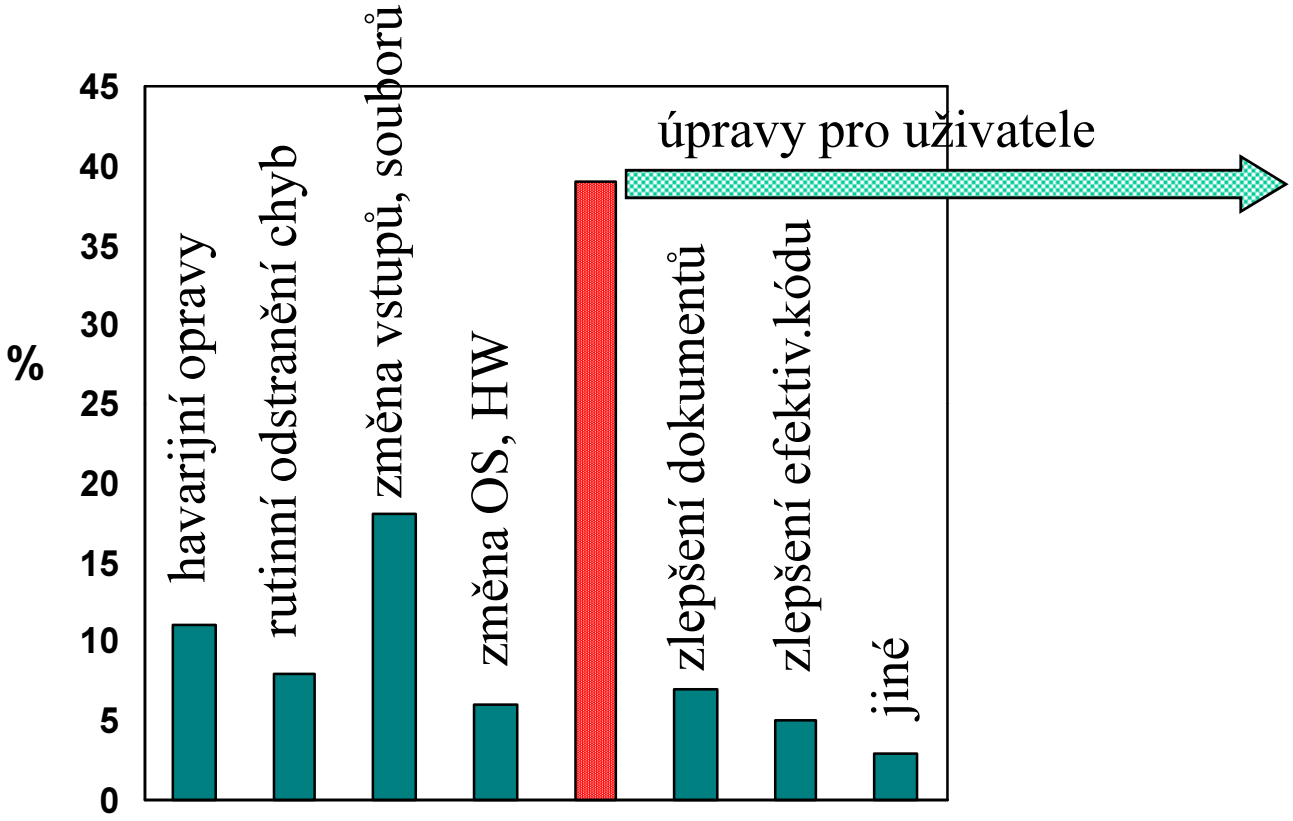
Typické činnosti programátorů

- Čtení poznatků
- Navrhování aplikace, komponent, dokumentace
- Plánování přístupu, úloh, času
- Programování
- Tvorba dokumentace
- Testování
- Přehledy
- Setkání
- Odstranění defektů

