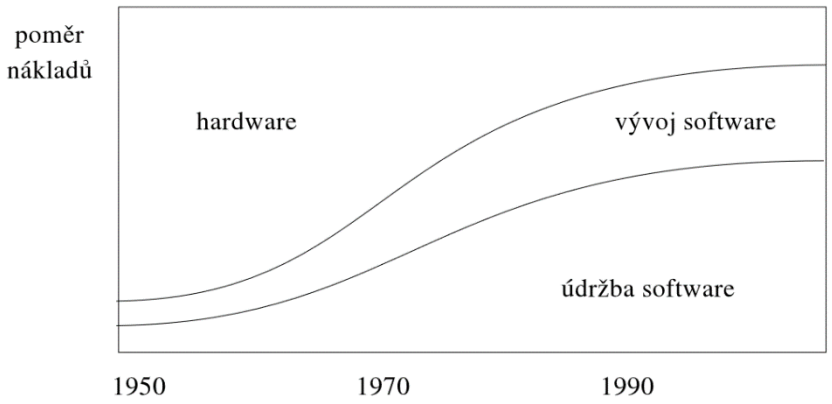
# Úvod do softwarového inženýrství

- SW Inženýrství je:
  - Systematický přístup k vývoji, nasazení a údržbě softwaru
  - Inženýrská disciplína zabývající se praktickými problémy vývoje rozsáhlých softwarových systémů
  - Není to programovaní

- Proč vytváříme software?
  - Zlepšení služeb – informační systémy, . . .
  - Snížení nákladů – řízení výroby, . . .
  - Nemožnost řešení bez použití počítačů – předpověď počasí, . . .
- Je nutné zlepšovat vlastnosti SW, hlavně jeho spolehlivost, bezpečnost a použitelnost.
- Je potřeba zvyšovat produktivitu vývoje SW.




- Počátek SW inženýrství´ - 60. Léta 20. Století
  - Problémy při vývoji větších programů
  - Zavedení pojmů softwarové inženýrství a softwarová krize na konferencích (1968-1969)
  - SW krize se projevovala (a stále projevuje)
    - Neúnosným prodlužováním a prodražováním projektů
    - Nízkou kvalitou výsledných produktů
    - Problematickou údržbou a inovacemi
    - Špatnou produktivitou práce programátorů
    - Řada projektů končila neúspěchem
  - První kroky k metodickému přístupu k programování – strukturované programování

- Průměrný SW projekt tedy v porovnání s původním plánem:
  - Stál o 89 % více,
  - Trval 2,22krát déle a
  - Poskytuje pouze 61 % funkčnosti.
- Průměrný projekt byl tedy téměř 7krát horší, než se původně plánovalo!

| Překročení nákadů o | Projektů |
|---|---|
| **méně než 20 %** | **15,5 %** |
| 21 - 50 % | 31,5 % |
| 51 - 100 % | 29,6 % |
| 101 - 200 % | 10,2 % |
| 201 - 400 % | 8,8 % |
| více než 400 % | 4,4 % |

| Výsledná funkčnost | Projektů |
|---|---|
| méně než 25 % | 4,6 % |
| 25 - 49 % | 27,2 % |
| 50 - 74 % | 21,8 % |
| 75 - 99 % | 39,1 % |
| **100 %** | **7,3 %** |

| Překročení času o | Projektů |
|---|---|
| **méně než 20 %** | **13,9 %** |
| 21 - 50 % | 18,3 % |
| 51 - 100 % | 20,0 % |
| 101 - 200 % | 35,5 % |
| 201 - 400 % | 11,2 % |
| více než 400 % | 1,1 % |

