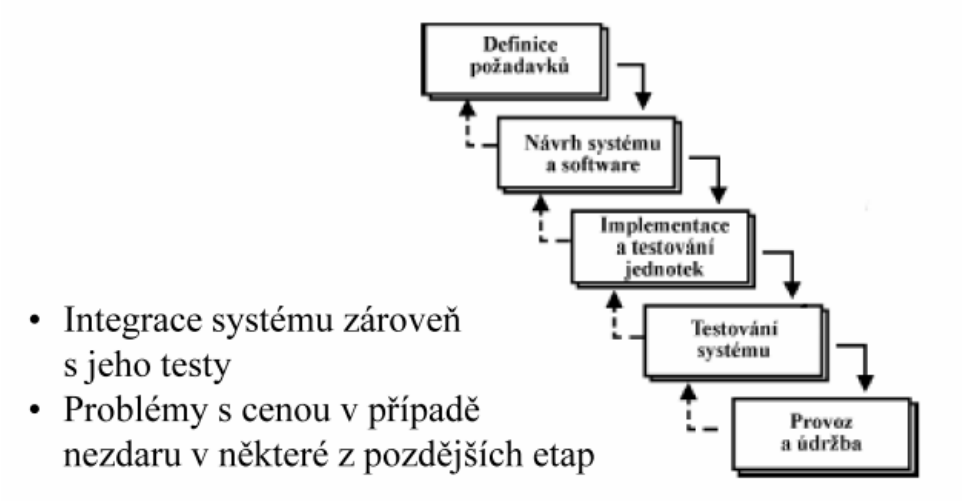
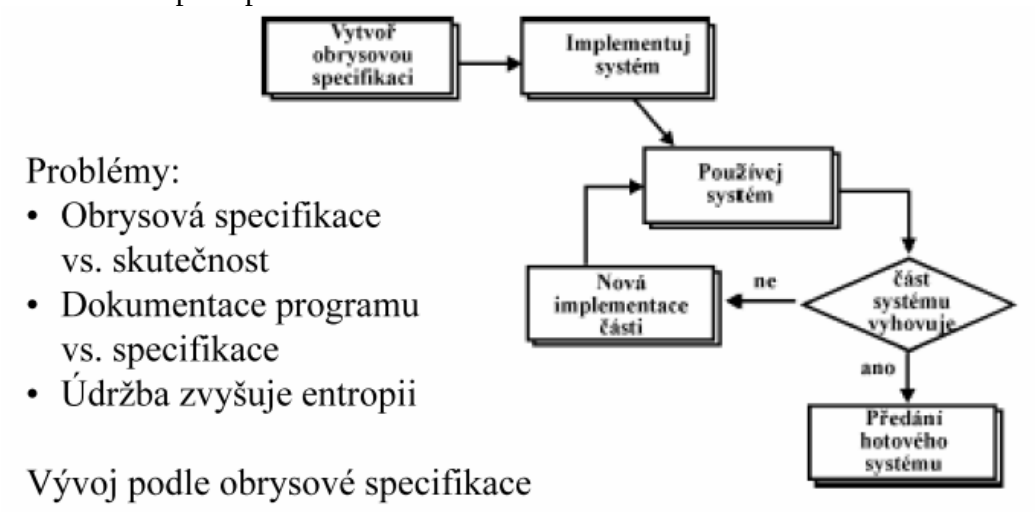
1. Software

- software
  - souhrn počítačových programů, procedur, pravidel a průvodní dokumentace a dat, který náleží k provozu počítačového systému
  - vyvíjen a řešen inženýrskými pracemi
  - fyzicky se neopotřebuje
  - obvykle vyroben na míru
  - problémy s údržbou
  - vysoká cena
  - programátorská produktivita
  - invariance programovacího jazyka
- softwarový produkt - výrobek určený k předání uživateli
- typy produktů
  - **generické produkty** - systémy prodávané na volném trhu, které si může koupit libovolný zákazník
  - **smluvní, zakázkové produkty** - systémy objednané určitým zákazníkem, systém je vytvářen na základě specifikací od zákazníka
- dobře řešený software
  - **udržovatelnost** – lze měnit podle měnících se potřeb zákazníka
  - **spolehlivost** – nezpůsobuje fyzické ani ekonomické škody
  - **efektivita** – neplýtvá prostředky systému
  - **použitelnost** – uživatelské rozhraní a dokumentace
- kritické faktory SW produktivity
  - složitost
  - velikost
  - komunikace
  - čas, plán prací
  - neviditelnost SW
- programování v malém
  - ověřené techniky
  - vývoj shora dolů
  - inspekce logiky a kódu
- programování ve velkém
  - plánovací mechanismy
  - dokumentovaná specifikace
  - strukturovaný tým
  - formalizované testy a inspekce
- základní aktivity při vývoji SW
  - **specifikace** – definování funkcionality a omezení
  - **vývoj** – implementace podle požadavků
  - **validace** – kontrola, zda SW plní specifikace
  - **evoluce** – další rozvíjení SW podle měnících se potřeb zákazníka
- viditelné znaky vývoje SW
  - **artefakty**
    - výpisy programů, dokumentace, data, zdrojové soubory
  - **procesy**
- charakteristiky výrobního procesu SW
  - srozumitelnost
  - spolehlivost
  - viditelnost
  - přijatelnost
  - robustnost
  - udržovatelnost
  - rychlost
  - podporovatelnost