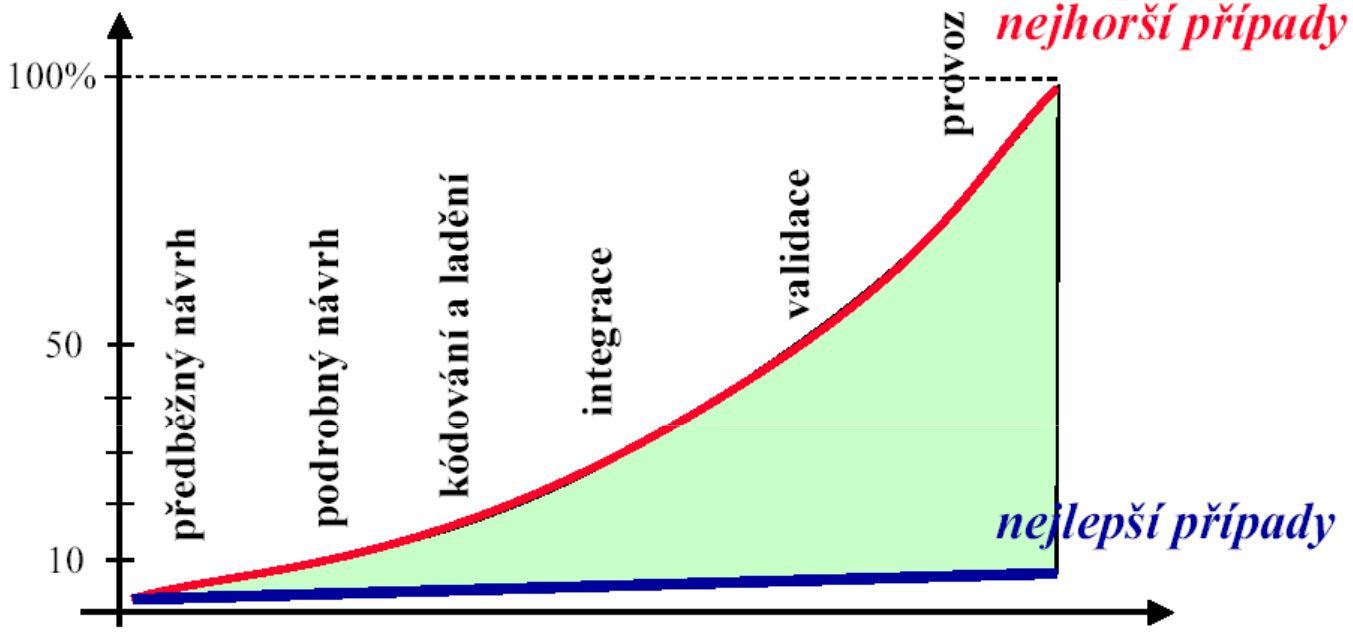


Údržba

Údržba je modifikace SW produktu po předání zákazníkovi za účelem opravy chyb, zvýšení výkonnosti a přizpůsobení měnícímu se okolí.



Relativní cena údržby

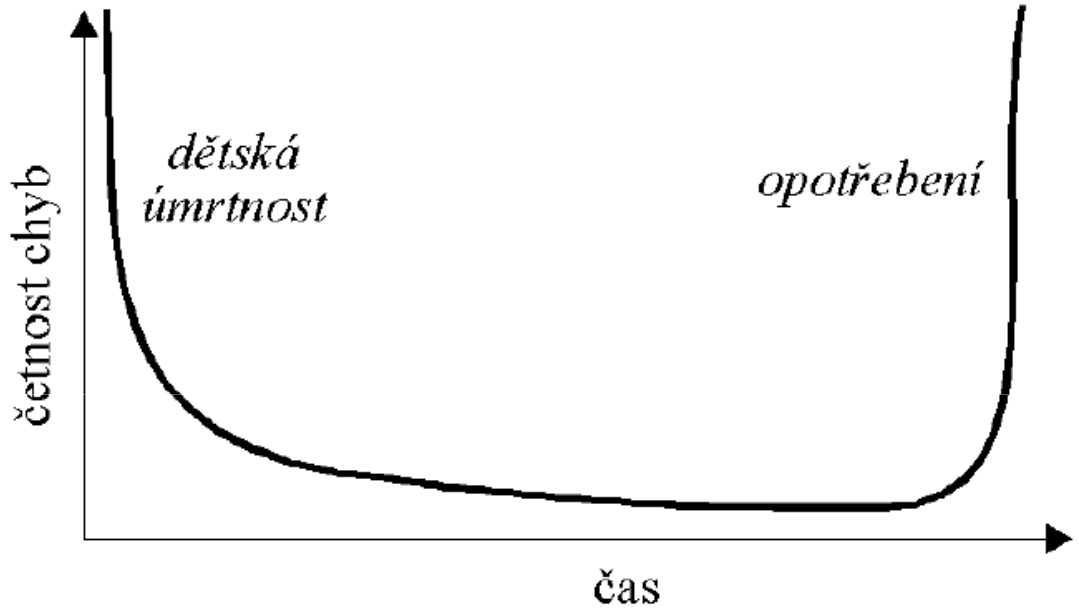


Pokud si dobře rozmyslíte strukturu nového SW a vše uděláte „pořádně“, bude vývoj sice o něco dražší, ale vynaložené náklady se vrátí v období provozu SW systému, kdy děláte jeho údržbu a na přání zákazníka provádíte modifikace.