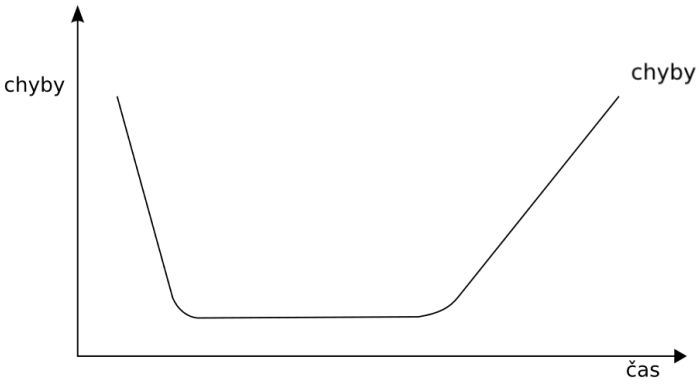
- Problémy při vývoji softwaru
  - Podstatné, vnitřní, nevyhnutelné problémy:
    - Složitost – velice těžce jí odhadneme, žádné dvě části nejsou stejné; složitost je zdrojem dalších problémů jako např. Komunikace v týmech; je náročné pochopit všechny možné stavy systému; problémy s úpravami a rozšířeními, . . .
    - Přizpůsobivost – když se něco změní, měl by se přizpůsobit software a ne naopak.
    - Nestálost – mění se okolí a mění se i software (nejde o nahrazení novým); přibývají požadavky na úspěšně používaný software; software přežívá hardwarové prostředky.
    - Neviditelnost – neexistuje přijatelný způsob reprezentace softwarového výrobku, který by pokryl všechny aspekty; dokonce ani nejsme schopni určit, co v dané reprezentaci chybí.

    - Syndrom 90 % hotovo: Při posuzování hotové části se nevychází z hotového, ale z odpracovaného (např. Podle plánu)

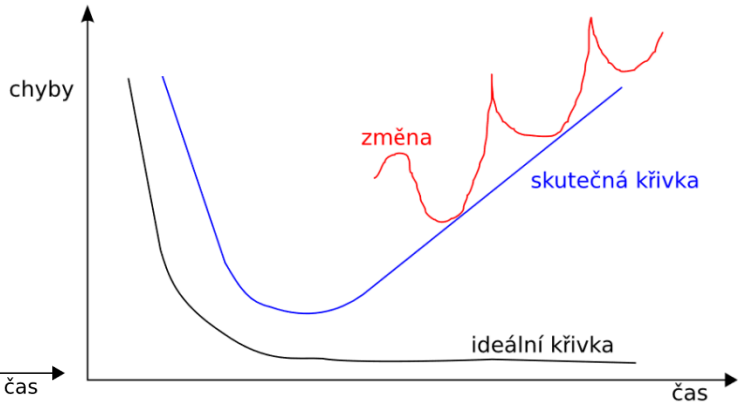
  - Problémy, které se nemusí projevit vždy:
    - Práce v týmu
      - Problémy s organizací práce na velkých softwarových projektech
      - Problémy s plánováním procesu tvorby softwaru
      - Komunikační problémy jsou jedním z hlavních zdrojů chyb v programech.
      - Extrémní odchylky v produktivitě mezi jednotlivými programátory, až 1:20
    - Nízká znovupoužitelnost při tvorbě softwaru
      - V procesu tvorby softwaru je málo standardů a většinou se software tvoří od začátku. S každým programem se vymýšlí už vymyšlené.
      - Málo produktů se sestavuje z už existujících součástí.
    - Problém míry
      - Metody použitelné na řešení malých problémů se nedají přizpůsobit na řešení velkých (složitých) problémů.
    - Tvorba dokumentace
      - Tvorba dokumentace je podobná tvorbě vlastního programu.
      - Enormní rozsah dokumentace co do kvantity i rozmanitosti
      - Např. Ve velkých vojenských softwarových projektech připadalo 400 anglických slov na každý příkaz v programovacím jazyce Ada.
      - Problémy s udržováním aktuálnosti dokumentace vzhledem ke změnám softwaru
      - Problémy s konzistencí a úplností dokumentace