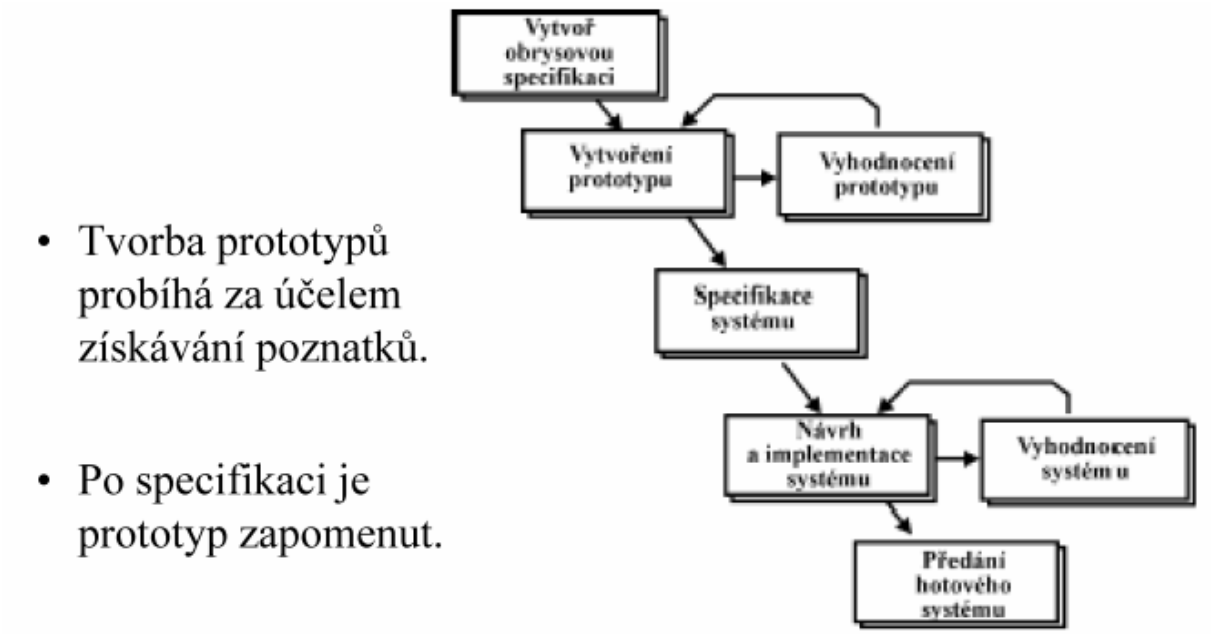
2. Životní cykly

- vodopád



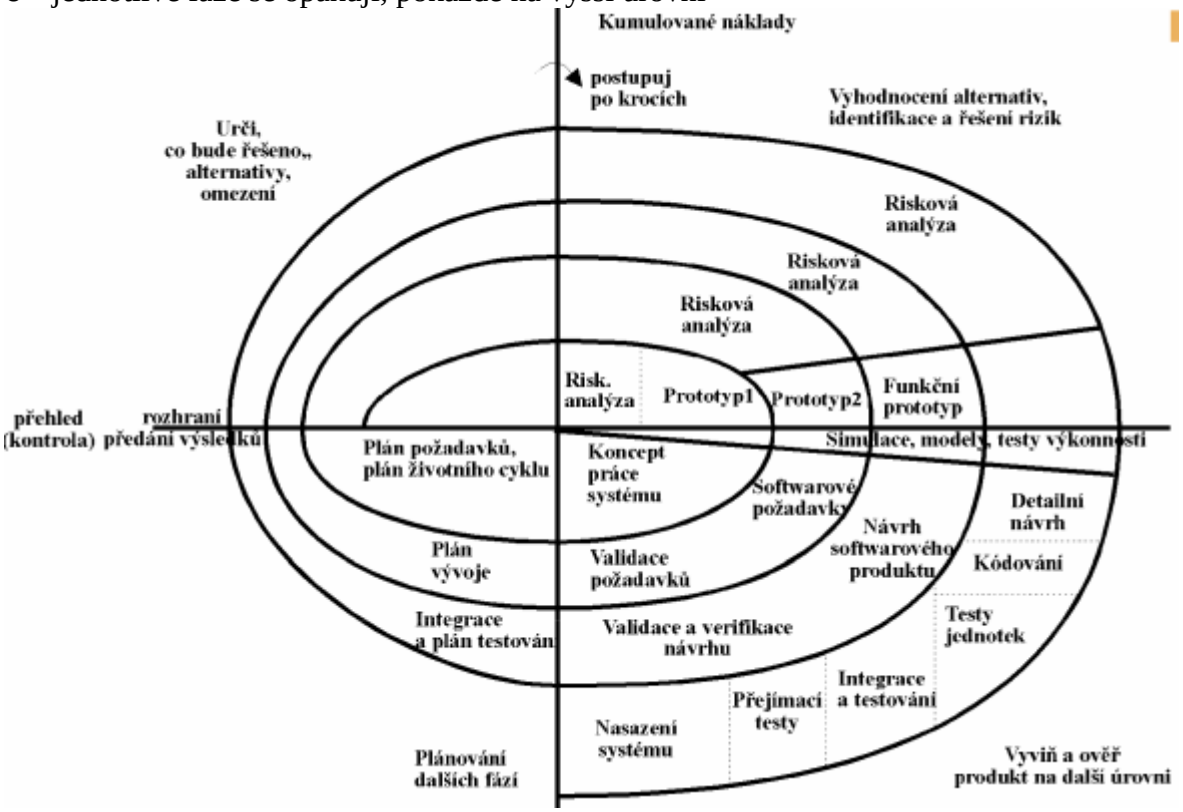
  - **problémy**
    - reálné projekty nedodržují pořadí etap
    - uživatel nedokáže na počátku formulovat přesné požadavky
    - zákazník musí být trpělivý, nevidí produkt během vývoje
    - pozdní odhalení nedostatků ohrožuje celý projekt
- iterativní životní cyklus
  - software je vyvíjen v iteracích, každá iterace je instancí vodopádu
  - snadnější upřesnění specifikací uživatelem
- inkrementální životní cyklus
  - podobný iterativnímu cyklu, jednotlivé iterace tvoří samostatné části systému odevzdávané postupně
  - 