Hypotézy o chybách

- Čím později v životním cyklu detekujeme chybu, tím nákladnější bude její oprava.
- Mnoho chyb zůstane skryto a je odhaleno až po ukončení fáze, v níž byly udělány.
- V požadavcích je mnoho chyb.
- Chyby v požadavcích jsou především chybná fakta, opomenutí, rozpory a nejednoznačnosti.
- Chyby v požadavcích lze detekovat.

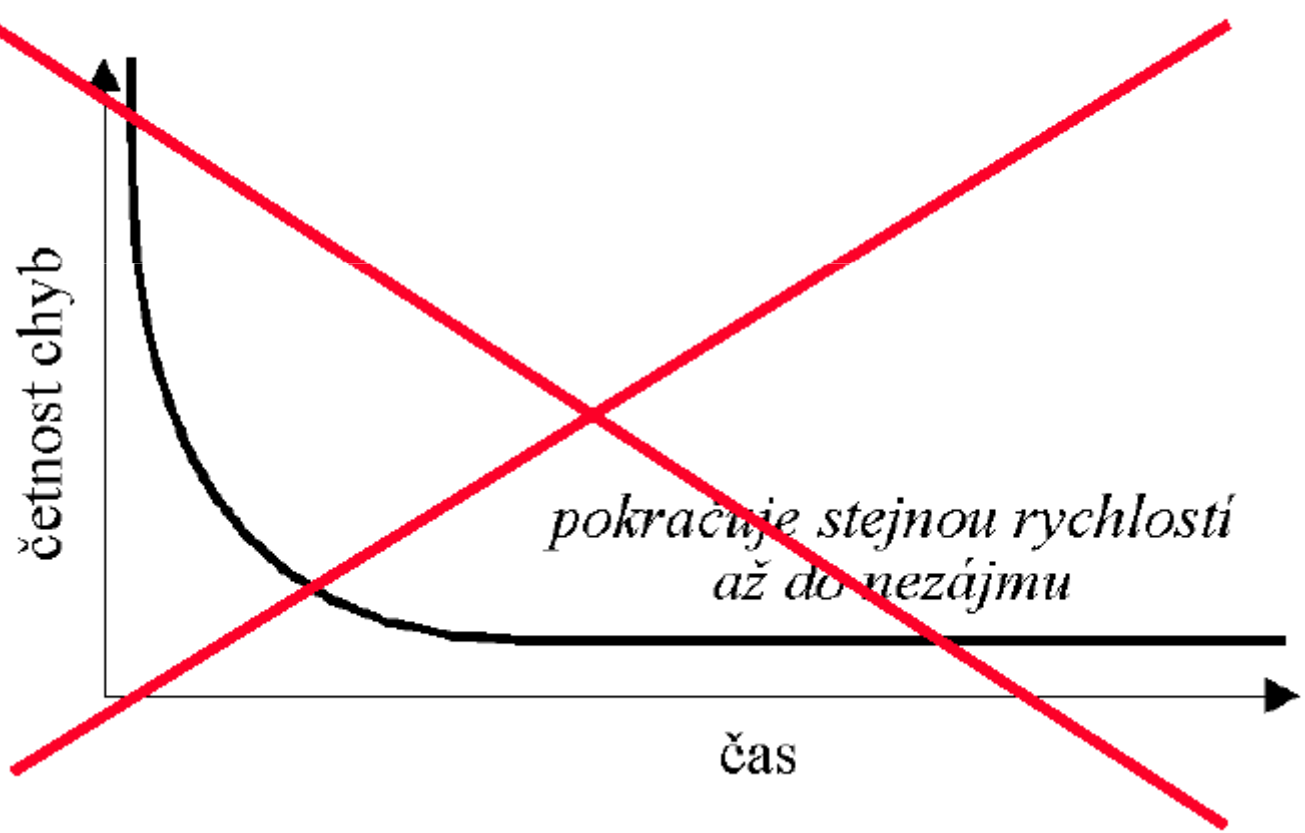
Chyby a opotřebení HW



- Na začátku je nový HW dostatečně výkonný, ale má poměrně dost chyb, které se ale zpravidla podaří časem odstranit.
- Na konci se začne projevovat opotřebení a HW začne zaostávat za svým výkonnějším okolím. Je třeba ho nahradit