- Náchylnost softwaru k chybám
  - Hodně chyb se projeví až při provozu (a ne při vývoji).
  - Odstraňování chyb vede k návratu v etapách vývoje softwaru.
- Způsob stárnutí softwaru
  - Software se fyzicky neopotřebuje. ALE: Přidávání nových funkcí ve spojení s častými opravami chyb vede k postupné degradaci struktury a k snižování spolehlivosti softwarových systémů

Typická chybová křivka hardwaru




Typická chybová křivka softwaru




- Specifikace požadavků
  - Problematická komunikace s uživatelem
  - Nejasná a neúplná formulace požadavků spojená s neucelenou představou uživatele o výsledném softwarovém systému
  - Nejednoznačnost spojená s častou specifikací požadavků v přirozeném jazyce (Termíny někdy každý pochopí jinak)

**Tvorba softwaru je tvůrčí proces, software nelze vyrábět.**

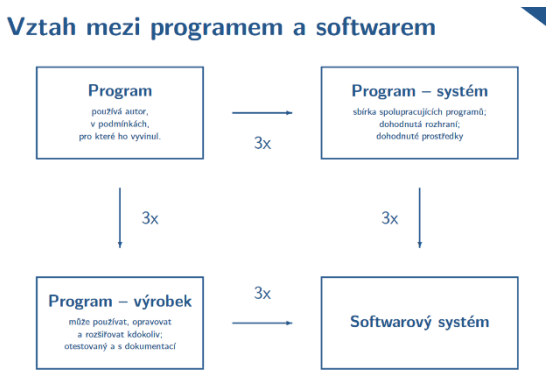
- **Postřehy Freda Brookse**
  - Přidáním dalších pracovníků do zpožděného projektu se tento projekt ještě více zpozdí.
  - Napsání překladače Algolu zabere 6 měsíců nezávisle na tom, kolik ho vytváří programátorů.
  - Efekt (syndrom) druhého systému – při návrhu druhé verze systému hrozí rizika:
    - Příliš složitý a neefektivní systém
      - **Systém není dokonalý, když k němu nelze nic přidat, ale tehdy, když z něho nelze nic odstranit.**
    - Nepoužití nových technologií

- **Rozvoj SW inženýrství**
  - **Výzkum programovacích praktik**
    - Uvědomění si lidského faktoru, práce v týmu
    - Podpora řízení tvorby SW
    - Modulární programování
    - Návrhové vzory

- **Výzkum metodik**
  - Vnímání životního cyklu vývoje SW
  - Strukturované metody, datově a procesně orientované metody, objektově orientované metody, agilní metodiky, . . .
  - Výzkum modelovacích jazyků (dnes UML)
- **Zabezpečení kvality**
  - Systematické testování, formální ověřování
- **Metody návrhu založené na modelech**
  - Transformace modelů do programu

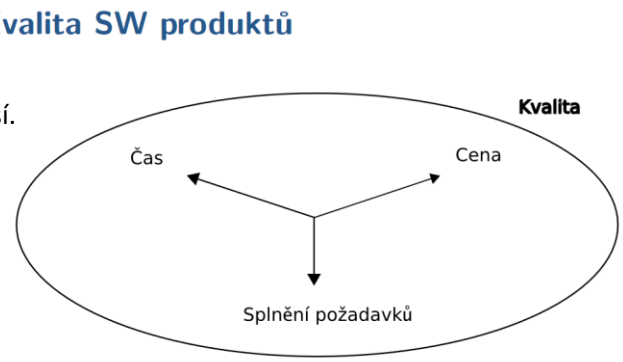
- **Metodiky vývoje softwaru**
  - **Metodiky**
    - **Disciplinovaný proces** nad vývojem softwaru **s cílem zajistit tento vývoj** více **predikovatelný** a **efektivnější**
    - Věnují se různým aspektům, které ovlivňují vývoj softwarového produktu, včetně samotného procesu tvorby softwaru
    - Zahrnují proces vývoje, nástroje, způsoby využití, plánování, . . .
  - **Pozor na terminologii!**
    - **Metoda** – postup pro dosažení určitého cíle
    - **Metodika** – souhrn doporučených praktik a postupů
    - **Metodologie** – nauka o metodách, jejich tvorbě a použití
  - **Ale! Metodika vývoje softwaru = Software Development Methodology**