- prototypový životní cyklus



- vyvíjí se jednotlivé prototypy, které se následně zahodí a vyvíjí se znovu
- díky prototypům si zákazník může ujasnit své požadavky

- spirálový cyklus
  - jednotlivé fáze se opakují, pokaždé na vyšší úrovni

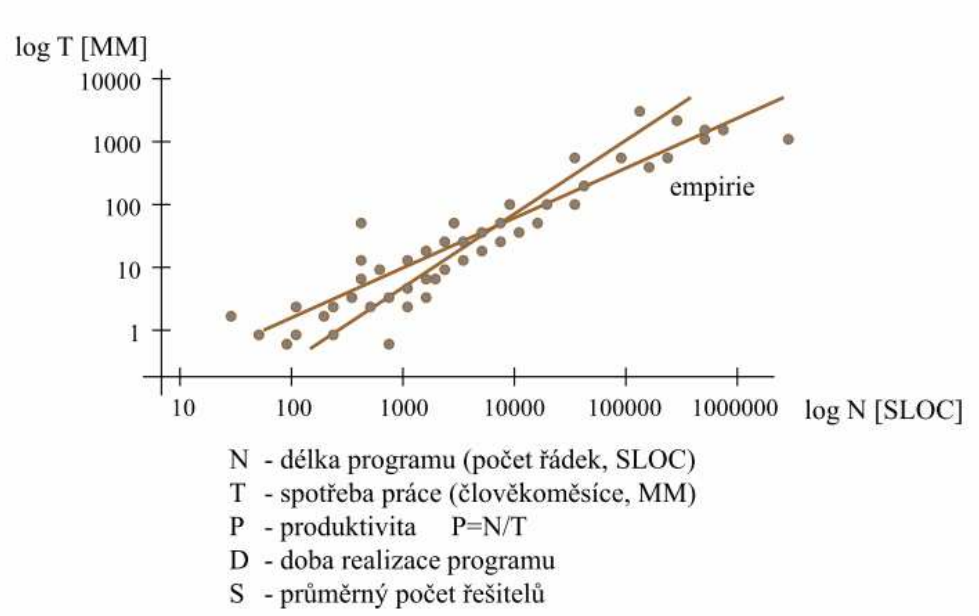


3. Lehmannovy zákony, Brooksův zákon

- zákon trvalé proměny
  - Systém používaný v reálném prostředí se neustále mění, dokud není levnější systém restrukturalizovat, nebo nahradit zcela novou verzí.
- zákon rostoucí složitosti
  - Při evolučních změnách je program stále méně strukturovaný a vzrůstá jeho vnitřní složitost. Odstranění narůstající složitosti vyžaduje dodatečné úsilí.

- zákon 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.
- zákon invariantní spotřeby práce
  - Celkový pokrok při vývoji projektů je statisticky invariantní. Jinak řečeno, rychlost vývoje programu je přibližně konstantní a nekoreluje s vynaloženými prostředky.
- zákon 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.
- Brooksův zákon
  - přidání řešitelské kapacity u zpožděného projektu může zvětšit jeho zpoždění (náklady na začlenění nového pracovníka jsou většinou větší než jeho přínos)

4. SW fyzika, Putnamova rovnice



N - délka programu (počet řádek, SLOC)
T - spotřeba práce (člověkoměsíce, MM)
P - produktivita P=N/T
D - doba realizace programu
S - průměrný počet řešitelů

- s rostoucí délkou programu se zvětšuje spotřeba práce
- s rostoucí délkou programu klesá produktivita programátorů

Putnamova rovnice

$$log\ P \mathrel{\hat{=}} a\ log\ (T/D^2) + b$$

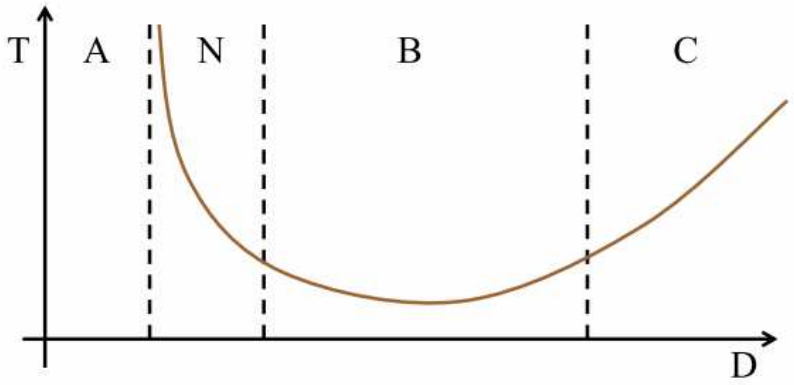
a, b - vhodné konstanty, z empirie => $a \cong -2/3$

$$P \mathrel{\hat{=}} d\ (T/D^2)^{-2/3} = d\ T^{-2/3}\ D^{4/3}$$

$$N \mathrel{\hat{=}} c\ T^{1/3}\ D^{4/3}$$

Délka programu, práce, doba řešení

Pracnost a doba řešení



A - nedosažitelná oblast
N - napjaté termíny
B - oblast stability
C - doba řešení je zbytečně dlouhá