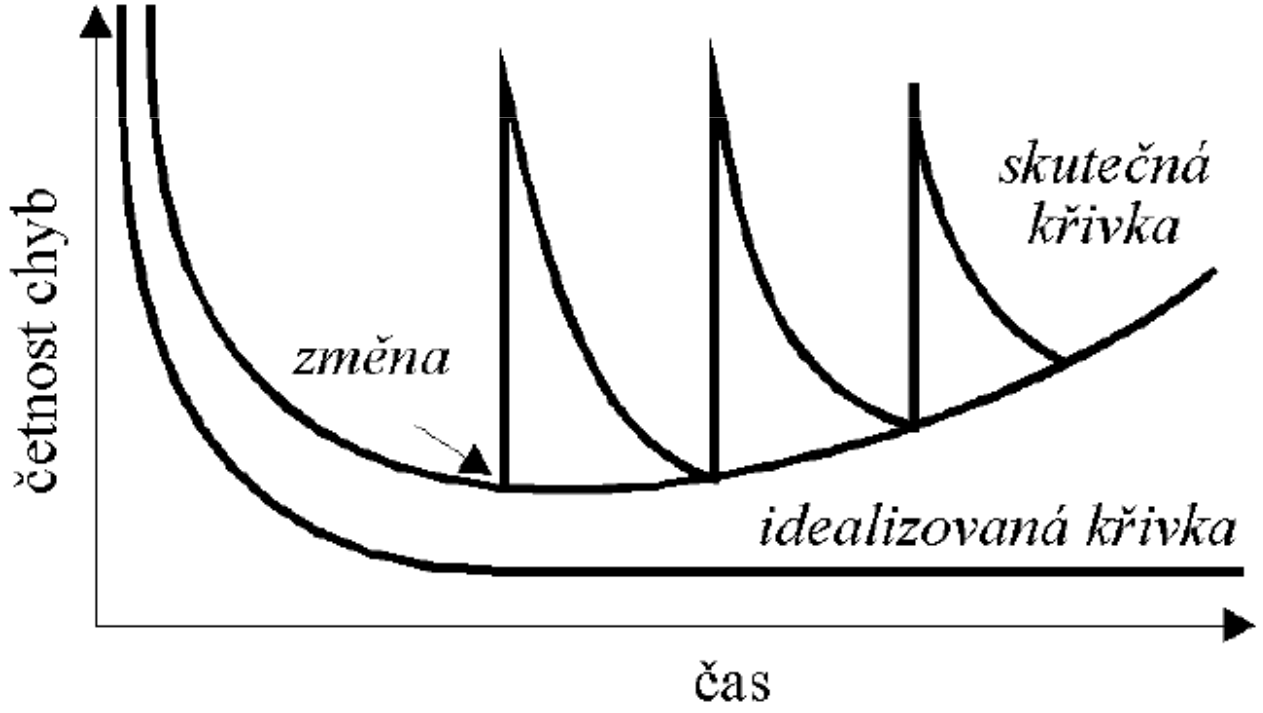
Chyby a opotřebení SW

Je naivní si myslet, že s postupem času odstraníte v SW všechny chyby a zmizí tak veškeré problémy ☺



Chyby a opotřebení SW

Realita je taková, že zákazník požaduje během provozu nové a nové úpravy, což dělá systém složitějším zanáší do systému další chyby. Jednou přijde den, kdy bude lepší to celé naprogramovat znovu. (Lehmanovy zákony)



Lehmanovy „zákony“ (1980, 1985)

Z. trvalé proměny: Systém používaný v reálném prostředí se neustále mění, dokud není levnější systém restrukturalizovat, nebo nahradit zcela novou verzí.

Z. rostoucí složitosti: Při evolučních změnách je program stále méně strukturovaný a vzrůstá jeho vnitřní složitost. Odstranění narůstající složitosti vyžaduje dodatečné úsilí.

Z. vývoje programu: Rychlost změn globálních atributů systému se může jevit v omezeném časovém intervalu jako náhodná. V dlouhodobém pohledu se však jedná o seberegulující proces, který lze statisticky sledovat a předvídat.

Lehmanovy „zákony“ (1980, 1985)

Z. invariantní spotřeby práce: Celkový pokrok při vývoji projektů je statisticky invariantní. Jinak řečeno, rychlost vývoje programu je přibližně konstantní a nekoreluje s vynaloženými prostředky.

Z. omezené velikosti přírůstku: Systém určuje přípustnou velikost přírůstku v nových verzích. Pokud je limita překročena, objeví se závažné problémy týkající se kvality a použitelnosti systému.

Programování v týmu

LOC - Lines of Code - velikost program
E - Effort- úsilí - doba v měsících
PP - Programmer´s produktivity - produktivita programátora
GPP - Group Prog. Productivity - produktivita týmu

$PP = LOC/E$ (řádky kódu za měsíc)

N programátorů => $N.(N-1)/2$ interakcí ~ N^2

λN^2 – úsilí na komunikaci (každý komunikuje s každým)

$GPP = LOC/(E+\lambda N^2)$ skupinová produktivita

$GPP/PP = E/(E+\lambda N^2)$

Brooksův zákon

Přidání řešitelské kapacity u opožděného projektu může zvětšit jeho zpoždění.

(Náklady na začlenění nového pracovníka do týmu jsou zpravidla větší než jeho přínos)