- **Softwarový produkt**
  - **Program**
    - Funkční část produktu
  - **Softwarový produkt**
    - Sbírka počítačových programů, procedur, pravidel a s nimi spojená dokumentace
    - Zahrnuje např.: požadavky, specifikace, popisy návrhu, zdrojové texty, testovací data, příručky,...
  - **Aktéři ve vývoji softwarového produktu (softwaru)**
    - **Zákazník –** sponzoruje vývoj SW, specifikuje požadavky na SW
    - **Dodavatel –** vyvíjí systém, má závazky vůči zákazníkovi, komunikuje s uživatelem (testování, ...)
    - **Uživatel –** testuje a používá systém, upřesňuje požadavky na SW




- **Typy softwarových produktů**
  - **Generické**
    - Software se **prodává libovolnému zájemci** (krabicový software).

        - Musí být velice důkladně otestován, protože opravy chyb jsou vzhledem k velkému rozšíření drahé.
    - **Zákaznické (na objednávku)**
        - Software se **vytváří na základě požadavků** pro konkrétního zákazníka.
        - Většinou pro specializované aplikace, pro které vhodný generický software neexistuje.
        - Cena zákaznického softwaru je výrazně vyšší.
        - Dvě možnosti jeho tvorby:
            - **Zadáním zakázky SW firmě**
            - **V rámci vlastní firmy**




- **Proces vývoje softwaru**
    - Proces, ve kterém:
    - Se **potřeby uživatele transformují na požadavky** na SW,
    - **Požadavky** na SW **se transformují na návrh**,
    - **Návrh se implementuje,**
    - **Implementace se testuje**
    - A nakonec **předá uživateli.**

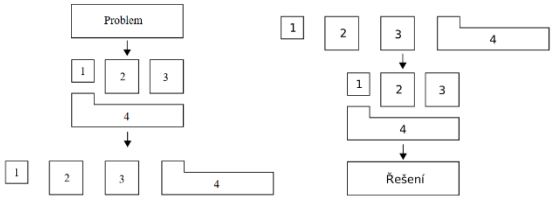
- **SW proces definuje: kdo, dělá co, a kdy**
    - ⇒ jak dosáhnout požadovaného cíle

- **Životní cyklus softwaru**
    - Rozděluje proces vývoje softwaru na za sebou jdoucí období
    - Pro každé období stanovuje cíl
    - **Období** = etapa životního cyklu softwaru

- **Činnosti spojené s vývojem softwaru**
    - **Analýza a specifikace požadavků (8 %),**
    - **Architektonický a podrobný návrh (7 %),**
    - **Implementace (12 %),**
    - **Integrace a testování (6 %),**
    - **Provoz a údržba (67 %).**
    - Úsilí věnované pečlivé analýze a návrhu se vrátí úsporou nákladů později

    - **Analýza a specifikace požadavků**
        - **Získávání, analýza, definování a specifikace požadavků** ⇒ transformace **neformálních požadavků** uživatele **do strukturovaného popisu požadavků**,
        - Cílem je **identifikovat požadavky uživatele**, ne návrh, jak je realizovat,
        - Provedení **studie vhodnosti, identifikace a analýza rizik**,
        - **Plánování akceptačního testování.**

- **Dekompozice složitých problémů:**
  - **Rozdělení** (dekompozice) **složitějšího problému na jednodušší** (lehčí zvládnutí problému)
  - Rozhraní podsystémů
  - **Přináší**:
    - Lépe zvládnutelné podsystémy
    - Soustředění pozornosti na jeden podsystém
    - Prezentovatelnost dílčího problému bez rušivých vlivů
    - Podsystémy se mohou vyvíjet **nezávisle**
    - **Skutečně velké systémy se bez dekompozice nedají zvládnout**




  - **Zvýšená pozornost na:**
    - Koordinace tvorby rozhraní
    - Integrace a testování podsystémů