- rozložení řešitelské kapacity – vrchol cca 40% projektu (zapojeno nejvíce pracovníků)

5. Cocomo

- slouží pro odhadování ceny SW

$$E = a \cdot F \cdot (KSLOC)^{b}$$

$$T = c \cdot E^{d}$$

a,b,c,d: parametry volené podle úrovně modelu a vývojového módu

E = úsilí (člověkoměsíce)
T = délka vývoje (měsíce)
Korekční faktor F = sada atributů ovlivňujících výpočet (atributy SW produktu, HW, vývojového týmu, produktu) – mohou "zlepšit" nebo "zhoršit" výpočet

- vývojové módy projektu
  - **organický mód** – malý tým, známá oblast, dobré podmínky
  - **přechodný mód** – větší projekt, více komunikace
  - **vázaný mód** – neznámá oblast, speciální hw, časté změny
- úroveň modelu
  - základní
  - střední
  - podrobná

COCOMO lze také použít pro odhad nákladů při modifikaci existujících aplikací

$$ESLOC = ASLOC \cdot (0{,}4\,DM + 0{,}3\,CM + 0{,}3\,IM) / 100$$

ESLOC - ekvivalentní počet SLOC
ASLOC - odhadnutý počet modifikovaných SLOC
DM - procento modifikace v návrhu
CM - procento modifikace v kódu
IM - integrační úsilí (procento původní práce)

COCOMO II

3 různé modely

- APM (Aplication Composition Model)
  pro projekty s použitím moderních nástrojů a GUI
- EDM (Early Design Model)
  pro hrubé odhady v úvodních etapách, kdy se architektura vyvíjí
- PAM (Post Architecture Model)
  pro odhady poté, co byla specifikována architektura

$$Úsilí = (multiplikátory\ okolí)[velikost]^{(faktory\ procesu)}$$

$$Plán = (multiplikátor)[Úsilí]^{(faktory\ procesu)}$$

6. Funkční body

- normalizovaná metrika SW projektu
- měří aplikační oblast, tj. výslednou funkcionalitu
- typy funkčních bodů:
  - **Externí vstupy** (EI - External Inputs) - co zadal uživatel
  - **Externí výstupy** (EO - External Outputs) - co vidí uživatel
  - **Externí dotazy** (EQ - External Enquiry) – co vidí uživatel poté co něco zadal (lol)
  - **Vnitřní logické soubory** (ILF - Internal Logical Files) – vnitří data systému
  - **Soubory vnějšího rozhraní** (EIF - External Interface Files) – dotazy na vnější systémy
- charakteristiky systému – 14 charakteristik, které mají vliv na systém (0 až 5)

Počet funkčních bodů
=
[0.65 + (0.01 x součet hodnocení charakteristik systému)]
x
[počet nepřizpůsobených funkčních bodů]

7. Zadávací dokumentace

- používá se při výběrovém řízení pro specifikaci poptávky a upřesnění požadavků na produkt
- obsah dokumentace
    - identifikační údaje zadavatele
    - předmět zakázky
    - požadavky na rozsah a kvalitu plnění
    - hodnota zakázky
    - kritéria hodnocení
    - variantní řešení
    - požadavky na prokázání kvalifikace uchazeče
    - požadavky na způsob zpracování nabídkové ceny
    - požadavky na způsob předložení

8. Smlouva na vývoj SW

- identifikace zadavatele a zhotovitele
- předmět smlouvy
- místo a doba plnění
- cena díla a platební podmínky
- práva a povinnosti smluvních stran
- licenční ujednání
- ochrana důvěrných informací
- záruka
- přílohy
    - definice požadavků na dílo
    - technické řešení provádění díla
    - autorská práva
    - čestné prohlášení

9. Servisní smlouva, SLA

- identifikace zadavatele a zhotovitele
- předmět smlouvy
- místo a doba plnění
- cena za provozní podporu a platební podmínky
- práva a povinnosti smluvních stran
- ochrana důvěrných informací
- přílohy
    - způsob řešení incidentů
    - způsob implementace nových změn
    - konzultace, školení

10. Projektová kalkulace

- udržování číselníků
    - stanoví se číselník interních nákladů pro jednotlivé role v týmu, cena za manday, používá se interní cena a cena, za kterou se prodává
    - **číselníky**
        - moduly projektu
        - role na projektu
        - katalog rolí
        - projektové oblasti
- seznam činností+pracnosti
- hw + licence
- seznam subdodávek

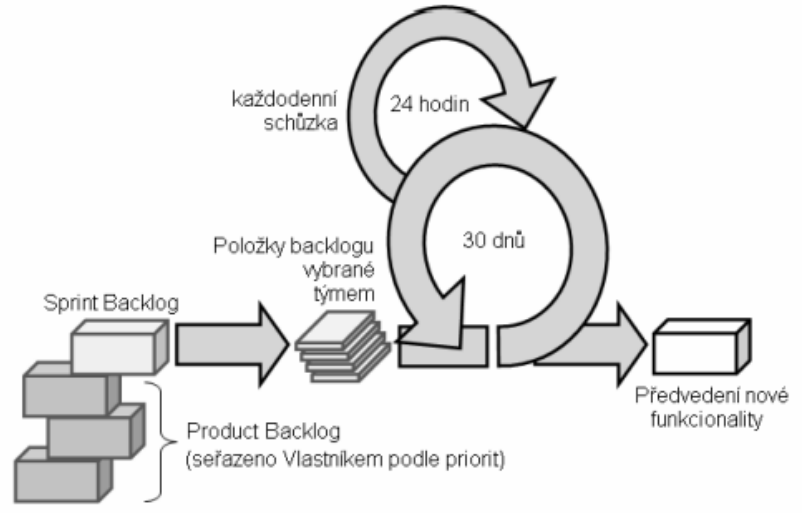
- cena podpory – většinou pomocí procenta z ceny vývoje
- rizika – co se na projektu může pokazit, pravděpodobnost X dopad, vytvoří se nějaký cenový buffer v celkové ceně

11. Metriky

- slouží ke sledování času, nákladů, kvality
- klasifikace metrik
  - **metriky produktu** – kód, funkčnost, dokumentace
  - **metriky procesů** – aktivity při vývoji
  - **metriky zdrojů** – HW, znalosti, lidé,…
- typy měření
  - **přímé (tvrdé)** – cena, úsilí, LOC, rychlost,…
  - **nepřímé (měkké)** – funkcionalita, kvalita, spolehlivost, udržovatelnost
- metriky orientované na velikost zdrojového kódu (LOC oriented)
  - snadné na použití a měření
  - jsou závislé na jazyce a konkrétním programátorovi
  - LOC, KLOC, chyby/KLOC, cena/KLOC, dokumentace/KLOC, komentáře/KLOC
- funkčně orientované metriky
  - měření pomocí funkčních bodů
  - chyby/FP, cena/FP, dokumentace/FP, FP/osoba/měsíc
- metriky složitosti
  - **Halsteadova metrika**
    - řeší se poměr slovní zásoby (počet typů operandů, operátorů) a délky programu
    - n1 = počet dostupných operátorů, n2 = počet dostupných operandů
    - N1 = počet použitých operátorů, N2 = počet požitých operandů
    - n = slovní zásoba, N = délka programu
    - *ideální délka* = n1 * log2 n1 + n2 * log2 n2
  - **McCabeovy metriky**
    - *cyklomatická složitost* – kolik existuje různých průchodů kódem, CS = počet větví + 1
    - CS nového kusu kódu větší než 10 je problematická, složité testování, více chyb
  - **McClureova metrika**
    - *složitost* = počet větvení + počet řídících proměnných
    - zohledňuje složitost rozhodovacích podmínek
- měření kvality SW
  - **správnost –** počty chyb, odchylky od specifikace/KLOC
  - **udržovatelnost –** cena za opravy, změny
  - **integrita**
  - **použitelnost** – potřebný čas na zaškolení, nárust produktivity při užívaní SW,…
- propojení
  - propojení modulů, kdo s kým komunikuje, snaha minimalizovat propojení
  - pro každý modul se počítá, jak moc je propojený s okolím
    - vstupní a výstupní data a pokyny (di,ci,do,co)
    - globální data a pokyny (gd, gc)
    - kolik modulů se volá z daného modulu a z kolika modulů je volán daný modul (w, r)
    - Mc = k/m (k = 1)
    - k = di + a*ci + do + b*co + gd + c*gc + w + r

12. Agilní a tradiční postup vývoje + SCRUM

- rysy agilního vývoje
  - **iterativní a inkrementální vývoj s krátkými iteracemi**