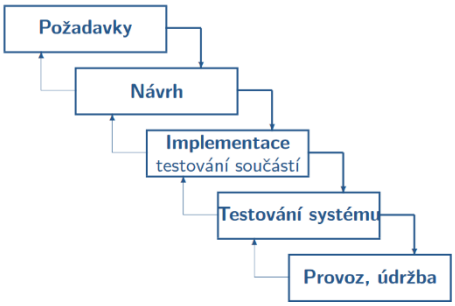
- **Architektonický návrh**
  - Ujasnění koncepce systému
  - Dekompozice systému
  - Definování vztahů mezi částmi systému,
  - Specifikace funkcionality a ohraničení podsystémů,
  - Plánování testování systému,
  - Plánování nasazení systému do provozu, dohoda o postupu nasazování podsystémů, dohoda o plánu zaškolování uživatelů
- **Podrobný návrh**
  - Podrobná specifikace softwarových součástí
  - Specifikace algoritmů realizujících požadované funkce,
  - Specifikace rozhraní pro jednotlivé součásti,
  - Specifikace logické a fyzické struktury údajů které zpracovává příslušná součást,
  - Specifikace způsobu ošetřování chybových a neočekávaných stavů,
  - Plán prací při implementaci součásti,
  - Plán testování součásti, návrh testovacích dat,
  - Specifikace požadavků na lidské zdroje (odhad trvání a nákladů projektu)

- **Implementace a testování součástí**
  - Programová realizace softwarových součástí,
  - Vypracování dokumentace k součástem,
  - Testování implementovaných součástí,
  - Začátek školení budoucích uživatelů
- **Integrace a testování systému**
  - Spojení součástí do podsystémů,
  - Testovaní podsystémů,
  - Integrace podsystémů do celého systému,
  - Testování podsystémů a celého systému
  - Oprava nalezených chyb, návraty k etapě implementace.

- **Akceptační testování a instalace**
  - Testování systému uživatelem
  - Operace přebírání SW produktu
  - Školení používání systému, nasazení systému

- **Provoz a údržba**
  - Zabezpečení provozu softwaru,
  - Řešení problémů s nasazením softwaru,
  - Řešení problémů s používáním softwaru,
  - Opravy, rozšiřování, přizpůsobování softwaru podle požadavků okolí

- **Model životního cyklu softwaru:**
  - **Model životního cyklu:**
    - Definuje **etapy vývoje softwaru** a jejich **časovou následnost,**
    - Pro každou etapu definuje **nutné činnosti**
    - Pro každou etapu definuje její **vstupy a výstupy**
  - **Další vlastnosti**:
    - **Nedefinuje délku** trvání kroků **a** jejich **rozsah**,
    - Každá **etapa vytváří reálné výstupy**,
    - **Správnost** každé **etapy lze vyhodnotit**
  - **Rozdíly** v modelech jsou zejména v **definovaní etap a jejich posloupnosti**.
  - **Vodopádový model životního cyklu softwaru**




    - Životní cyklus jde postupně od první etapy až do poslední
    - Následující etapa začne až po ukončení předcházející
    - Možnost návratu k předchozí etapě
    - **Vlastnosti:**
      - Lineární (sekvenční) model, intenzivně používán v 70. letech
      - Cílem bylo zavést do vývoje řád, umožňující řešit náročnější problémy
      - Dekompozice, kontrola výstupů etap -> snížení počtu chyb
      - Uživatel se účastní pouze při definováni požadavků a zavádění
    - **Výhody:**
      - Jednoduché na řízení
      - Při stálých požadavcích: nejlepší struktura výsledného produktu
    - **Nevýhody:**
      - Zákazník není schopen předem přesně stanovit všechny požadavky, to zapříčiní že se v modelu musí vrátit
      - Při změnách požadavků dlouhá doba realizace
      - Zákazník vidí spustitelnou verzi až v závěrečných fázích projektu –> **odhalení nedostatků ve specifikaci požadavků příliš pozdě** (validace)