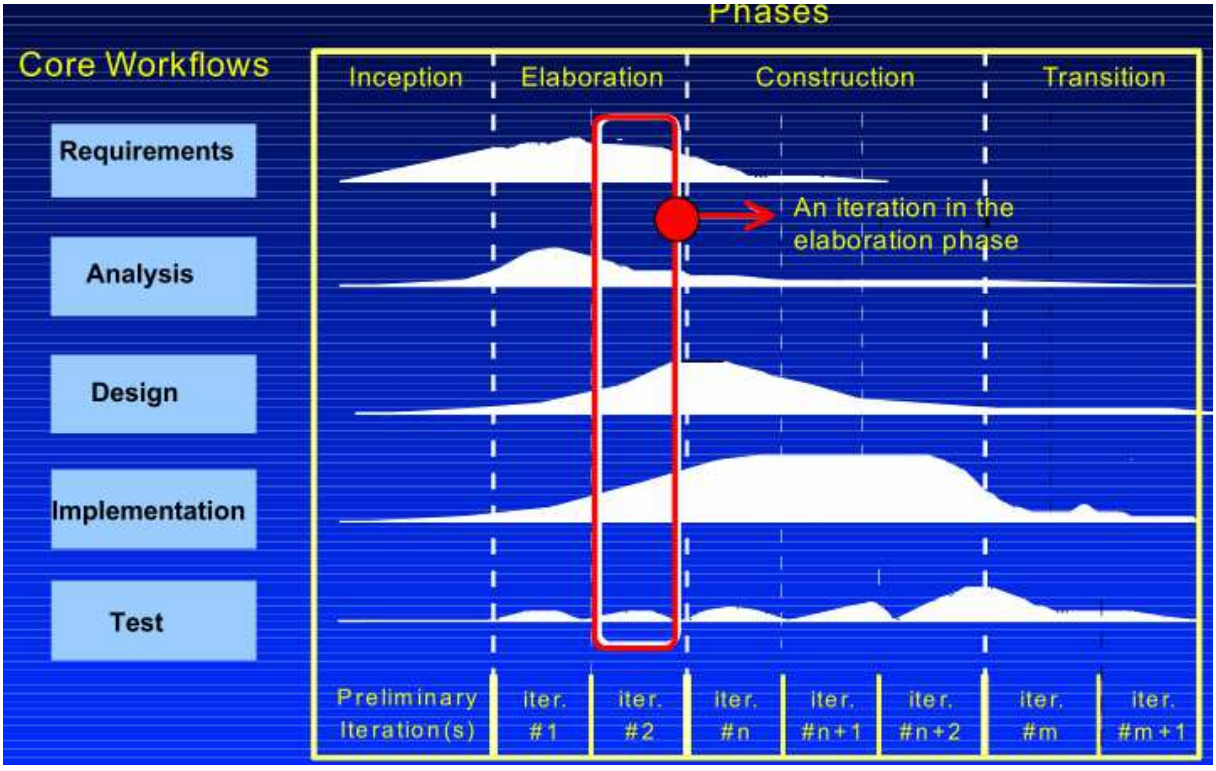
        - funkcionalita je dodávána po částech
    - **komunikace mezi zákazníkem a vývojovým týmem**
        - zákazník má možnost připomínkovat vývoj
    - **průběžné automatizované testování**
        - při každé změně je nutno provést kompletní sadu testů
- metodiky
    - **Extreme Programming (XP)**
        - všechny běžné postupy a praktiky dotahuje do extrému
        - vhodné pro malé týmy a projekty s nejasným nebo nebo rychle se měnícím zadáním
        - párové programování, nepřetržité testy, refaktorizace
        - rychlá zpětná vazba, malé přírůstky, jednoduchost, prostor pro změny, důraz na kvalitu práce
    - **Feature-Driven Development (FDD)**
        - vývoj po vlastnostech, rysech
        - *fáze*
            - 1. vytvoření celkového modelu
            - 2. vytvoření seznamu vlastností
            - 3. plánování podle vlastností
            - 4. návrh podle vlastností
            - 5. implementace podle vlastností
        - iterují se fáze 4 a 5, délka cyklu jsou 2 týdny
    - **SCRUM**
        - vývoj po sprintech, každý sprint má cca 1 měsíc
        - product backlog – seznam položek
        - sprint backlog – seznam položek, vybraných pro vývoj v daném sprintu
        - pravidelné každodenní schůzky vývojového týmu, kde se řeší, co bylo vyvinuto a co se bude dále vyvíjet
        - každý sprint zakončen předvedním demo-aplikace => zpětná vazba
        - *role v týmu*
            - **product owner –** určuje, co se bude v dalším sprintu implementovat včetně pořadí, komunikuje se zákazníkem
            - **scrum master** – řeší problémy programátorů, řídí vývoj, umožňuje programátorům soustředit se na práci
            - **stakeholders** – zákazníci, testeři
            - **managers** – pomáhají nastavit prostředí
        - 

    - **Test-Driven Development (TDD)**
        - důraz na testování
        - nestanovuje proces specifikace, návrhu a dokumentace
        - testy se píší dříve než kód
- rozdíly mezi agilním a tradičním přístupem
    - vývoj po malých částech VS. vývoj po velkých částech

    - proměnlivý plán vývoje VS. pevně stanovený plán a termíny
    - použití individuálních postupů a procesů VS. dodržování stanovených procedur a návodů
    - okamžitá zpětná vazba zákazníka VS. zákazník vidí výsledek často až na konci vývoje
    - neustále testování jednotlivých částí VS. testování v pozdějších fázích
- <u>shodné znaky agilního a tradičního přístupu</u>
    - snaha zajistit doručení kvalitního produktu, řízení procesu vývoje
    - specifikace požadavků, analýza zadání
    - rozdělení vývoje do určitých fází – specifikace, návrh, vývoj, testování
    - zajištění dokumentace k produktu
    - důraz na řízení týmu, komunikace uvnitř týmu

13. Unified Process

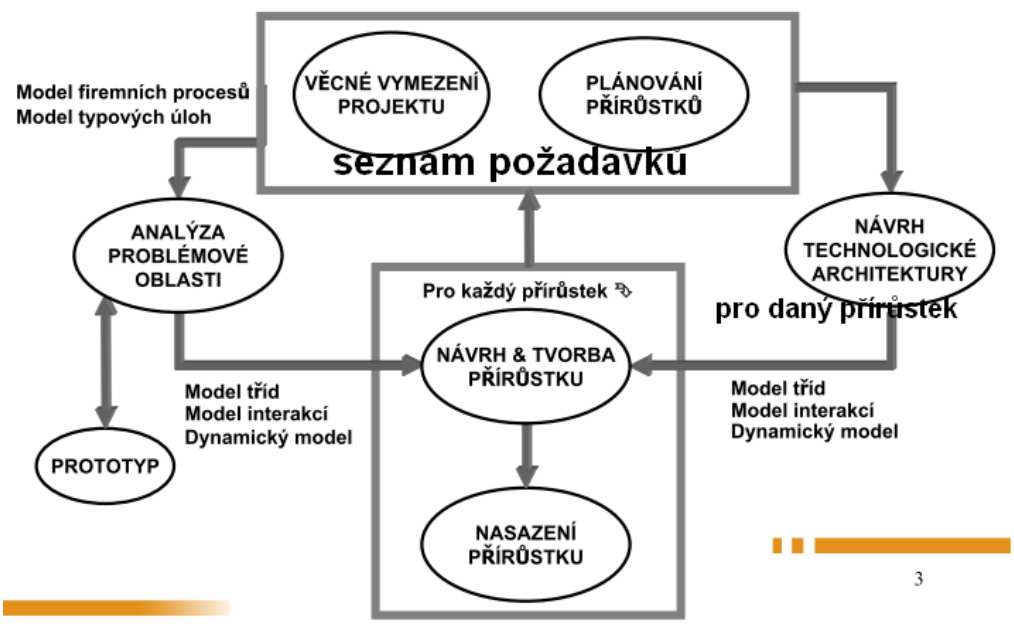
- generický proces vývoje SW
- iterativní a inkrementální
- řízen požadavky a případy užití, neustálé sbírání připomínek
- orientovaný na architekturu (dělení systému na komponenty)
- využití UML (zejména diagram případů užití a diagram tříd)
- <u>životní cyklus</u>
    - **zahájení**
        - zachycení požadavků, vytvoření bussines případu
    - **rozpracování**
        - plán projektu, zpřesnění požadavků, architektura systému
    - **konstrukce**
        - implementace produktu
    - **zavedení**
        - příprava produktu k nasazení, zaškolení, předání



14. Select perspective

- metodika určená pro modelování,vývoj a údržbu SW, vychází z use-case diagramu
- kombinace procesního modelování a objektového modelování
- UML diagramy a procesní mapy, proces = případ užití

- inkrementální postup, co přírůstek to use-case
- architektura systému založená na komponentách



- na základě hierarchie procesů se detekují případy užití (elementární procesy na nejnižší úrovni), které se dále rozkreslí a programují
- procesy => případy užití => třídy => komponenty

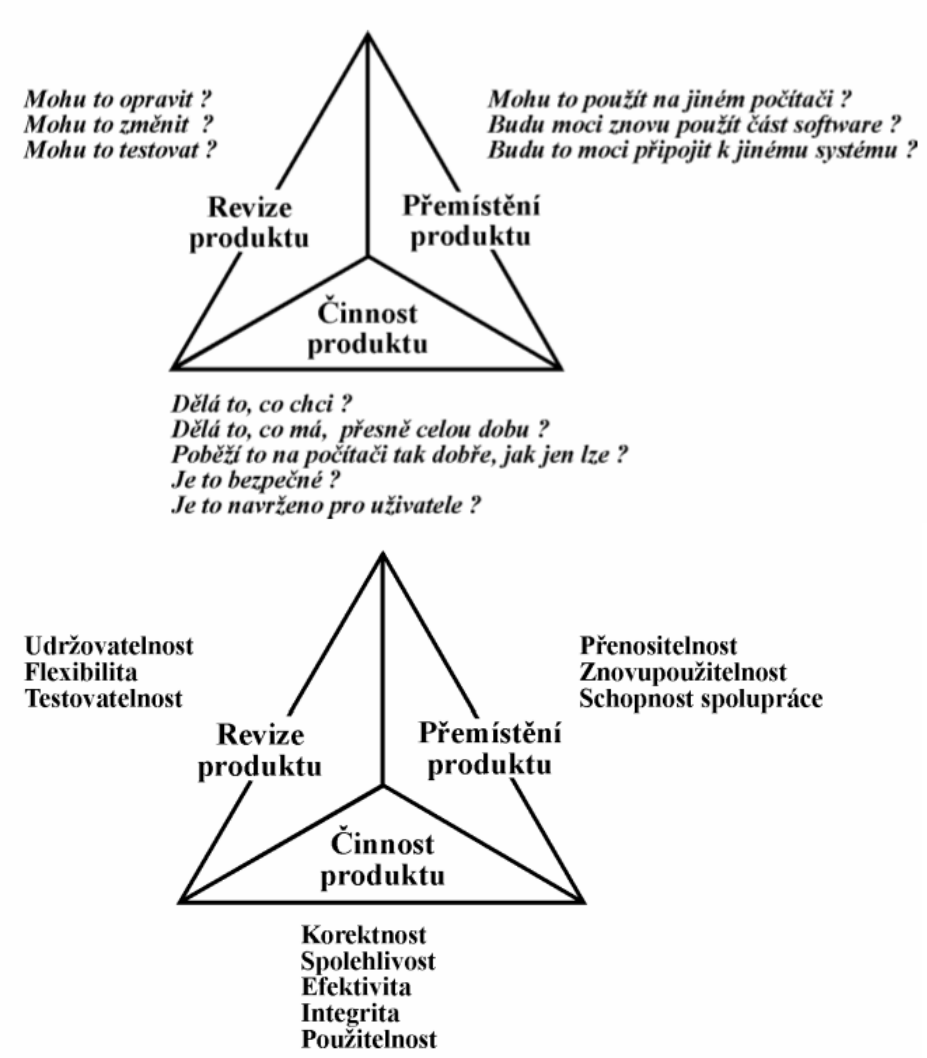
15. WebML

- metodika a nástroj pro modelování webů

- datový model
  - návrh datové struktury
  - konceptuální model podobný ERD
- hypertextový model
  - **submodely**
    - složení aplikace z jednotlivých stránek a složení stránek z jednotlivých elementů
    - navigační model, určuje propojení jednotlivých stránek
- prezentační model
  - konkrétní podoba webové aplikace, lze prezentovat jako XML

16. Kvalita – prvky SWQA

- aspekty kvality
  - odchylky od požadavků na SW
  - nedodržení standardů
  - odchylky od implicitních požadavků
- faktory kvality McCall



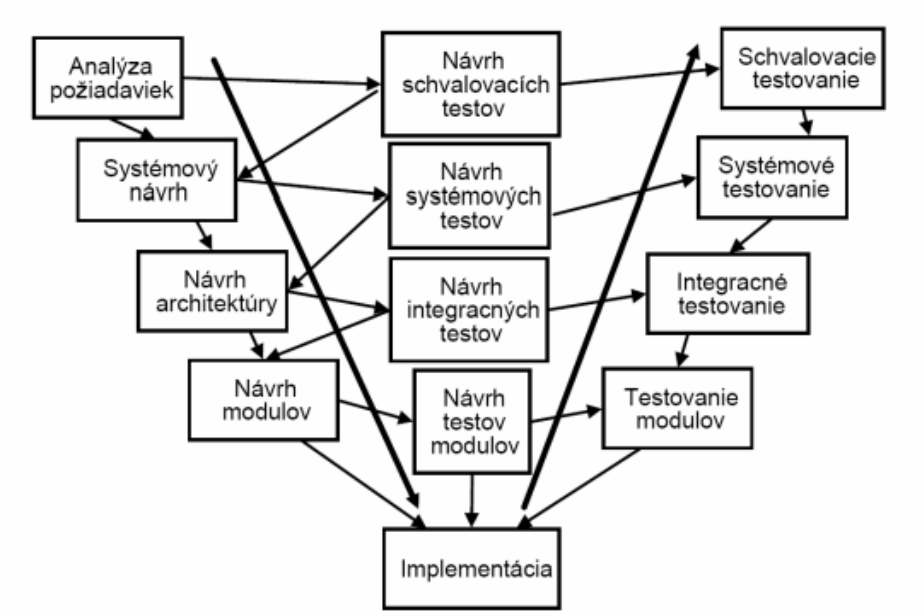
- IBM ortogonální klasifikace defektů (ODC)
  - **funkce -** chyba ovlivňující schopnosti uživatelů či rozhraní
  - **rozhraní -** chyba při interakci s ostatními komponentami
  - **ověřování -** chyba v logice programu při validaci dat a hodnot
  - **přiřazení -** chyba při inicializaci datové struktury nebo bloku kódu
  - **časování/serializace -** chyba, která zahrnuje časování sdílených a RT prostředků
  - **sestavení/balení/spojování -** chyba související se správou verzí a repozitářem projektu
  - **dokumentace -** chyba, která ovlivňuje publikace a návody
  - **algoritmus -** chyba, která se týká efektivity nebo správnosti algoritmu

- SQA
  - způsob monitorování kvality SW
  - oblasti SQA
    - **testování**
    - **inspekce, recenze**
    - **měření SW**
    - **standardy a procedury**
- přezkoušení SW
  - méně formální (konverzace, přezkoušení, prezentace**)**, více formální (prohlídka, recenze, inspekce)

17. Testy – jednotlivé typy testů

- spuštění programu s cílen nalézt chyby
- ukazuje i funkce a výkon
- testy provádí jiná osoba než vývojář

- V-procesní model
    - vodopádový model doplněný o testovací fáze



- validace – test proti specifikaci, zjišťuje se, zda program dělá, co má (black box)
- verifikace – test proti vnitřní činnosti, zjišťuje se, zda program dělá věci správně (white box)
- testování jednotek, modulů
    - většinou white box (může být i black box)
    - testování během vývoje a po dokončení modulu
    - použití testovacích driverů – simulují okolní části systému
- integrace a integrační testování
    - propojení jednotlivých modulů a funkcionalit, testování je souběžné s integrací
    - testuje se komunikace mezi jednotlivými moduly
    - testování vždy na konci inkrementu
    - **integrační postupy**
        - shora dolů
            - nejprve se implementuje jádro systému, neúplné části se simulují a postupně se nahrazují jednotlivými moduly
            - použití "stubs" (pahýly, protézy) – náhražkové objekty se shodným rozhraním
            - uživatel má přehled o aplikaci, ale nemůže používat jednotlivé funkce
        - zdola nahoru
            - nejprve se implementují individuální moduly, které jsou kombinovány do subsytémů
            - jako nadřazené objekty se používají drivery (použití driverů může být pracné)
            - chvíli trvá, než uživatel bude moci použít nějakou ukázku produktu

18. Inspekce, recenze

- odhalení chyb ve funkci, logice a implementaci SW
- ověřuje se, zda zkoumaný objekt splňuje požadavky
- recenze
    - vyjádření subjektivního pohledu na zkoumaný objekt
    - rozsáhlá diskuze o objektu
- inspekce
    - kontrolují se stanovené požadavky, zda jim objekt vyhovuje
    - kontrolní seznam
    - zajištění jednotného vývoje podle standardu
    - zapojení autora při inspekci, je důležité moderovat diskuzi inspektorů s autorem

19. Standardy, CMM

- hodnocení kvality výroby SW pomocí standardů
  - vyspělost organizace – CMM
  - systémy kvality – ISO
  - ocenění kvality - MBNQA
- CMM
  - měří vyspělost organizace
  - sestava úrovní podle vyspělosti
  - **Úroveň 1: Výchozí**
    - chaotický proces
    - nepředvídatelná cena, plán a kvalita
  - **Úroveň 2: Opakovatelný**
    - plán je pod kontrolou
    - cena a kvalita je proměnlivá
    - řízené požadavky, subkontrakty
    - zajištění kvality SW
  - **Úroveň 3: Definovaný**
    - orientace na kvalitu
    - spolehlivé ceny a plány
    - zlepšování organizačního procesu
    - školení
  - **Úroveň 4: Řízený**
    - řízení kvality
    - měření a kvantitativní řízení procesu výroby
  - **Úroveň 5: Optimalizující**
    - kontinuální investice směřující k automatizaci a zlepšení výrobního procesu
    - prevence chyb, řízené změny výrobních procesů